Urban Planning Processes

A Research Note[1]

Version 7. Incomplete Draft

Dines Bjørner

Fredsvej 11, DK-2840 Holte, Danmark
E–Mail: bjorner@gmail.com, URL: www.imm.dtu.dk/~db

Abstract. [2] We examine concepts of urban planning. That is, we emphasize, in this research note, the processes of urban planning.[3] In so doing we abstract from the information (the 'data') that urban planning is based on and results in. We distinguish between two kinds of urban planning processes: the basic, 'ab initio' (or base), process of determining "the general layout of the land (!)", and the derived, 'follow-up', processes focused on social and technological infrastructures. Base urban planning applies to descriptions of "the land": geographic, that is, geodetic, geotechnical, meteorological, and, in general, such information as are based in the given nature. Examples of derived urban plannings are such which are focused on humans and on social and technological artifacts: transport, electricity, water, waste, health care, schools, etc. This incomplete research note also discusses issues of urban planning project management, cf. Sect. 5.2, and urban planning document, cf. Sect. 5.3.

Contents

Correspondence and offprint requests to: Dines Bjørner, Fredsvej 11, DK 2840 Holte, Denmark
[2] This is Version 7 of the present document.

- Version 1 was issued on 28 April 2017, 13:15.
- Version 2 was issued on 30 April 15:36.
- Version 3 was issued on 1 May 17:29.
- Version 4 was issued on 9 May 2017, 13:58.
- Version 5 was issued 14 May 2017, 10:06 am.
- Version 6 was issued 19 May 2017, 9:42 am.
- Version 7 is now being worked on: 22 May 2017: 15:32 .

[3] This research note is being prepared for my stay at the *China Intelligent Urbanization Co-creation Center for High Density Region*, College of Architecture & Urban Planning (CAUP, http://en.tongji-caup.org/) at TongJi University, Shanghai, in September 2017 as hosted by Prof. Otthein Herzog.

1. Introduction

Urban planning is a technical and political process concerned with the development and use of land, planning permission, protection and use of the environment, public welfare, and the design of the urban environment,

including air, water, and the infrastructure passing into and out of urban areas, such as transportation, communications, and distribution networks.[4]

In this research note we shall try to understand one of the aspects of the domain underlying urban planning, namely that of some possible urban planning (development) processes. We are trying to understand and describe a domain, not requirements for IT for that domain and certainly not the IT (incl. its software).

1.1. A Triptych of Software Development

Before hardware and software systems can be designed and coded we must have a reasonable grasp of "its" requirements; before requirements can be prescribed we must have a reasonable grasp of "the underlying" domain. To us, therefore, software engineering contains the three sub-disciplines:

- domain engineering,
- requirements engineering and
- software design.

By a domain description we understand a collection of pairs of narrative and commensurate formal texts, where each pair describes either aspects of an endurant (i.e., a data) entity or aspects of a perdurant (i.e., an action, event or behaviour) entity.

1.2. On Domain Modeling

This research note is part of a series of experiments in domain modeling [23, 35, 41, 29, 27, 30, 17, 10, 60, 7, 62, 61, 6, 48, 4] – see Sect. 7.2 on Page 16. The concept of domain modeling is explored in a series of papers and reports [39, 37, 40, 36, 38, 32, 34, 22, 25, 12, 21, 46, 11]. The purpose of the present experiment, besides its hopeful contribution to *urban planning research & development* at TongJi University (Shanghai), is to explore modeling principles, techniques and tools not yet identified in [39, 37].

1.3. On Formality

We consider software programs to be formal, i.e., mathematical, quantities — rather than of social/psychological interest. We wish to be able to reason about software, whether programs, or program specifications, or requirements prescriptions, or domain descriptions. Although we shall only try to understand some facets of the domain of urban planning we shall eventually let such an understanding, in the form of a precise, formal, mathematical, although non-deterministic, i.e., "multiple choice", description be the basis for subsequent requirements prescriptions for software support, and, again, eventually, "the real software itself", that is, tools, for urban planners. We do so, so that we can argue, eventually prove formally, that the software **is correct** with respect to the (i.e., its) formally prescribed requirements, and that the software **meets customer**, i.e., domain users' **expectations** – as expressed in the formal domain description.

1.4. On Formal Notations

To be able to *prove formal correctness* and *meeting customer expectations* we avail ourselves of some formal notation. In this research note we use the `RAISE` [54] Specification Language, `RSL`, [53]. Other formal notations, such as `Alloy` [56], `Event B` [1], `VDM-SL` [49, 50, 52] or `Z` [63] could be used We choose `RSL` since it, to our taste, nicely embodies Hoare's concept of *Communicating Sequential Processes*, `CSP` [55]. In general we refer to the following set of textbooks [8]:

[4] https://en.wikipedia.org/wiki/Urban_planning

also published by QingHua University Press:

See [13, 14, 15, 18, 19, 20].

1.5. On the Form of This Research Note

The present form of this research note, as of 22 May 2017: 15:32 , is that of recording a development. The development is that of *trying to come to grips with what urban planning is*. We have made the decision, from an early start, that urban planning *"as a whole"* is a collection of one base and an evolving number of (initially zero) derived urban planning behaviours. Here we have made the choice to model the various behaviours of a complext of urban planning functions.

2. An Urban Planning System

We think of urban planning to be "dividable" into base urban planning, base_up_beh, and derived urban plannings, der_up_beh, where subindex i indicate that there may be several, i.e., $i \in \{d_1, d_2, ..., d_n\}$, such derived urban plannings. We think of base urban planning to "convert" physical (geographic, that is, geodetic, geotechncal, meteorological, etc.) information about the land area to be developed into a base plan, that is, cartographic, cadestral and other such information (zoning, etc.). And we think of derived urban planning to "convert" base plans into technological and/or societal plans. Technological and societal urban planning concerns are typical such as transport, electricity, water, waste, health care, schools, etc.Each urban planning *behaviour*, whether 'base' or 'derived', is seen as a *sequence* of the application of "the same" urban planning *function*, i.e., an urban planning *action*. Each urban planning action takes a number of information *argument*s and yield information *result*s. The base urban planning behaviour may **start** one or more derived urban planning behaviours, der_up_beh$_i$, at the end of "completion" of a base urban planning *action*. Let (indices) $\{d_1, d_2, ..., d_n\}$ identify a set of separate derived urban plannings, each concerned with a distinct, reasonably delineated technological and/or societal urban planning concern. During a base urban planning development the actions start any of these derived urban plannings once. Thus we think of urban planning as a system of a single base urban planning process (i.e., behaviour), base_up_beh, which "spawns" zero, one or more (but a definite number of) derived urban planning processes (i.e., behaviours), der_up_beh$_i$. A derived urban planning processes, der_up_beh$_i$, may themselves start other derived urban planning processes, der_up_beh$_j$, der_up_beh$_k$, ..., der_up_beh$_\ell$. Figure 1 on the next page is intended to illustrate the following: At time $t0$ a base urban planning is started. At time $t1$ the base urban planning initiates a number of derived urban development, $D1, ..., Di$. At time $t2$ the base urban planning initiates the Dj derived urban planning. At

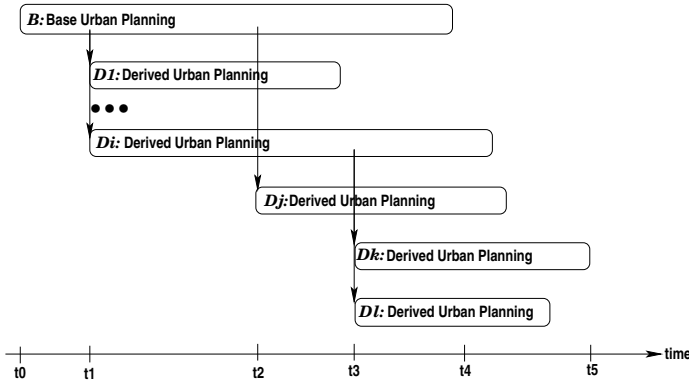

Fig. 1. An Urban Planning Development

time $t3$ the derived urban planning Di initiates two derived urban plannings, Dk and $D\ell$. At time $t4$ the base urban planning ends. And at time $t5$ all urban plannings have ended. Urban planning actions are provided with "input" in the form of either geographic, geodetic, geotechnical, meteorological, etc., information, b_geo:bGEO[5], or auxiliary information, b_aux:bAUX, or requirements information, b_req:bREQ. The auxiliary ("management") information is such as *time and date, name (etc.) of information provider, "trustworthiness" of information,* etc. The requirements information serves to direct, to inform, the urban planners towards *what kind of urban plan* is desired.

3. Base Urban Planning

We begin this section with abstractions of the, perhaps, two most important aspects of urban planning, such as it may be seen by its individual practitioners: the information being handled: the "input", so-to-speak, to urban planning function(s) and these urban planning function(s). In two sections, in-between the information and the function sections (3.1 and 3.4), we very briefly discuss the iterative nature of urban planning, Sect. 3.2 on the following page, and initial values, Sect. 3.3, of the various information values.

3.1. Urban Planning Information Categories

Among the arguments of urban planning are

1 information, bGEO, about the geographic area subject to planning: its geodetic "make-up", its geotechnical and meteorological properties, etc.,
2 related, but not geographic, information, bAUX,
3 and some requirements, bREQ.

type
1 bGEO
2 bAUX
3 bREQ

Among results of urban planning are

4 "the plan" (or "plans"), bPLA,
5 and possibly some other (ancillary) documents, bANC.

[5] The b_ value prefixes and the b type prefixes shall designate base urban planning entities.

type
4 bPLA
5 bANC

For this research note we shall leave the bGEO, bAUX, and bREQ argument types and the bPLA, and bANC result types further undefined[6] Typically bGEO would be be heavily text-annotated graphical documents which would show "the lay of the land", its geodetic[7], its geotechnical[8], and its meteorological[9], properties. Typically bAUX would then be some mixture of graphical and text documents that would "explain", in usually informal, casual ways, some of the relations between the geographic documents, when they were recorded, how they were vetted, their accuracy, etc. Typically bREQ would be informal, casual text documents, perhaps including pseudo-technical terms, which expresses expectations as to what the "powers-to-be" might consider suitable urban plans[10] and urban designs[11].

3.2. The Iterative Nature of Urban Planning

We take it that urban planning proceeds in "cycles":

6 In each cycle the base urban planning function, base_up_fct, is applied to an input argument triple, (b_geo,b_aux,b_req):(bGEO×bAUX×bREQ):bTRI, of "fresh" geodetic/geotechnical/meteorological (etc.), auxiliary and requirements information.

type
6 bTRI = bGEO × bAUX × bREQ

7 Each cycle, that is, each application of base_up_fct, results in a "most recent", not necessarily "final", plan and ancillary information, (b_pla,b_anc):bPLA×bANC:bRES.

type
7 bRES = bPLA × bANC

8 But, to "drive" the urban planning process, base_up_beh, towards "final", that is, an adequately satisfactory plan etc., the urban planning function, base_up_fct, need also be provided with the results of the previous iteration's result — which we take to be a ("quintuplet") pair of an (i.e., the "previous") "input" triple and the previous result pair.

type
8 bQUI = bTRI × bRES

We shall refer to the input argument triple as 'the triplet' and the "driver" quintuplet (also) as a resumption. The above decisions on triplet arguments and quintuplet resumptions, including the latter's "feedback" to a next iteration function invocation is motivated as follows. We think of each invocation, i.e., step, of the urban planning function to "apply" itself to a small fragment of urban planning. Each such "small" step is to result in useful contributions to the evolving urban plan. The ancillary information emerging from each step informs about which aspects of urban planning was pursued in that step: where, in the plans, the outcome of those analysis and plan development can be seen. The reason for small step invocations are to allow ongoing reviews (not shown here), to pass on the intermediary results to other urban planning developments, etc. The decision to "feed" back "records" of the entire state of urban planning development motivated by the need for these "small step" invocations to analyse the ongoing, full state.

[6] Understanding the bGEO, bAUX, bREQ, bPLA and the ANC types is a major urban planning issue.
[7] https://en.wikipedia.org/wiki/Geodesy
[8] https://en.wikipedia.org/wiki/Geotechnical_investigation
[9] https://en.wikipedia.org/wiki/Meteorology
[10] https://en.wikipedia.org/wiki/Urban_planning
[11] https://en.wikipedia.org/wiki/Urban_design

3.3. Initialisation

Urban planning proceeds in iterating from initial

9 geographic, auxiliary and requirements information, as well as
10 (usually "empty") plans and ancillaries.

We extend the notion of initial values to

11 triplet arguments,
12 result pairs, and
13 "quintuplet" argument/result pairs.

towards such results (plans and ancillaries) that are deemed satisfactory.

```
value
9  b_geo_init: bGEO, b_aux_init:bAUX, b_req_init:bREQ
10 b_pla_init: bPLA, b_anc_init:bANC
11 b_tri_init: bTRI = (b_geo_init,b_aux_init,b_req_init) assert: b_tri_fit(b_tri_init,b_tri_init)
12 b_res_init: bRES = (b_pla_init,b_anc_init)
13 b_qui_init: bQUI = (b_tri_init,b_res_init)
```

We refer to Item 23 on the next page for an explanation of the b_tri_fit predicate.

3.4. A Simple Functional Form

14 The base urban planning *function*, base_up_fct, thus applies to

- (i) a "most recent" triplet of geographic, auxiliary and requirements information, and to
- (ii) a "past quintuplet", a resumption, that is, pair of geographic, auxiliary and requirements information as well as plan and ancillary information and yields such a resumption "quintuplet" pair of a triplet and a pair.

15 The application of base_up_fct to such arguments, i.e., base_up_fct(b_geo,b_aux,b_req)(b_qui) yields a "quintuplet" result, a resumption, ((b_geo″,b_aux″,b_req″),(b_pla″,b_anc″)).

We "explain" the relations between "input" arguments and "output" (**as**) results:

16 The "input" argument (geo,aux,req) is "carried forward", (b_geo″,b_aux″,b_req″), to be redeposited as part of the result.
17 The main part of the result, (b_pla″,b_anc″), is related, $\mathcal{P}$, to the input argument including the previous "result", the resumption.

```
14  base_up_fct: bTRI → bQUI → bQUI
15  base_up_fct(b_tri)(b_qui) as (b_tri′,(b_pla,b_anc))
16      b_tri = b_tri′ ∧
17      P_base(b_tri)(b_qui)(b_pla,b_anc)
```

For the time being we shall leave the base urban planning function, base_up_fct, that is, $\mathcal{P}_{\text{base}}$, uninterpreted.

3.5. Oracles and Repositories

Oracles are simple behaviours that provide functions (via behaviours) with information. Repositories are simple behaviours that functions (via behaviours) provide with information and which can then "reproduce" this information to continued behaviours (and functions).

3.5.1. The Base 'Input' Oracle

An urban planning oracle, when so requested, will select some information – usually in some non-deterministic fashion, and usually subject to some constraint – and present this information to the requestor, i.e., an urban planning behaviour. In this section, i.e. Sect. 3.5.1, we shall deal with one specific oracle, b_tri_beh: one that "assembles" triplets, b_tri, of geographic, b_geo:bGEO, auxiliary, b_aux:bAUX, and requirements, b_req:bREQ, information. We introduce a pair of specification components:

18 a channel, b_tri_ch, over which a base urban planning behaviour,base_up_beh, offers to receive triplets, b_tri:bTRI, from an oracle, b_tri_beh,
19 and the oracle, b_tri_beh, which "remembers" its most recently communicated triplet[12].

```
channel
18  b_tri_ch:bTRI
value
19  b_tri_beh: (bGEO × bAUX × bREQ) → out b_tri_ch  Unit
```

20 The oracle assembles (b_geo:bGEO, b_aux:bAUX, b_req:bREQ) a base triplet which satisfies a predicate b_tri_fit(tri,(b_geo,b_aux,b_req)) – see Item 23.
21 That triplet is offered, b_tri_ch ! (b_geo,b_aux,b_req), to the base urban behaviour –
22 whereupon the oracle resumes being the oracle, now, however, with the recently assembled base triplet as its resumption.

```
19  b_tri_beh(tri) ≡
20      let b_geo:bGEO, b_aux:bAUX, b_req:bREQ • b_tri_fit(tri,(b_geo,b_aux,b_req)) in
21      b_tri_ch ! (b_geo,b_aux,b_req) ;
22      b_tri_beh(b_geo,b_aux,b_req)
19      end
24  pre: b_tri_fit(tri,tri)
```

23 The fitness predicate, b_tri_fit(tri,tri′), checks whether a "newly" assembled base triplet, tri, stands in the relation $\mathcal{P}$(tri,tri′) to a a similar base triplet, tri′.
24 The fitness predicate holds for b_tri_fit(tri,tri).
25 The oracle, b_tri_beh, is initialised with the initial triplet value b_tri_init, cf. formula Item 11 on the previous page.

```
23  b_tri_fit: bTRI × bTRI → Bool
23  b_tri_fit(tri,tri′) ≡ P(tri,tri′)

23  b_tri_beh(b_tri_init): assert: b_tri_fit(b_tri_init,b_tri_init)
```

3.5.2. The Base Resumption Repository

The "quintuplet" pair of an (i.e., the "previous") "input" triple and the previous result pair, ((b_geo:bGEO,-b_aux:bAUX,-b_req:bREQ),(b_pla:bPLA,b_anc:bANC)) — a "quintuplet" which is also the result of each urban planning action — is thought of as residing in a *repository* behaviour, qui_beh, which "receives" (b_qui_ch?) "quintuplets" from the urban planning behaviour, or "offers" (b_qui_ch!(b_qui)) such to the urban planning behaviour.

26 There is therefore a channel, quin_ch, between the urban planning behaviour and the "quintuplet" behaviour,
27 quin_beh.
28 It either

[12] The oracle is initialised with b_tri_beh(geo_init,b_aux_init,b_req_init).

29 accepts or
30 offers quintuplets.

```
channel
26   b_qui_ch:bQUI
value
27   b_qui_beh: bQUI → in,out b_qui_ch  Unit
27   b_qui_beh(b_qui) ≡
29       b_qui_beh(b_qui_ch?)
28       []
30       b_qui_ch!(b_qui) ; b_qui_beh(b_qui)
```

3.6. A Simple Behavioural Form

Urban planning, however, is a time-consuming "affair". So we model it as a behaviour.

31 The base_up_beh_0[13] behaviour takes no argument, **Unit**, avails itself of the input channel for obtaining proper input, b_tri and b_qui, for the base urban function, base_up_fct, and output channel, for depositing a resumption, and (then) "goes on forever", as indicated by **Unit**.
32 The simple (version of the) base_up_beh_0 behaviour
33 obtains the base triplet and the base resumption information,
34 performs the base_up_fct planning function and
35 provides its result, a resumption, to the base quintuplet repository –
36 whereupon it reverts to being base_up_beh_0.

```
value
31   base_up_beh_0: Unit → in b_tri_ch out b_qui_ch Unit
32   base_up_beh_0() ≡
33       let (b_tri,b_qui) = (b_tri_ch?,b_qui_ch?) in
34       let b_qui = base_up_fct(b_tri)(b_qui) in
35       b_qui_ch ! b_qui end end ;
36       base_up_beh_0()
```

The base_up_beh_0 behaviour repeatedly "performs" urban planning, "from scratch", as if new geographical, auxiliary and requirements information was "new" in every re-planning — "ad infinitum" ! We now revise base_up_beh_0 into base_up_beh_1 — a behaviour "almost" like base_up_beh_0, but one which may terminate.

37 base_up_beh_1
38 first behaves like base_up_beh_0 (Items 33–35)
39 then checks whether the obtained base resumption is satisfactory, that is, is OK as an end-result of base urban planning.
40 If so then base_up_beh_1 terminates,
41 else it resumes being base_up_beh_1.

```
value
37   base_up_beh_1() ≡
38       let b_qui = b_base_up_fct(b_tri_ch?)(b_qui_ch?) in b_qui_ch!b_qui ;
39       if b_qui_satisfactory(b_qui)
40           then skip
41           else base_up_beh_1() end
37       end

39   b_qui_satisfactory: bQUI → Bool
```

[13] As there will be several versions, from simple towards more elaborate, of the base_up_beh behaviour, we index them.

The b_qui_satisfactory predicate inquires the base quintuplet, b_qui, as for its suitability as a final candidate for an urban plan[14].

4. Derived Urban Plannings

4.1. Derived Urban Plan Indices

We think of *base* urban planning function, modeled by base_up_fct, as being concerned with the overall "division" of the geographical area, land and water, into zones for building, recreation, and other. Aggregations of these zones, one, more or all (usually several), can then be further "[derive] planned" into

- (d_1) light, medium and heavy industry zones,
- (d_2) mixed shopping and residential zones,
- (d_3) apartment bldg. zones,
- ..., etc., etc.,
- (d_{m-1}) villa zones, and
- (d_m) recreational zones.

Additional forms of derived plannings are:

- (d_{m+1}) transport,
- (d_{m+2}) electricity supply,
- (d_{m+3}) water supply,
- (d_{m+4}) waste management,
- (d_{m+5}) health care,
- (d_{m+6}) fire brigades,
- ..., etc., etc.,
- (d_n) schools.

We refer to the d_i's as derived urban plan indices.

42 We think of this variety of "derived" plannings thus as indexed as hinted at above,
43 and dups as the set of all indices.

type
42 DP == {|d_1,d_2,...,d_n|}
value
43 dups:DP-**set** = {d_1,d_2,...,d_n}

4.2. A "Reservoir" of Derived Urban Planning Indices

44 To secure that at most one derived planning is initiated we introduce a global variable, dps_var, initialised to an empty set of derived planning tokens and updated with the addition of selected DP tokens.

variable
44 dps_var:DP-**set** := {} **comment** dps_var *denotes a reference*

4.3. A Derived Urban Planning Index Selector

45 A function, sel_dps, selects zero, one or more DP "fresh" indices, that is, DP tokens that have not been selected before.

value
45 sel_dps: **Unit** → DP-**set**
45 sel_dps() ≡ **let** dps:DP-**set**•dps⊆dups \ **c** dps_var **in** dps_var := **c** dps_var ∪ dps; dps **end**
comment
44 [**c** *denotes a contents-taking operator*]

We shall revise the above selector in Sect. 4.10.3 on Page 14.

[14] The b_qui_satisfactory argument, b_qui, embodies not only that plan, but also the basis for its determination.

4.4. The Derived Urban Plan Generator

46 We therefore edit the base_up_beh_1 behaviour slightly by inserting, "in parallel" (‖) with the "resumption" of base_up_beh_1 (cf. Item 41 on Page 9), an internal non-deterministic choice behaviour, der_up_beh$_i$()[15], which selects zero, one of more DP tokens, and initiates corresponding derived planning behaviours, der_up_beh$_i$(), as well as their corresponding "input" triplet oracles, d_tri_beh$_i$() — but only at most once. These derived planning behaviours, der_up_beh$_i$, and "input" triplet oracles, d_tri_beh$_i$() are like base_up_beh_1, respectively b_tri_beh, only now they are "tuned" to the specific derived planning issues (i.e., $_i$).

```
value
46   der_up: Unit → Unit
46   der_up() ≡ let dps = sel_dps() in ‖{der_up_beh_i()‖d_tri_beh_i()|i:DP•i ∈ dps} end
```

We shall introduce the der_up_beh$_i$ and d_tri_beh$_i$ behaviours below.

4.5. The Revised Base Urban Planning Behaviour

We "take over" the basic structure and definition ("contents") of the urban planning function and behaviour from that of the base versions.

47 We think of zero, one or more derived plannings (der_up_beh$_1$, der_up_beh$_2$, ..., der_up_beh$_n$) being initiated after some stage of *base* function, base_up_fct, has concluded.

```
value
37'' base_up_beh_d() ≡
41''    let b_qui=base_up_fct(b_geo_ch?,b_aux_ch?,b_req_ch?)(b_qui_ch?) in b_qui_ch!b_qui ;
39''    if base_satisfactory(b_qui_ch?)
37''       then skip
38''       else
47            der_up() ‖
40''          base_up_beh_d() end end
```

4.6. The Derived Urban Planning Functions

An important form of information for each derived urban planning function is the resumption, i.e., the quintuplet information from the base urban behaviour: bQui.

48 The new forms of information are: the derived urban planning auxiliary, dAUX$_i$, and derived urban planning requirements information, dREQ$_i$, as well as the derived urban planning plans, dPLA$_i$, and their ancillary information, dANC$_i$.

49 The primary arguments for the derived urban planning function, base_up_fct, is therefore a derived triplet of the base urban planning "quintuplet", b_qui:bQUI, the derived urban planning auxiliary information, d_aux$_i$:dAUX$_i$, and the derived urban planning requirements information, d_req$_i$:dREQ$_i$,

50 The result of derived urban planning function, der_up_fct, as for the base urban planning function, base_up_fct, is that of a "quintuplet", also a resumption, dQUI$_i$, of the three primary arguments and

51 the result, a pair of a derived plan, d_pla$_i$, and derived ancillaries, d_anc$_i$.

52 As for the base urban planning function, base_up_fct, it has a secondary, derived "quintuplet" argument (which, as for base_up_fct, helps "kick-start" urban planning). This second argument is the result of a previous application of the der_up_fct.

[15] When behaviour and function invocations where the names of these behaviors or functions names are prefixed with der_, e.g., der_name, and are indexed by some $_i$, i.e., der_name$_i$, then we mean the invocation of one specific i indexed behaviour or function from the indexed set of such, as defined by their behaviour and function definitions, see below.

53 The derived urban planning function $\mathsf{der\_up\_fct}_i$ signature is therefore that of a function from a triplet of a most recent base quintuplet, derived urban planning auxiliary and derived urban planning requirements information to functions from derived "quintuplet" arguments to derived "quintuplet" results.

54 The triplet argument, $\mathsf{d\_tri}_i$, and the first part of the result, also a triplet, $\mathsf{d\_tri}'_i$, are the same.

55 The derived urban planning function $\mathsf{der\_up\_fct}_i$ is further characterised by a predicate, $\mathcal{P}_{\mathsf{der}_i}$, which we leave further undefined.

type
48 dAUX_1, dAUX_2, ..., dAUX_n
48 dREQ_1, dREQ_2, ..., dREQ_n
48 dPLA_1, dPLA_2, ..., dPLA_n
48 dANC_1, dANC_2, ..., dANC_n
49 dTRI_i[16]= $\mathsf{bQUI}\times\mathsf{dAUX}_i\times\mathsf{dREQ}_i$ [i:DP•i∈dups]
51 $\mathsf{dRES}_i = \mathsf{dPLA}_i \times \mathsf{dANC}_i$ [i:DP•i∈dups]
50 $\mathsf{dQUI}_i = \mathsf{dTRI}_i\times\mathsf{dRES}_i$ [i:DP•i∈dups]
value
52 $\mathsf{der\_up\_fct}_i$[17]: $\mathsf{dTRI}_i \to \mathsf{dQUIm} \to \mathsf{dQUI}$ m: i:DP
53 $\mathsf{der\_up\_fct}_i(\mathsf{d\_tri}_i)(\mathsf{d\_qui}_i)$ **as** $(\mathsf{d\_tri}'_i,\mathsf{d\_res}_i)$
54 $\mathsf{d\_tri}_i = \mathsf{d\_tri}'_i \wedge$
55 $\mathcal{P}_{\mathsf{der}_i}(\mathsf{d\_tri}'_i,\mathsf{d\_res}_i)$

55 $\mathcal{P}_{\mathsf{der}_i}$: $\mathsf{dTRI}_i \times \mathsf{dRES}_i \to$ **Bool**

4.7. The Derived Urban Planning Behaviour

56 We think of zero, one or more derived plannings ($\mathsf{der\_up\_beh}_{i_1}$, $\mathsf{der\_up\_beh}_{i_2}$, ..., $\mathsf{der\_up\_beh}_{i_m}$) being initiated after some stage of the *der_up_fct*$_i$ function has concluded.

value
37''' $\mathsf{der\_up\_beh}_i() \equiv$
41''' **let** d_qui=$\mathsf{der\_up\_fct}_i$(b_geo_ch?,b_aux_ch?,b_req_ch?)(b_qui_ch?) **in** d_qui_ch[i]!d_qui ;
39''' **if** $\mathsf{der\_satisfactory}_i$(d_qui_ch[i]?)
37''' **then skip**
38''' **else**
56 der_up() ‖
40''' $\mathsf{der\_up\_beh}_i$() **end end**

4.8. The Derived Resumption Repository

4.8.1. The Consolidated Derived Resumption Map

57 The derived urban planning functions (and thus behaviours) operate, not on simple resumptions, as do the base urban planning functions (and behaviours), but on the aggregation of all derived functions' (etc.) quintuplets, that is, an indexed set of quintuplets – modeled as a derived resumptions map.

type
57 $\mathsf{dQUIm} = \mathsf{DP} \xrightarrow[m]{} \mathsf{dQUI}_i$

4.8.2. The Consolidated Derived Resumption Repository Channel

58 Communications between the individual derived urban planning behaviours and the consolidated derived resumption repository are via an indexed set of channels communicating derived resumptions maps.

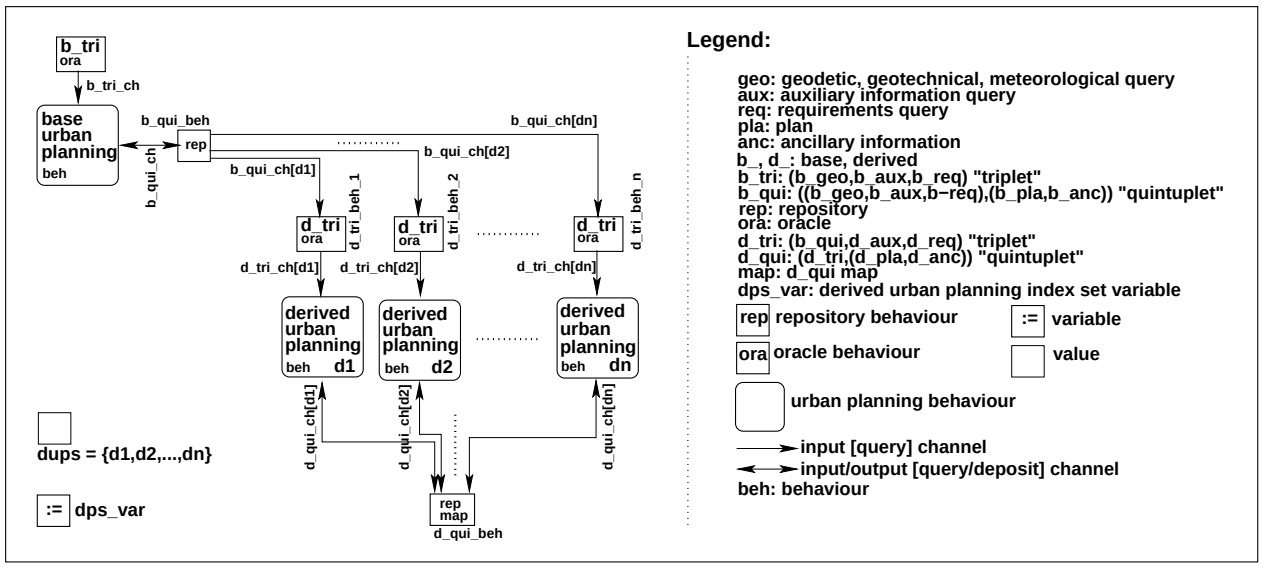

Fig. 2. An Urban Planning: n+1 Planning Behaviours, 2 Repository Behaviours, n+1 Oracles, a Variable, a Value and $3n$+2 Channels

channel
58 {d_qui_ch[i]:dQUIm|i:DP• i ∈ dups}

4.8.3. The Consolidated Derived Resumption Repository

59 The consolidated derived resumption repository behaviour either ([]) updates its state map with recieved individual derived resumptions, or offers the entire such state maps to whichever derived urban planning behaviour so requests.

value
59 d_qui_beh: dQUIm → **in,out** der_qui_ch[i] **Unit** i:DP
59 d_qui_beh(d_qui_m) ≡ d_qui_beh(d_qui_m†[i↦d_qui_ch[i]?]) [] d_qui_ch[i]!(d_qui_m) ; d_qui_beh(d_qui_m)

4.8.4. Initial Consolidated Derived Urban Plannings

value
d_qui_m = [d_1 ↦ init_d_qui$_1$, ..., d_n ↦ init_d_qui$_n$]

4.8.5. Initialisation of The Derived Quintuplet Oracle

As for base oracle and repository behaviours we initialise the derived quintuplet oracle:

der_qui_beh(init_d_qui_m)

4.9. A Visual Rendition of Urban Planning Development

The urban planning domain, when "operating at full speed", consists of the base urban planning behaviour (i.e., project), zero, one or more derived urban planning behaviours, each of the latter initiated by either the base urban planning project or a derived urban planning project. See Fig. 2. To provide the urban planning project with information there is the geodetic, geo-technical, meteorological, etc., source of information, here

modeled as a channel being inquired: b_geo_ch ?; the auxiliary information, here modeled as a channel being inquired: b_aux_ch ?; and the requirements information, here modeled as a channel being inquired: b_req_ch ?. To each of these three kinds of queries there are therefor corresponding channels: one for the base urban planning geodetic, geotechnical, meteorological, etc., source of information, and one + n channels for both the auxiliary and the requirements information for both the base and the full set of derived urban planning projects. Here n is the number of all foreseen derived urban plannings. The planning behaviours, both the base and the derived, invoke respective urban planning functions, and these produce, such as we have modeled them, quintuplets of information, which are deposited with respective quintuplet repository behaviours: the base quintuplet repository behaviour, and the derived quintuplet repository behaviour — which maintains these quintuplets for all (invoked and thus ongoing) derived urban planning projects. We kindly ask you to review Fig. 2 on the previous page. All you have to 'master' is the fact that there is one base urban planning project, with its repository of base urban planning "quintuplets", and between 0 and n derived urban planning projects, with their shared, derived urban planning "quintuplets", Then there are the channels: the query (input) channels providing auxiliary and requirements information to both the one base urban planning project and the n derived urban planning projects; and the query/repository channels providing "quintuplet" aggregated information to the base urban planning project, as well as "quintuplet" aggregated information to the derived urban planning projects. Finally there are the "global" value representing the index set of derived urban planning indices, and variable which holds the index set of derived urban planning indices of ongoing derived urban planning projects.

4.10. Revised Selection of Derived Urban Plannings

4.10.1. Review

The derived urban planning generator function, der_up, cf. Item 46 on Page 11,

```
value
46  der_up: Unit → Unit
46  der_up() ≡ let dps = sel_dps() in ‖{der_up_beh_i()‖d_tri_beh_i()|i:DP•i ∈ dps} end
```

was invoked with no arguments, der_up(), cf. Item 47 on Page 11 and Item 56 on Page 12

```
47          der_up() ‖ [ respectively ]
56          der_up() ‖
```

4.10.2. A Potential Derived Urban Plan Indices Selector

Selection of potential derived urban planning indices was therefore rather arbitrary. We now let the selection depend on the aggregated resumption state of all (ongoing and) derived urban planning behaviours.

60 Function sel_dups examines either the base resumption or the aggregated resumption state of all (ongoing and) derived urban planning behaviours and yields a set of derived urban planning indices.
61 How it does that is, of course, not defined here.

```
value
60  sel_dups: (bQUI|dQUIm) → DP-set
61  sel_dups(dquim) ≡ ...
```

4.10.3. A Revised Derived Urban Plan Index Set Selector

62 We revise the derived urban plan index se selector fuction give earlier, cf. Item 45 on Page 10. A function, sel_dps, selects zero, one or more DP "fresh" indices, that is, DP tokens that have not been selected before.

```
value
62  sel_dps: DP-set → DP-set
62  sel_dps(dups) ≡ let dps:DP-set•dps⊆dups∩dups \ c dps_var in dps_var := c dps_var ∪ dps; dps end
```

4.10.4. Revision of Derived Urban Plan Invocation

We need to revise the two occurrences of der_up() – in the base urban planning behaviour, and in the (indexed set of) derived urban planning behaviours. Thus

```
47          der_up() ‖ [ respectively ]
56          der_up() ‖
```

is to be replaced by:

```
47          der_up(b_qui_ch?) ‖ ... [ respectively ]
56          der_up(d_qui_ch[ i ]?) ‖ ...
```

4.11. The Urban Planning System

63 Finally we can define an urban planning development as a system of concurrent behaviours:

- the base urban planning behaviour,
- the base "quintuplet" repository and
- the derived and consolidate "quintuplet" repository

```
value
63   up_sys: Unit → Unit
63   up_sys() ≡ base_up_beh() ‖ b_qui_beh(b_qui_init) ‖ d_qui_beh(d_qui_m)
```

Recall that the derived urban planning behaviours as well as the derived triplet behaviours are started by the base as well as the derived urban planning behaviours.

5. Further Work

5.1. Reasoning About Deadlock, Starvation, Livelock and Liveness

The current author is quite unhappy about the way in which he has defined the urban planning, oracle and repository behaviours. Such issues as which invariants are maintained across behaviours are not addressed. In fact, it seems to be good practice, following Dijkstra, Lamport and others, to formulate appropriate such invariants and then "derive" behaviour definitions accordingly. In a rewrite of this research note, if ever, into a proper paper, the current author hopes to follow proper practices.

5.2. Urban Planning Project Management

In this research note we have focused on the urban planning project behaviours, their interactions, and their information "passing". Usually publications about urban planning: research papers, technical papers, survey papers, etcetera, focus on specific "functions". In this research note we do not. Such "functions" are, in this note, embodied in

- b_tri_fit (formula Item 23 on Page 8),
- base_up_fct (formula Item 14 on Page 7 and Item 34 on Page 9),
- base_satisfactory (formula Item 39 on Page 9),
- $\mathsf{der\_satisfactory}_i$ (formula Item 39 on Page 9) and
- $\mathsf{der\_up\_fct}_i$ (formula Item 52 on Page 11).

We focus instead on what we can say about the domain of urban planning: the fact, or the possibility, that an initial, a core, here referred to as a base, urban planning effort (i.e., project, hence behaviour) can "spew off", generate, a number of (derived, i.e., in some sense subsidiary), more specialised, urban planning projects.

MORE TO COME

5.3. Document Handling

5.4. Information Categories

Urban planning consumes information and produces urban planning documents, i.e., plans — referred to in Sects. 3–4 as information ("contained" in triplets and in quintuplets).

type

1 bGEO
2 bAUX
3 bREQ
4 bPLA
5 bANC
6 bTRI = bGEO × bAUX × bREQ
7 bRES = bPLA × bANC
8 bQUI = bTRI × bRES

48 $dAUX_1$, $dAUX_2$, ..., $dAUX_n$
48 $dREQ_1$, $dREQ_2$, ..., $dREQ_n$
48 $dPLA_1$, $dPLA_2$, ..., $dPLA_n$
48 $dANC_1$, $dANC_2$, ..., $dANC_n$
49 $dTRI_i = bQUI \times dAUX_i \times dREQ_i$ [i:DP • i∈dups]
51 $dRES_i = dPLA_i \times dANC_i$ [i:DP • i∈dups]
50 $dQUI_i = dTRI_i \times dRES_i$ [i:DP • i∈dups]
57 $dQUIm = DP \xrightarrow[m]{} dQUI_i$ [i:DP • i∈dups]

6. Conclusion

TO BE WRITTEN

7. Bibliography

7.1. Bibliographical Notes

I have thought about domain engineering for more than 20 years. But serious, focused writing only started to appear since [9, Part IV] — with [5, 2] being exceptions: [11] suggests a number of domain science and engineering research topics; [21] covers the concept of domain facets; [46] explores compositionality and Galois connections. [12, 45] show how to systematically, but, of course, not automatically, "derive" requirements prescriptions from domain descriptions; [24] takes the triptych software development as a basis for outlining principles for believable software management; [16, 31] presents a model for Stanisław Leśniewski's [51] concept of mereology; [22, 25] present an extensive example and is otherwise a precursor for the present paper; [26] presents, based on the `TripTych` view of software development as ideally proceeding from domain description via requirements prescription to software design, concepts such as software demos and simulators; [28] analyses the `TripTych`, especially its domain engineering approach, with respect to [57, 58, Maslow]'s and [59, Peterson's and Seligman's]'s notions of humanity: how can computing relate to notions of humanity; the first part of [33] is a precursor for [39] with the second part of [33] presenting a first formal model of the elicitation process of analysis and description based on the prompts more definitively presented in [39]; and with [34] focus on domain safety criticality. The published paper [39] now constitutes the base introduction to domain science & engineering.

7.2. Domain Modeling Experiments

- Credit Card System[18], [35] 2016. Result of my PhD lectures at Uppsala, May 2016
- Weather Information Systems[19] [41] Result of my PhD lectures at Bergen, November 2016
- Documents[20] [27] 2013.

[18] http://www.imm.dtu.dk/ dibj/2016/uppsala/accs.pdf
[19] http://www.imm.dtu.dk/ dibj/2016/wis/wis-p.pdf
[20] http://www.imm.dtu.dk/~dibj/doc-p.pdf
[21] http://www.imm.dtu.dk/~dibj/comet/comet1.pdf

- Transport Systems[21] [30] 2010.
- The Tokyo Stock Exchange Trading Rules[22] and[23] [42] 2010.
- On Development of Web-based Software[24] 2010.
- What is Logistics ?[25] [17] 2009.
- Pipelines – a Domain Description[26] and[27], [29] 2009.
- Platooning[28],
- A Container Line Industry Domain[29], [10] 2007
- Models of IT Security: Security Rules & Regulations[30] [43] 2006.
- Markets[31] [4]
- **Railway Systems Descriptions: 1996–2003**

  - Dines Bjørner: Formal Software Techniques in Railway Systems[32] [3]
  - Chris George, Dines Bjørner and Søren Prehn: Scheduling and Rescheduling of Trains[33], [47] 1996
  - Dines Bjørner: A Railway Systems Domain[34] An "old" UNU-IIST report, 1997
  - Dines Bjørner: Formal Software Techniques in Railway Systems[35], 2002
  - Albena Strupchanska, Martin Penicka and Dines Bjørner: Railway Staff Rostering[36], 2003 [62]
  - Dines Bjørner: Dynamics of Railway Nets[37], 2003 [6]
  - Martin Penicka, Albena Strupchanska and Dines Bjørner: Train Maintenance Routing[38], 2003 [61]
  - Panagiotis Karras and Dines Bjørner: Train Composition and Decomposition: Domain and Requirements[39], 2003
  - Dines Bjørner: Dynamics of Railway Nets: On an Interface between Automatic Control and Software Engineering[40] [6] 2003

7.3. References

[1] Jean-Raymond Abrial. The B Book: Assigning Programs to Meanings *and* Modeling in Event-B: System and Software Engineering. Cambridge University Press, Cambridge, England, 1996 and 2009.

[2] Dines Bjørner. Michael Jackson's Problem Frames: Domains, Requirements and Design. In Li ShaoYang and Michael Hinchley, editors, *ICFEM'97: International Conference on Formal Engineering Methods*, Los Alamitos, November 12–14 1997. IEEE Computer Society. Final Version.

[3] Dines Bjørner. Formal Software Techniques in Railway Systems. In Eckehard Schnieder, editor, *9th IFAC Symposium on Control in Transportation Systems*, pages 1–12, Technical University, Braunschweig, Germany, 13–15 June 2000. VDI/VDE-Gesellschaft Mess– und Automatisieringstechnik, VDI-Gesellschaft für Fahrzeug– und Verkehrstechnik. Invited talk.

[4] Dines Bjørner. Domain Models of "The Market" — in Preparation for E–Transaction Systems. In *Practical Foundations of Business and System Specifications (Eds.: Haim Kilov and Ken Baclawski)*, The Netherlands, December 2002. Kluwer Academic Press. Final draft version.

[5] Dines Bjørner. Domain Engineering: A "Radical Innovation" for Systems and Software Engineering ? In *Verification: Theory and Practice*, volume 2772 of *Lecture Notes in Computer Science*, Heidelberg, October 7–11 2003. Springer–Verlag. The Zohar Manna International Conference, Taormina, Sicily 29 June – 4 July 2003. .

[6] Dines Bjørner. Dynamics of Railway Nets: On an Interface between Automatic Control and Software Engineering. In *CTS2003: 10th IFAC Symposium on Control in Transportation Systems*, Oxford, UK, August 4-6 2003. Elsevier Science Ltd. Symposium held at Tokyo, Japan. Editors: S. Tsugawa and M. Aoki. Final version.

[7] Dines Bjørner. Towards a Formal Model of CyberRail. In *Building the Information Society, IFIP 18th World Computer Congress, Tpical Sessions, 22–27 August, 2004, Toulouse, France — Ed. Renéne Jacquart*, pages 657–664. Kluwer Academic Publishers, August 2004. Original report also listed some of DB's students as co–authors.

[8] Dines Bjørner. *Software Engineering, Vol. 1: Abstraction and Modelling; Vol. 2: Specification of Systems and*

[22] http://www.imm.dtu.dk/~dibj/todai/tse-1.pdf
[23] http://www.imm.dtu.dk/~dibj/todai/tse-2.pdf
[24] http://www.imm.dtu.dk/~dibj/wfdftp.pdf
[25] http://www.imm.dtu.dk/~dibj/logistics.pdf
[26] http://www.imm.dtu.dk/~dibj/pipeline.pdf
[27] http://www.imm.dtu.dk/~dibj/pipe-p.pdf
[28] http://www.imm.dtu.dk/~dibj/platoon-p.pdf
[29] http://www.imm.dtu.dk/~dibj/container-paper.pdf
[30] http://www.imm.dtu.dk/~dibj/it-security.pdf
[31] http://www2.imm.dtu.dk/~db/themarket.pdf
[32] http://www2.compute.dtu.dk/~dibj/rails.pdf
[33] http://www.imm.dtu.dk/ dibj/amore/docs/scheduling.pdf
[34] http://www.imm.dtu.dk/ dibj/UNU-IIST-railways.pdf
[35] http://www.imm.dtu.dk/ dibj/amore/docs/dines-ifac.pdf
[36] http://www.imm.dtu.dk/ dibj/amore/docs/albena-amore.pdf
[37] http://www.imm.dtu.dk/ dibj/amore/docs/ifac-dynamics.pdf
[38] http://www.imm.dtu.dk/ dibj/amore/docs/martin-amore.pdf
[39] http://www.imm.dtu.dk/ dibj/amore/docs/panos-amore.pdf
[40] http://www2.imm.dtu.dk/~db/ifac-dynamics.pdf

Languages; Vol. 3: Domains, Requirements and Software Design. Texts in Theoretical Computer Science, the EATCS Series. Springer, 2006.

[9] Dines Bjørner. *Software Engineering, Vol. 3: Domains, Requirements and Software Design*. Texts in Theoretical Computer Science, the EATCS Series. Springer, 2006.

[10] Dines Bjørner. A Container Line Industry Domain. Techn. report, Fredsvej 11, DK-2840 Holte, Denmark, June 2007. Extensive Draft.

[11] Dines 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.

[12] Dines 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.

[13] Dines Bjørner. *Software Engineering, Vol. 1: Abstraction and Modelling*. Qinghua University Press, 2008.

[14] Dines Bjørner. *Software Engineering, Vol. 2: Specification of Systems and Languages*. Qinghua University Press, 2008.

[15] Dines Bjørner. *Software Engineering, Vol. 3: Domains, Requirements and Software Design*. Qinghua University Press, 2008.

[16] Dines Bjørner. On Mereologies in Computing Science. In *Festschrift: Reflections on the Work of C.A.R. Hoare*, History of Computing (eds. Cliff B. Jones, A.W. Roscoe and Kenneth R. Wood), pages 47–70, London, UK, 2009. Springer.

[17] Dines Bjørner. What is Logistics ? A Domain Analysis. Techn. report, Incomplete Draft, Fredsvej 11, DK-2840 Holte, Denmark, June 2009.

[18] Dines Bjørner. ***Chinese:*** *Software Engineering, Vol. 1: Abstraction and Modelling*. Qinghua University Press. Translated by Dr Liu Bo Chao et al., 2010.

[19] Dines Bjørner. ***Chinese:*** *Software Engineering, Vol. 2: Specification of Systems and Languages*. Qinghua University Press. Translated by Dr Liu Bo Chao et al., 2010.

[20] Dines Bjørner. ***Chinese:*** *Software Engineering, Vol. 3: Domains, Requirements and Software Design*. Qinghua University Press. Translated by Dr Liu Bo Chao et al., 2010.

[21] Dines Bjørner. Domain Engineering. In Paul Boca and Jonathan Bowen, editors, *Formal Methods: State of the Art and New Directions*, Eds. Paul Boca and Jonathan Bowen, pages 1–42, London, UK, 2010. Springer.

[22] Dines Bjørner. Domain Science & Engineering – *From Computer Science to The Sciences of Informatics, Part I of II: The Engineering Part*. *Kibernetika i sistemny analiz*, (4):100–116, May 2010.

[23] Dines Bjørner. The Tokyo Stock Exchange Trading Rules. R&D Experiment, Fredsvej 11, DK-2840 Holte, Denmark, January, February 2010.

[24] Dines Bjørner. Believable Software Management. *Encyclopedia of Software Engineering*, 1(1):1–32, 2011.

[25] Dines 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.

[26] Dines Bjørner. Domains: Their Simulation, Monitoring and Control – A Divertimento of Ideas and Suggestions. In *Rainbow of Computer Science, Festschrift for Hermann Maurer on the Occasion of His 70th Anniversary.*, Festschrift (eds. C. Calude, G. Rozenberg and A. Saloma), pages 167–183. Springer, Heidelberg, Germany, January 2011.

[27] Dines Bjørner. Documents – a Domain Description[41]. Experimental Research Report 2013-3, DTU Compute and Fredsvej 11, DK-2840 Holte, Denmark, Spring 2013.

[28] Dines 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).

[29] Dines Bjørner. Pipelines – a Domain Description[42]. Experimental Research Report 2013-2, DTU Compute and Fredsvej 11, DK-2840 Holte, Denmark, Spring 2013.

[30] Dines Bjørner. Road Transportation – a Domain Description[43]. Experimental Research Report 2013-4, DTU Compute and Fredsvej 11, DK-2840 Holte, Denmark, Spring 2013.

[31] Dines Bjørner. *A Rôle for Mereology in Domain Science and Engineering*. Synthese Library (eds. Claudio Calosi and Pierluigi Graziani). Springer, Amsterdam, The Netherlands, October 2014.

[32] Dines Bjørner. Domain Analysis: Endurants – An Analysis & Description Process Model. In Shusaku Iida, José Meseguer, and Kazuhiro Ogata, editors, *Specification, Algebra, and Software: A Festschrift Symposium in Honor of Kokichi Futatsugi*. Springer, May 2014. (paper[44], slides[45]).

[33] Dines Bjørner. Domain Analysis: Endurants – An Analysis & Description Process Model. In Shusaku Iida, José Meseguer, and Kazuhiro Ogata, editors, *Specification, Algebra, and Software: A Festschrift Symposium in Honor of Kokichi Futatsugi*. Springer, May 2014.

[34] Dines Bjørner. Domain Engineering – A Basis for Safety Critical Software. Invited Keynote, ASSC2014: Australian System Safety Conference, Melbourne, 26–28 May, December 2014.

[41] http://www.imm.dtu.dk/~dibj/doc-p.pdf
[42] http://www.imm.dtu.dk/~dibj/pipe-p.pdf
[43] http://www.imm.dtu.dk/~dibj/road-p.pdf
[44] http://www.imm.dtu.dk/~dibj/jaist-da.pdf
[45] http://www.imm.dtu.dk/~dibj/jaist-s.pdf

[35] Dines Bjørner. A Credit Card System: Uppsala Draft. Technical Report: Experimental Research, Fredsvej 11, DK–2840 Holte, Denmark, November 2016. http://www.imm.dtu.dk/~dibj/2016/credit/accs.pdf.
[36] Dines Bjørner. Domain Analysis and Description – Formal Models of Processes and Prompts. *Submitted for consideration to Formal Aspects of Computing*, 2016. http://www.imm.dtu.dk/~dibj/2016/process/process-p.pdf.
[37] Dines Bjørner. Domain Facets: Analysis & Description. *Submitted for consideration to Formal Aspects of Computing*, 2016. http://www.imm.dtu.dk/~dibj/2016/facets/faoc-facets.pdf.
[38] Dines Bjørner. Domains: Their Simulation, Monitoring and Control – A Divertimento of Ideas and Suggestions. Experimental Research, Fredsvej 11, DK–2840 Holte, Denmark, 2016. http://www.imm.dtu.dk/~dibj/2016/demos/-faoc-demo.pdf.
[39] Dines Bjørner. Manifest Domains: Analysis & Description. *Formal Aspects of Computing*, ...(...):1–51, 2016. DOI 10.1007/s00165-016-0385-z http://link.springer.com/article/10.1007/s00165-016-0385-z.
[40] Dines Bjørner. To Every Manifest Domain a `CSP` Expression — A Rôle for Mereology in Computer Science. Submitted for consideration to Journal of Logical and Algebraic Methods in Programming, Fredsvej 11, DK–2840 Holte, Denmark, December 2016. http://www.imm.dtu.dk/~dibj/2016/mereo/mereo.pdf.
[41] Dines Bjørner. Weather Information Systems: Towards a Domain Description. Technical Report: Experimental Research, Fredsvej 11, DK–2840 Holte, Denmark, November 2016. http://www.imm.dtu.dk/~dibj/2016/wis/wis-p.pdf.
[42] Dines Bjørner. The Tokyo Stock Exchange Trading Rules. R&D Experiment, Fredsvej 11, DK-2840 Holte, Denmark, January and February, 2010. Version 1, 78 pages: many auxiliary appendices, Version 2, 23 pages: omits many appendices and corrects some errors..
[43] Dines Bjørner. *[44] Chap. 9: Towards a Model of IT Security — – The ISO Information Security Code of Practice – An Incomplete Rough Sketch Analysis*, pages 223–282. JAIST Press, March 2009.
[44] Dines Bjørner. *Domain Engineering: Technology Management, Research and Engineering*. A JAIST Press Research Monograph # 4, 536 pages, March 2009.
[45] Dines Bjørner. The Rôle of Domain Engineering in Software Development. Why Current Requirements Engineering Seems Flawed! In *Perspectives of Systems Informatics*, volume 5947 of *Lecture Notes in Computer Science*, pages 2–34, Heidelberg, Wednesday, January 27, 2010. Springer.
[46] Dines Bjørner and Asger Eir. Compositionality: Ontology and Mereology of Domains. Some Clarifying Observations in the Context of Software Engineering in July 2008, eds. Martin Steffen, Dennis Dams and Ulrich Hannemann. In *Festschrift for Prof. Willem Paul de Roever Concurrency, Compositionality, and Correctness*, volume 5930 of *Lecture Notes in Computer Science*, pages 22–59, Heidelberg, July 2010. Springer.
[47] Dines Bjørner, Chris W. George, and Søren Prehn. *Scheduling and Rescheduling of Trains*, chapter 8, pages 157–184. *Industrial Strength Formal Methods in Practice*, Eds.: Michael G. Hinchey and Jonathan P. Bowen. FACIT, Springer–Verlag, London, England, 1999.
[48] Dines Bjørner, Chris W. George, and Søren Prehn. Computing Systems for Railways — A Rôle for Domain Engineering. Relations to Requirements Engineering and Software for Control Applications. In *Integrated Design and Process Technology. Editors: Bernd Kraemer and John C. Petterson*, P.O.Box 1299, Grand View, Texas 76050-1299, USA, 24–28 June 2002. Society for Design and Process Science. Extended version.
[49] Dines Bjørner and Cliff B. Jones, editors. *The Vienna Development Method: The Meta-Language*, volume 61 of *LNCS*. Springer, 1978.
[50] Dines Bjørner and Cliff B. Jones, editors. *Formal Specification and Software Development*. Prentice-Hall, 1982.
[51] R. Casati and A. Varzi. *Parts and Places: the structures of spatial representation*. MIT Press, 1999.
[52] John Fitzgerald and Peter Gorm Larsen. *Modelling Systems – Practical Tools and Techniques in Software Development*. Cambridge University Press, The Edinburgh Building, Cambridge CB2 2RU, UK, 1998. ISBN 0-521-62348-0.
[53] Chris W. George, Peter Haff, Klaus Havelund, Anne Elisabeth Haxthausen, Robert Milne, Claus Bendix Nielsen, Søren Prehn, and Kim Ritter Wagner. *The RAISE Specification Language*. The BCS Practitioner Series. Prentice-Hall, Hemel Hampstead, England, 1992.
[54] Chris W. George, Anne Elisabeth Haxthausen, Steven Hughes, Robert Milne, Søren Prehn, and Jan Storbank Pedersen. *The RAISE Development Method*. The BCS Practitioner Series. Prentice-Hall, Hemel Hampstead, England, 1995.
[55] Charles Anthony Richard Hoare. *Communicating Sequential Processes*. C.A.R. Hoare Series in Computer Science. Prentice-Hall International, 1985. Published electronically: http://www.usingcsp.com/cspbook.pdf (2004).
[56] Daniel Jackson. *Software Abstractions: Logic, Language, and Analysis*. The MIT Press, Cambridge, Mass., USA, April 2006. ISBN 0-262-10114-9.
[57] Abraham Maslow. A Theory of Human Motivation. *Psychological Review*, 50(4):370–96, 1943. http://psych-classics.yorku.ca/Maslow/motivation.htm.
[58] Abraham Maslow. *Motivation and Personality*. Harper and Row Publishers, 3rd ed., 1954.
[59] Christopher Peterson and Martin E.P. Seligman. *Character strengths and virtues: A handbook and classification*. Oxford University Press, 2004.
[60] Martin Pěnička and Dines Bjørner. From Railway Resource Planning to Train Operation — a Brief Survey of Complementary Formalisations. In *Building the Information Society, IFIP 18th World Computer Congress, Topical Sessions, 22–27 August, 2004, Toulouse, France — Ed. Renéne Jacquart*, pages 629–636. Kluwer Academic Publishers, August 2004.
[61] Martin Pěnička, Albena Kirilova Strupchanska, and Dines Bjørner. Train Maintenance Routing. In *FORMS'2003: Symposium on Formal Methods for Railway Operation and Control Systems*. L'Harmattan Hongrie, 15–16 May 2003. Conf. held at Techn.Univ. of Budapest, Hungary. Editors: G. Tarnai and E. Schnieder, Germany. Final version.

[62] Albena Kirilova Strupchanska, Martin Pěnička, and Dines Bjørner. Railway Staff Rostering. In *FORMS2003: Symposium on Formal Methods for Railway Operation and Control Systems*. L'Harmattan Hongrie, 15–16 May 2003. Conf. held at Techn.Univ. of Budapest, Hungary. Editors: G. Tarnai and E. Schnieder, Germany. Final version.

[63] J. C. P. Woodcock and J. Davies. *Using Z: Specification, Proof and Refinement*. Prentice Hall International Series in Computer Science, 1996.

A. A Document System

A.1. The System

64 From a document system

65 one can observe an aggregate of documents

66 and an aggregate of document handlers.

67 From an aggregate of documents one can observe a set of documents.

68 From an aggregate of document handlers one can observe a set of document handlers.

```
type
64   DS
65   ADS
66   AHS
value
65   obs_ADS: DS → ADS
66   obs_AHS: DS → AHS
type
67   DOC, DOCS = DOC-set
value
67   obs_DOCS: ADS → DOCS
type
68   HAN, HANS = HAN-set
value
68   obs_HANS: AHS → HANS
```

A.2. Time

69 We postulate a notion of time, one that covers both a calendar date (from before Christ up till now and beyond). But we do not specify any concrete type (i.e., format such as: YY:MM:DD, HH:MM:SS).

70 And we postulate a notion of (signed) time interval — between two times (say: ±YY:MM:DD:HH:MM:SS).

71 Then we postulate some operations on time: Adding a time interval to a time obtaining a time; subtracting one time from another time obtaining a time interval, multiplying a time interval with a natural number; etc.

72 And we postulate some relatioins between times and between time intervals.

```
type
69. TIME
70. TIME_INTERVAL
value
71. add: TIME_INTERVAL × TIME → TIME
71. sub: TIME × TIME → TIME_INTERVAL
71. mpy: TIM_INTERVALE × Nat → TIME_INTERVAL
72. <,≤,=,≠,≥,>: ((TIME×TIME)|(TIME_INTERVAL×TIME_INTERVAL)) → Bool
```

A.3. Unique Identification

73 From a document one can observe its/a unique [document] identifier.

74 From a handler one can observe its/a unique [handler] identifier.

```
type
73. DI
74. HI
value
73. uid_DI: DOC → DI
74. uid_HI: HAN → HI
```

A.4. Documents

A.4.1. Attributes

From documents one can observe:

75 the "most current" textual[46] "contents", txt:TXT:

```
type
75 TXT
value
75 attr_TXT: DOC → TXT.
```

76 We can refer to a position in any text.

77 A pair of proper and ascending text positions delineate a text.

```
type
76 Pos
value
77 delineate: TXT → (Pos×Pos) ~→ TXT
77 delineate(txt)(p1,p2) as txt′
77     pre: proper_txt_pos(p1)(txt) ∧ proper_txt_pos(p2)(txt) ∧ ascending_txt_pos(p1,p2)(txt)

77 proper_txt_pos: Pos → TXT → Bool
77 proper_txt_pos(p)(txt) as ...
77 ascending_txt_pos: (Pos×Pos) → TXT → Bool
77 ascending_txt_pos(p1,p2)(txt) as ... pre: proper_txt_pos(p1)(txt) ∧ proper_txt_pos(p2)(txt)
```

76 From document we further observe pairs of

77 editing functions, edf:EDIT, which was "most recently" applied to the (predecessor) of a document text, txt:TXT, if any, and

78 undo functions, undo:UNDO, which "bring back" the document text, txt:TXT, which was edited.

79 Hence we can postulate a predicate, was_edited, which, when applied to a document that has been edited, yields **true**, otherwise **false**.

80 An axiom expresses that the composition of the an undo function with its "corresponding" edit function designates the identity function.

```
type
77 EDIT = TXT → TXT
78 UNDO = TXT → TXT
value
76 attr_undo_edit: DOC → (UNDO×EDIT)
79 edited: DOC → Bool
axiom
80 ∀ doc:DOC•edited(doc)⇒let txt=attr_TXT(doc),(u,e)=attr_undo_edit(doc) in e(u(txt))=txt end
```

81 From a document we can observe the time at which the most recent operation was peformed on that document:

[46] By text we mean text as in a book of fiction: novel or poetry, as in a mathematics text monograph or lecture notes, or we mean graphics, as in a geographic map, or in a visualisation of scientific data, or we mean tables of data as in a statistics yearbook, or we mean any form of combinations of these forms of text.

value 81. attr_TIME: DOC → TIME

82 And we can observe the identity of the handler who most recently performed an operation on a document:

value 82. attr_HI: DOC → HI

83 ***

83.

84 ***

84.

A.4.2. A Meta-linguistic "Trick"

85 Let attr_fcts designate the set of all the attribute observers defined on documents.
86 Let attr_fct denote a specific one of these attribute observers.
87 Then by

- attr_fcts\{attr_fct}

we shall mean the set attr_fcts without (say "minus") the element attr_fct.

88 Now, if we were to express that two documents, doc' and doc" are identical except for their unique identifiers and except for a given attribite observer, say attr_XYZ, then we would write

- doc'/{attr_XYZ} = doc"/{attr_XYZ}, i.e.: ∀ f ∈ attr_fcts\{attr_XYZ} • f(doc′)=f(doc″)

A.5. Handlers

A.5.1. Attributes

A.5.2. Operations

A.6. Behaviours

A.6.1. Generic Behaviours

We can, according to [39, Manifest Domains: Analysis & Description] "equate" parts (i.e., aggregates of documents and handlers, and document and handlers, as parts, respectively composite and atomic) with behaviours. So we shall introduce document and, later, handler behaviours. Document (and handler) behaviours are uniquely identified by document (respectively handler) identifiers.

```
value
89   part_i: Static_Attrs → Programmable_Attrs →
90                 in {i_ch[j]|j:J•i∈Jx(i)}, out {o_ch[k]|k:K•j∈Kx(i)} Unit
91   part_i(sa_i1,...,sa_im)(pa_i1,...,pa_in) ≡
91a        let pa'_i1 = P_i1(sa_i1,...,sa_im)(pa_i1,...,pa_in),
91b            ...,
91c            pa'_in = P_in(sa_i1,...,sa_im)(pa_i1,...,pa_in) in
92         Q_i(i_ch[f(i)],o_ch[g(i)])(sa_i1,...,sa_im)(pa'_i1,...,pa'_in) ;
93         part_i(sa_i1,...,sa_im)(pa'_i1,...,pa'_in) end
```

89 $part_i$ is the name of a generic behaviour. It has a number of arguments:

- First there are those that reflect the static attributes, sa_{i_1},...,sa_{i_m}:Static_Attrs, of the part that the behaviour models. The static attributes are those properties that remain unchanged "through the life" of the part. Examples of static attributes of documents could be: *document type*: whether an urban planning geographic, or an auxiliary, or a requirements document.
- Then there are the programmable attributes, pa_{i_1},...,pa_{i_n}:Programmable_Attrs. The programmable attributes are those whose value change in response to concerted actions. Examples of programmable attributes of documents are: *text:TXT*, *(undo,edit):(UNDO×EDIT)*, etcetera.

90 part_i behaviours *synchronize and communicate* with other behaviours over channels. Inputs from other behaviours are "declared" by {i_ch[j]|j:J•i∈Jx(i)}; outputs to other behaviours are "declared" by **out** {o_ch[k]|k:K•j∈Kx(i)}. The **Unit** clause indicates that part_i behaviours "go on forever and leave no "state" changes" !

91 The template pattern for behaviour invocation and continuation is $\mathsf{part}_i(\mathsf{sa}_{i_1},...,\mathsf{sa}_{i_m})(\mathsf{pa}_{i_1},...,\mathsf{pa}_{i_n})$ – and conforms with the behaviour signature of formula lines 89–90. The behaviour, typically, but not necessarily, "updates" all its programmable attributes,

a pa'_{i_1}, b ..., and c pa'_{i_m},

by evaluating corresponding clauses $\mathcal{P}_{i_x}(\mathsf{sa}_{i_1},...,\mathsf{sa}_{i_m})(\mathsf{pa}_{i_1},...,\mathsf{pa}_{i_n})$ where respective $\mathcal{P}_{i_x}$ need not "contain" all their listed attributes.

92 The $\mathcal{Q}_i(\mathsf{i\_ch}[\,f(\mathsf{i})\,],\mathsf{o\_ch}[\,g(\mathsf{i})\,])(\mathsf{sa}_{i_1},...,\mathsf{sa}_{i_m})(\mathsf{pa}'_{i_1},...,\mathsf{pa}'_{i_n})$ clause really is not "proper syntax". The clause is intended to express that behaviour part_i may or may not exchange information with other beaviours. Here that is shown to "occur after" evaluation of new programmable attributes. It could as well have "occurred before" in which case we would have to express "other forms" of updated programmable attribute values.

93 Finally behaviour part_i resumes being behaviour part_i, only now with updated programmable attributes (the static attributes having not changed).

A.6.2. Document Behaviours

94 ***

type 94
94 94
94 94
94

95 ***

type 95
95 95
95 95
95

A.6.3. Handler Behaviours

A.7. Mereology

96 ***

96

97 ***

97

98 ***

98