Documents: A Domain Analysis[1]

An Experiment in Domain Engineering

Version 2

Dines Bjørner

Computer Science and Engineering
Informatics and Mathematical Modelling
Technical University of Denmark
DK-28000 Kgs.Lyngby
Denmark

Graduate School of Information Science
Japan Adv. Inst. of Science & Technology
1-1, Asahidai, Tatsunokuchi
Nomi, Ishikawa 923-1292
Japan

bjorner@gmail.com

July 27, 2006. Compiled July 31, 2006

Abstract

We aim at exemplifying applications of Domain Engineering. That is, to describe, informally and formally, typically, a man-made universe of discourse. As it is. Without any reference to requirements to any computing system, let alone software to support activities in that domain. The domain is that of documents. We wish to understand what documents are: *original* and *master* documents, *versions*, *copies*, etc., document *creation*, *editing*, *reading*, *copying*, etc. The idea of domain modelling and the style used in this presentation is basically the same — more-or-less independent of the domain. The current Version 2 of this report is skimpy on the issue of system space of documents (and agenets), of the dynamic (time-wise) movement of documents, and hence on all the very many constraints that govern documents and their locations.

Contents

1 Introduction **2**
- 1.1 Aims and Objectives . . . 2
  - 1.1.1 Aims . . . 2
  - 1.1.2 Objectives . . . 3
- 1.2 Domain Engineering . . . 3
- 1.3 Related Work . . . 3
- 1.4 A Caveat . . . 3
- 1.5 Structure of Paper . . . 3

2 Documents: A Domain Analysis **4**
- 2.1 Basics . . . 4
  - 2.1.1 Analysis: Basic Entities . . . 4
  - 2.1.2 Narrative: Basic Entities . . . 4
  - 2.1.3 Formalisation: Basic Entities . . . 5
  - 2.1.4 Narrative: Basic Functions . . . 5
  - 2.1.5 Formalisation: Basic Functions . . . 6
  - 2.1.6 Formalisation: The Editing Functions . . . 6
  - 2.1.7 Analysis: Document Identifiers . . . 6
  - 2.1.8 Formalisation: Document Identifiers . . . 7

[1] Paper to be presented at IBM TRL, August 9

2.1.9 Analysis: Document Kinds . . . 7
2.1.10 Analysis: Document History . . . 8
2.1.11 Narrative: Document Kinds . . . 8
2.1.12 Formalisation: Document Kinds . . . 8
2.1.13 Narrative: Document History . . . 9
2.1.14 Formalisation: Document History . . . 9
2.2 Locations, Time and Agents . . . 10
2.2.1 Locations . . . 10
Analysis: Points: . . . 10
Narration: Locations: . . . 10
Formalisation: Locations: . . . 11
2.2.2 Time and Time Intervals . . . 12
Analysis: Time: . . . 12
Formalisation: Time: . . . 12
Comments on the Axiomatisation: Time: . . . 12
Analysis: Time Intervals: . . . 13
Formalisation: Time and Time Intervals: . . . 13
2.2.3 Time and Space . . . 13
Analysis: Time and Space: . . . 13
Annotations: Time and Space: . . . 14
Discussion of the *Blizard* Model of Space/Time: . . . 14
2.2.4 Agents . . . 14
Analysis: Agents: . . . 14
Narrative: Agents: . . . 15
Formalisation: Agents: . . . 15
2.3 Spaces of Documents and Agents . . . 15
2.3.1 Narrative: Spaces of Documents and Agents . . . 15
2.3.2 Formalisation: Spaces of Documents and Agents . . . 16
2.3.3 Formalisation: Types of Document Operations . . . 16

3 Conclusion **16**
3.1 What Have We Shown? . . . 16
3.2 Shortcomings . . . 17
3.3 What Needs be Done? . . . 17

4 Acknowledgements **17**

References **17**

1 Introduction

1.1 Aims and Objectives

The abstract said it all. But we can reiterate some points. In software development we proceed from modelling the application or business domain, via constructing requirements from the domain model, to designing the computing system, including software. We naturally wish to understand the business area well before we embark on requirements.

1.1.1 Aims

So the aims are to show you aspects of domain modelling. What it includes doing. How one might go about doing it (including analysis).

1.1.2 Objectives

And the objectives are to convince you to do likewise. To do software development professionally.

1.2 Domain Engineering

Domain engineering is the engineering, the managed construction of domain descriptions. As such domain engineering includes [1.] identifying, contacting, and throughout consulting all possibly relevant domain stakeholders; [2.] domain knowledge acquisition; [3.] analysing acquired domain knowledge; [4.] creating the domain model[2]; [5.–6.] verifying and then validating the model[3]; [7.] and possibly turning the domain model into a domain theory[4]. In this paper we shall only illustrate item [4.]. We refer to Part III (Chaps. 8–16) of volume 3 of the three volume book on Software engineering [1, 2, 3] for "the 'full' story" on Domain Engineering.

1.3 Related Work

For the kind of document domain analysis, as we delineate it, there is, to our knowledge, no related work. Well, there is probably some AI-related knowledge engineering (KIF[5]) work and there is definitely some Web-oriented (for example, DAML[6] Web-ontology) on 'documents'. But it appears that all of this work is about the textual content of documents — well-nigh the only thing we are not interested in modelling in this paper. That is, our model is not about the textual (statistical, linguistical or other) properties of documents, but of how documents are handled irrespective of their content.

1.4 A Caveat

This is, in principle, a first complete draft of a report, perhaps eventually a publishable paper, on the issue of domain analysis of one view of the domain of documents. It is based on various fragments, some appearing in the authors three volumes on software engineering [1, 2, 3], others appearing in draft versions of a report worked on during the early spring of 2006 at JAIST. The present draft report is a complete rewrite of those earlier fragments — with no rereading of these having beein done immediately before or during the writing of the current report. It is released to colleagues and some IBM TRL staff before or on 9 August 2006. It is much expected that the author will have time to polish this report into a possibly publishable paper during the early fall of 2006.

1.5 Structure of Paper

There are basically three sub-parts to the main part of this paper. In the first we analyse basic aspects of documents: whick kinds of dicuments there are, the operations on documents,

[2] A domain description is a syntactic construction. A domain model is a selected meaning (semantics) of that description.

[3] Verfication shall ensure that the model is correct (i.e., right). Validation shall ensure that we have the right model.

[4] A domain theory is the domain model + a(ny) number of theorems about the model: its properties, so to speak.

[5] http://logic.stanford.edu/kif/kif.html

[6] The DAML (The DARPA Agent Markup Language) homepage is http://www.daml.org/

© Dines Bjørner 2006, Fredsvej 11, DK–2840 Holte, Denmark

document history, etc. In the second part we analyse notions of locations, time and agents. And in the third part we enlarge the scope to that of spaces of documents and agents. A concluding sub-part reviews what has been done, the possible shortcomings of the present work, and hopes for future improvement and extensions.

2 Documents: A Domain Analysis

2.1 Basics

2.1.1 Analysis: Basic Entities

Think of *documents* as stapled or bound collections a sheets of paper. From blank sheets you can *create* a basically blank document. You and others can write something on the sheets (i.e., *edit* them). You and others can *read* these sheets, i.e., the document.

You can *copy* a document. Lets call the document being copied the *master*. Now two documents exists, one per copying action. To make n (say 100) copies is to copy the either the master a n times, or to mix copying copies (which thereby become "new masters"), or the "original" master. The point is: You know what you are doing: which way you did the copying. Our domain model of documents must capture that.

You can *edit* a document. After editing the document upon which the editing was done no longer "exists". An edited *version* has been made of that "prior" document. If, say, that basis document (the prior one) was typed, and your editing was made by pencil, then from the edited version you can, in a sense, "capture" ("just now edited") un-edited (the base, prior) document. In a sense you can "*undo*" the editing. Our domain model of documents must capture that.

You can *read* a document, say at location ℓ and time τ. Before you read it (at location ℓ and time τ) the document was known to have not been read by you at location ℓ and time τ. Afterwards, that is, after time τ, it is known to have been read by you at location ℓ and time τ.

You can *move*, say distribute, a document. Before it was moved it was a some location ℓ. After the move it is at some other location ℓ'. No two documents can at any time at a same location. A document cannot be at two distinct locations at any (one) time. Our domain model of documents must capture that.

Our domain model of documents must capture a lot more!

2.1.2 Narrative: Basic Entities

1. We postulate sorts, i.e., a types of documents, locations, time, agents and tex.
2. With a document we can associate, i.e., we can *observe*
   (a) the current *location* of the document,
   (b) the *time* it was most recently operated upon,
   (c) some identity of the *agent* (f.ex., person) who performed the most recent opration on the document,
   (d) which *kind of* agent *operation* was last applied to the document,

(e) and we can observe some *text*, "the document contents".[7]

3. The 'kinds of operations' are *tokens*. (See items 5 below.)

2.1.3 Formalisation: Basic Entities

type
1. D, L, T, A

value
2a. obs_L: D → L
2b. obs_T: D → T
2c. obs_A: D → A
2c. obs_OpKind: D → OpKind
2e. obs_Txt: D → Txt

type
3. OpKind == cr|ed|rd|cp|sh|mv|...

2.1.4 Narrative: Basic Functions

4. From a document one can *observe* the triple of *location*, *time* and performing *agent* of 'most recent operation'.

5. Documents can be
   (a) created,
   (b) edited,
   (c) read,
   (d) copied,
   (e) shredded,
   (f) moved,
   (g) and a few other things (...).[8]

6. Editing a document d results in a pair, $(d', (f_e, b_e))$:
   (a) f_e is a forward editing function.
   (b) b_e is the inverse, "backward" editing function,
   (c) and $d' = f_e(d)$ and $d = b_e(d')$,
   (d) that is, $f_e(b_e(d')) = d'$ and $b_e(f_e(d)) = d$.
   (e) The pair (f_e, b_e) is a result of the editing. It can only be known a posteriori.

[7] Other than being able to observe the text we shall not deal any further with format, statistical or linguistic properties of texts.

[8] We shall postpone till later considerations of moving (distributing), sharing , calculating over (incl. searching for) and tracing documents.

© Dines Bjørner 2006, Fredsvej 11, DK–2840 Holte, Denmark

2.1.5 Formalisation: Basic Functions

type
L, T, A, E
LTA = L × T × A
value
5a. crea: LTA → D
5b. edit: LTA → D → D × E
5c. read: LTA → D → D
5d. copy: LTA → D → D × D
5e. shrd: LTA → D → **Unit**

4. obs_LTA: D → LTA

2.1.6 Formalisation: The Editing Functions

type
E = FE × BE
6a. FE = D → D
6b. BE = D → D
value
7. edit: LTA → D → D × E
axiom
∀ d:D,(fe,be):E • d = be(fe(d)), i.e.,: fe°be=λd.d ∧ be°fe=λd.d
i.e.:
∀ lta:LTA,d:D • **let** (d′,(fe,be)) = edit(lta)(d) **in**
6c. d′ = fe(d) ∧ d = be(d′) **end**
i.e.:
6d. d′ = fe(be(d′)) ∧ d = be(fe(d))

2.1.7 Analysis: Document Identifiers

Let's do a bit more analysis. Documents which have just been created are referred to as *originals*. With any original document we can associate a *unique document identifier*. You may think of the unique document identifier being a simple isomorphic function of three quantities: the identity of the agent who (or which) created the document, a unique representation of the location at which the document was created, and a unique representation of the time at which the document was created. No agent can be at two locations at any one time and perform more than one operation at a time.and operations requires agents, hence all original document identifiers are unique. We shall soon see that no two otherwise distinct documents which may exist at any time can have the same unique identifier.

Thus there are orginal ("blank") documents distinguishable, albeit, by unique document identifiers. *Editing* a document need not change its identification. Its is, for example, still "one and the same" sheet of papers, albeit with, perhaps, a different textual contents[9]. *Reading*

[9] And, if not a collection of sheets of paper, then it might be an electronic document for which we would want the same property as for the humanly manifest document. Yes? Yes!

and *moving* a document need not change its identification either. *Copying* a document results in one more document: the *master* whose identification could (hence in our model will) be the same, since it is "physically" the same, as the document being copied, and the *copy*, the, or a, "new" document — whose identification must be made unique, hence different from the identification of the master document. We suggest that the identification of the copy document be an *isomorphic function* of the identification of the master document and the triplet identification of the location, the location and the agent of copying. Since no agent can be performing more than one operation at a time all documents will have unique identifiers.

2.1.8 Formalisation: Document Identifiers

type
  L, T, A, D, E, UDI
  LTA = L × T × A
value
  crea: LTA → **Unit** → D
  edit: LTA → D → D × E
  read: LTA → D → D
  copy: LTA → D → D × D
  ...
  obs_UDI: D → UDI
  udi: (LTA|(LTA×LTA)) → UDI
axiom
  ∀ ℓta:LTA,d:D,e:E •
    obs_UDI(crea(ℓta))=udi(ℓta) ∧
    obs_UDI(d) = **let** (d′,efs)=edit(ℓta)(d) **in** obs_UDI(d′) **end** ∧
    obs_UDI(d) = obs_UDI(read(ℓta)(d)) ∧
    let (md,cd) = copy(ℓta)(d) **in**
      obs_UDI(d) = obs_UDI(md) ∧
      obs_UDI(cd) = udi(obs_UDI(md),ℓta) ∧ ... **end**

The last line of the above axiom does not guarantee uniqeness across all documents. That issue will be addressed below.

2.1.9 Analysis: Document Kinds

So there are originals, that is, documents that have been created, but upon which no actions have so far been performed. Then there are documents which have been edited, read, are copies of masters, and hence there are master documents; and there are documents which have been moved. For the time being we shall be content with just these kinds of documents (and hence 'kind of operation' tokens).

An edited document may be textually distinct from the document from which it was edited. A document which has been read (at some location and time and by some agent) is, textually, the same as when that document had not been read (at that location and time and by that agent). A document which has been copied becomes a master document, otherwise textually indistinguishable from the document from which the copy was made. The copy of a

© Dines Bjørner 2006, Fredsvej 11, DK–2840 Holte, Denmark

master document is a copied document and is distinguishable, not textually, but "kind-wise", from the master document.

2.1.10 Analysis: Document History

So "most recently" edited documents are distinct textually distinct from the documents from which they were edited.

The reason, that is, the pragmatics behind why we introduce the notion of 'document kind' is so that we can reason about documents: "That document has been read, by such and such, several times." "That document is an edited version of a document which we was read by agent α, from a document which was copied by ..., etc." We wish to also be able to also reason about locations and times of most recent actions, and about the document as it was before such an action: "the document from which it resulted".

Although in today's practical life one may not record, on documents, who first created them (where and when), who edited them (where and when), who read them (where and when), who copied them (where and when), or from which masters they were copied (where and when), etc., we may still, in this everyday world, be able to reason about such things. Therefore we must model it.[10]

2.1.11 Narrative: Document Kinds

7. Thus documents are "most recently" either
   (a) created, i.e., they are [blank] "originals",
   (b) edited,
   (c) read,
   (d) the basis for copying, i.e., they are "masters", or
   (e) the result of copying, i.e., they are copies,
   (f) etcetera.

2.1.12 Formalisation: Document Kinds

type

7 mD,D = oD | eD | rD | cD

axiom

∀ odoc:oD,edoc:eD,rdoc:rD,cdoc:cD •
7a obs_OpKind(odoc) = cr
7b obs_OpKind(edoc) = ed
7c obs_OpKind(rdoc) = rd
7d obs_OpKind(cdoc) = cp
∀ lta:LTA,d:D •
7a obs_OpKind(crea(lta)) = cr
7b obs_OpKind(edit(lta)(d)) = (ed,efs)

[10] The issue of the certainty (i.e., uncenrtainty) with which we may say reason will be discussed later.

7c obs_OpKind(read(lta)(d)) = rd
7d **let** (d′,d″)=copy(lta)(d) **in** obs_OpKind(d′)=cp∧obs_OpKind(d′)∈{cr,ed,rd,cp,...} **end**
...

2.1.13 Narrative: Document History

8. With a document we can associate its history of operations performed on that document. More specifically we can think of a document history to be a list of triples: the operation performed, the document, and a location, time and agent triple.
9. An original document has no pre-history. It has the history that it was created at such and such a location and time by such and such an agent.
10. An edited document has a history which consists of the its pre-history, namely the history of the document from which it was edited, and then the most recent historical fact, namely that it was edited at such and such a location and time by such and such an agent, and that the editing can be captured, i.e., the text of the document from which it was edited, as well as the text resulting from the editing.
11. A document which has "just" been read has the pre-history of the document before it was "just" read as well as the most recent historical fact (location, time and agent of reading).
12. The two documents which have partaken in a copying action: the master and the copy documents have different "most recent" histories, but the same pre-history:
    (a) The master copy has, as "most recent" history that it served as a master for a copying action (by whom, when and where).
    (b) The copy has, as "most recent" history that it was the result of a copying action (by whom, when and where).
    (c) Since the unique identifier of the copy "embeds" that of the master one can identify the master.
    (d) Perhaps one should "insert" as part of the "most recent" master history the unique identifier of the copy: that identifier can anyway be constructed!
13. Etcetera.

2.1.14 Formalisation: Document History

type
8 H == D*
value
8 obs_H: D → H
wf_H(dl) ≡ **len** dl≥1∧dl(1)=create(lta)∧create(lta)∉ **elems tl** dl
axiom
∀ lta:LTA,d:D •
9 obs_H(create(lta)) = ⟨create(lta)⟩ ∧
10 obs_H(edit(lta)(d)) = obs_H(d)^⟨edit(lta)(d)⟩ ∧

© Dines Bjørner 2006, Fredsvej 11, DK–2840 Holte, Denmark

```
11    obs_H(read(lta)(d)) = obs_H(d)^⟨read(lta)(d)⟩ ∧
12    let (md,cd) = copy(lta)(d) in
12a       obs_H(md)=obs_H(d)^⟨md⟩∧
12b       obs_H(cd)=obs_H(d)^⟨cd⟩ ∧
12c       ... end
```

2.2 Locations, Time and Agents

We have mentioned that we do not study textual issues. But we are concerned about locations, time and agents. There are two notions of location: the location at which an operation on a document took, or takes place, and the location of the document "right now". The latter is, of course (?), the same as the location to where the most recent move action brought that document! There are two similar notions of time the time at which an operation on a document took, or takes place, and the time "right now", or at any point in the oast, or at any point in the future. And there is a notion of agent: the person (or other) who (resp. which) performs actions on documents. A document is "full" of location, time and agent attributations. The location, time and agent notions deserve a special study.

2.2.1 Locations

Analysis: Points: We assume a space as a dense set of points, with a point being atomic, but not necessarily infinitisimally "small" in the physical sense of XYX dimension units. Thus any two points that are distinct at most shares that they may be more-or-less close, or more-or-less distant from one another. We may even introduce a notion of some point being "next to" another point, in which case there will probably be an indefinite if not infinite number of points close to a given point, and such that if two points π_i and π_j are next to one another and π_j and π_k are also next to one another, then either π_i is the same as π_k, or π_j is the same as π_k, or they are all three distinct.

Narration: Locations:

14. By a location we shall understand a dense set of points,
    (a) such that for every two distinct points π_i and π_k in the location (if the location contains more than one point)
        i. there is an ordered set of at least two distinct points in the location, $\{\pi_i, \pi_{i_1}, \ldots, \pi_{i_j}, \ldots, \pi_{i_{k-1}}, \pi_{i_k}\}$,
        ii. such that each pair of these points, $\pi_{i_{j-1}}$, π_{i_j}, or π_{i_j}, $\pi_{i_{j+1}}$,
        iii. for $i \leq j-1$ and $j \leq k$,
        iv. are 'next to' one another.
15. Two locations are 'different' if the two sets are different.
16. Two locations are 'distinct' if they are different and they have no points in common (that is, share no points).

17. Two locations are 'next to' one another
    (a) if they are different,
    (b) if there is at least a non-empty set of points in each location, ps_i and ps_j
    (c) whose points are "not in the location of the other",
    (d) such that pairs of points in ps_i and ps_j are "next to" one another
    (e) and such that all other points in the two locations are shared.
18. Two locations are 'neigbouring' if they are distinct and there is at least one point in each location, p_i and p_j such that these are "next to" one another.

Formalisation: Locations:

type
14 P, L
value
14 obs_Ps: L → P-**set**
axiom
14 ∀ ℓ:L • **let** ps = obs_Ps(ℓ) **in**
14a ∀ pi,pk:P • {pi,pk}⊆ps •
14(a)i **let** pl:P* • **elems** pl = ps ∧ **len** pl=**card** ps **in**
14(a)ii ∀ j:**Nat** • {j,j+1}⊆**inds** pl
14(a)iii next_to(pl(j),pl(j+1))
14(a)iv **end end**

Three location relations:

value
15 diff(ℓ$_i$,ℓ$_j$) ≡ obs_Ps(ℓ$_i$)≠obs_Ps(ℓ$_j$)

16 dist(ℓ$_i$,ℓ$_j$) ≡ obs_Ps(ℓ$_i$)isect obs_Ps(ℓ$_j$)={}

17 next_to(ℓ$_i$,ℓ$_j$) ≡
17a diff(ℓ$_i$,ℓ$_j$)∧
17b ∃ ps_i,ps_j:P-**set**•ps_i⊆obs_Ps(ℓ$_i$)∧ps_j⊆obs_Ps(ℓ$_j$)∧
17c ∀ p_i,p_j:P:p_i ∈ ps_i∧p_j ∈ ps_j•p_i∉ obs_Ps(ℓ$_j$)∧p_j∉ obs_Ps(ℓ$_i$)∧
17d next_to(p_i,p_j)∧
17e obs_Ps(ℓ$_i$)\{ps_i}=obs_Ps(ℓ$_j$)\{ps_j}

15 diff: L × L → **Bool**
16 dist: L × L → **Bool**
17 next_to: (P×P) | (L×L) → **Bool**

A remaining location relation:

18 neighbour: L × L → **Bool**
18 neighbour(ℓ$_i$,ℓ$_j$) ≡

© Dines Bjørner 2006, Fredsvej 11, DK–2840 Holte, Denmark

18 distinct(ℓ_i,ℓ_j) ∧
18 ∃ ps_i,ps_j:P-**set** •
18 ps_i⊆obs_Ps(ℓ_i)∧ps_j⊆obs_Ps(ℓ_j)∧ps_i≠{}∧ps_j≠{}∧
18 ∀ p_i,p_j:P:p_i ∈ ps_i∧p_j ∈ ps_j•p_i∉ obs_Ps(ℓ_j)∧p_j∉ obs_Ps(ℓ_i)∧
18 next_to(p_i,p_j)

2.2.2 Time and Time Intervals

Analysis: Time: Time, besides philosophically being an elusive issue, can also, as here, be taken concretely. We shall present a set of axioms due to Johan van Benthem [4]. We shall think of time to be a notion that is absolute wrt. some calendar, that is, time includes year, month, day, hour, minute, second, etc., in some simple encoded form.

Formalisation: Time:

axiom

[TRANS: Transitivity] ∀ p,p′,p″:P • p < p′ < p″ ⇒ p < p″

[IRREF: Irreflexitivity] ∀ p:P • p ≮ p

[LIN: Linearity] ∀ p,p′:P • (p=p′ ∨ p<p′ ∨ p>p′)

[L−LIN: Left Linearity]
∀ p,p′,p″:P • (p′<p ∧ p″<p) ⇒ (p′<p″ ∨ p′=p″ ∨ p″<p′)

[BEG: Beginning] ∃ p:P • ∼∃ p′:P • p′<p

[END: Ending] ∃ p:P • ∼∃ p′:P • p<p′

[SUCC: Successor]
[PAST: Predecessors] ∀ p:P,∃ p′:P • p′<p
[FUTURE: Successor] ∀ p:P,∃ p′:P • p<p′

[DENS: Dense] ∀ p,p′:P (p<p′ ⇒ ∃ p″:P • p<p″<p′)

[DENS: Converse Dense] ≡ [TRANS: Transitivity]
∀ p,p′:P (∃ p″:P • p<p″<p′ ⇒ p<p′)

[DISC: Discrete]
∀ p,p′:P • (p<p′ ⇒ ∃ p″:P • (p<p″ ∧ ∼∃ p‴:P • (p<p‴<p″))) ∧
∀ p,p′:P • (p<p′ ⇒ ∃ p″:P • (p″<p′ ∧ ∼∃ p‴:P • (p″<p‴<p′)))

Comments on the Axiomatisation: Time: A *strict partial order, SPO,* is a point structure satisfying TRANS and IRREF. TRANS, IRREF and SUCC imply infinite models. TRANS and SUCC may have finite, "looping time" models.

We choose the *SPO* model of time.

Analysis: Time Intervals: Time is "absolute" wrt. some "zero". A time interval may be measured in years, months, etc., but these are year intervals, month intervals, etc., not "absolute" wrt. some calendar. When we subtrate one time (a date etc.) from another such time we get a time interval. We cannot add time, but we can add a time interval to a time and get a time. We can add, subtract and multiply time intervals. If we divide a time interval by a non-zero such we get a fraction, i.e., a real number. Etcetera. At the stage of this paper (/today) we shall not need the notion of time interval, nor the operations on times.

Formalisation: Time and Time Intervals:

type
  T, TI
value
  elapsed_time: T × T → TI
  −: (T|TI) × TI → TI
  −: T × T → TI
  +: T × TI → T
  +: TI × TI → TI
  ∗: TI × **Real** → TI
  /: TI × TI $\xrightarrow{\sim}$ **Real**
axiom
  ∀ t,t′:T • t′>t ⇒ ∃ tδ:TI • tδ = t′−t
  ∀ t:T,tδ:TI • ∃ t′:T • ⇒ t+tδ = t′ ∨ t−tδ = t′
  ∀ ti,ti′:TI • ∃ ti″:TI • ti+ti′ = ti″ ∨ ti−ti′ = ti″
  ...

2.2.3 Time and Space

Analysis: Time and Space: We show an analysis of some time/spce notions. The analysis is due to Wayne D. Blizard [5]. In the axiomatisation below entities are the documents (or agents, A and B) and points are (our) locations (p, q and r), with A^t_q expresses that an entity A at time t is at location p (axiom (I)). See our comments below.

Formalisation: Time and Space:

(I)		$\forall A \forall t \exists p$	:	A^t_p	
(II)		$(A^t_p \wedge A^t_q)$	$\supset$	$p = q$	
(III)		$(A^t_p \wedge B^t_p)$	$\supset$	$A = B$	
(IV)		$(A^t_p \wedge A^{t'}_p)$	$\supset$	$t = t'$	
$(V$	$i)$	$\forall p, q$	:	$N(p,q) \supset p \neq q$	Irreflexivity
$(V$	$ii)$	$\forall p, q$	:	$N(p,q) = N(q,p)$	Symmetry
$(V$	$iii)$	$\forall p \exists q, r$	:	$N(p,q) \wedge N(p,r) \wedge q \neq r$	No isolated pts.
$(VI$	$i)$	$\forall t$	:	$t \neq t'$	
$(VI$	$ii)$	$\forall t$	:	$t' \neq 0$	
$(VI$	$iii)$	$\forall t$	:	$t \neq 0 \supset \exists \tau : t = \tau'$	
$(VI$	$iv)$	$\forall t, \tau$	:	$\tau' = t' \supset \tau = t$	
(VII)		$A^t_p \wedge A^{t'}_q$	$\supset$	$N(p,q)$	
$(VIII)$		$A^t_p \wedge B^t_q \wedge N(p,q)$	$\supset$	$\sim (A^{t'}_q \wedge B^{t'}_p)$	

© Dines Bjørner 2006, Fredsvej 11, DK–2840 Holte, Denmark

Annotations: Time and Space:

- (II–IV,VII, VIII): The axioms are universally 'closed', that is, we have omitted the usual $\forall A, B, p, q, t$s.
- (I): For every entity, A, and every time, t, there is a location, p, at which A is located at time t.
- (II): An entity cannot be in two locations at the same time.
- (III): Two distinct entities cannot be at the same location at the same time.
- (IV): Entities always move: An entity cannot be at the same location at different times. *This is more like a conjecture, and could be questioned.*
- (V): These three axioms define N.
- (V i): Same as $\forall p :\sim N(p, p)$. "Being a neighbour of", is the same as "being distinct from".
- (V ii): If p is a neighbour of q, then q is a neighbour of p.
- (V iii): Every location has at least two distinct neighbours.
- (VI): The next four axioms determine the time successor function $'$.
- (VI i): A time is always distinct from its successor: Time cannot rest. There are no time fix points.
- (VI ii): Any time successor is distinct from the begin time. Time 0 has no predecessor.
- (VI iii): Every nonbegin time has an immediate predecessor.
- (VI iv): The time successor function $'$ is a one-to-one (i.e., a bijection) function.
- (VII): The *continuous path axiom:* If entity A is at location p at time t, and it is at location q in the immediate next time t', then p and q are neighbours.
- (VIII): *No "switching":* If entities A and B occupy neighbouring locations at time t the it is not possible for A and B to have switched locations at the next time t'.

Discussion of the *Blizard* Model of Space/Time: Except for axiom (IV) the system applies to systems of entities that "sometimes" rest, i.e., do not move. These entities are spatial and occupy at least a point in space. If some entities "occupy more" space volume than others, then we may suitably "repair" the notion of the point space P (etc.), however, this is not shown here.

2.2.4 Agents

Analysis: Agents: Agents create and perform operations on documents. Typically agents are humans — but could be machines, i.e., computers.

Narrative: Agents:

19. An agent, besides having an identity, a:A, is a further undefined notion.
20. No two agents have the same identity, so if two arbitrarily chosen agents (one could "choose" the same agenet twice) are different, then they ar at least different because of their distinct identity.
21. One could choose to associate various other attributes with agents:
    (a) the set of documents in *possession* of the agent, say referenced by unique document identifiers,
    (b) the current location of the agent,
        i. with the possibility of also constraining agents to occupy distinct locations,
    (c) the granted authority to perform certain (or all) operations on certain (or all), including creating, documents

    but we choose to not deal with that issue in this paper.

Formalisation: Agents:

type
19. AG, A
value
19. obs_A: AG → A
axiom
21. ∀ a,a′:A • a=a′ ≡ obs_A(a)=obs_A(a′) ∧ obs_A(a)≠obs_A(a′)⇒a≠a′
value
21a. obs_Ds: AG → D-**set**
21b. obs_L: AG → L
axiom
21(b)i. ∀ a,a′ • a≠a′ ⇒ distinct(obs_L(a),obs_L(a′))
type
Auth
value
21c. obs_Auths: AG → Auth-**set**

2.3 Spaces of Documents and Agents

2.3.1 Narrative: Spaces of Documents and Agents

22. There is a concept of space.
23. Documents and agents coexist in space.
24. All documents and all agents in that space
    (a) have distinct unique document identifiers and
    (b) distinct agent identifiers.

© Dines Bjørner 2006, Fredsvej 11, DK–2840 Holte, Denmark

25. Some properties of documents can now be re-expressed

    No two document can have the same

    i. "most recent time and location",
    ii. no two document can have the same "most recent location and agent", and
    iii. no two document can have the same "most recent time and agent"[11].

(a) We also need to redefine the signature of all document operations — and the related axioms.

2.3.2 Formalisation: Spaces of Documents and Agents

type
22. Ω
    LT = L×T, LA=L×A, TA=T×A
value
22. obs_Ds: Ω → D-**set**, obs_As: Ω → AF-**set**
    obs_LT: D → LT, obs_LA: D → LA, obs_TA: D → TA,
axiom
24. ∀ ω:Ω,d,d′:D, ag,ag′:AG • {d,d′}⊆obs_Ds(ω) ∧ {ag,ag′}⊆obs_AGs(ω) ⇒
24a. d=d′ ≡ obs_UDI(d)=obs_UDI(d′) ∧
24b. ag=ag′ ≡ obs_A(ag)=obs_A(ag′) ∧
25. ∀ ω:Ω,d,d′:D • {d,d′}⊆obs_Ds(ω) ⇒
25()i. d≠d′ ⇒ obs_LT(d)≠obs_LT(d′) ∧
25()ii. d≠d′ ⇒ obs_LA(d)≠obs_LA(d′) ∧
25()iii. d≠d′ ⇒ obs_TA(d)≠obs_TA(d′) ∧

2.3.3 Formalisation: Types of Document Operations

value
5a. crea: LTA → Ω × UDI
5b. edit: LTA → UDI → Ω → Ω × E
5c. read: LTA → UDI → Ω → Ω
5d. copy: LTA → UDI → Ω → Ω × UDI
5e. shrd: LTA → UDI → Ω → Ω
axiom
... /∗ left as an exersize ! ∗/

3 Conclusion

3.1 What Have We Shown?

We have established a framework for reasoning about a concept of documents: orginals, unedited and edited documents, documents that have been read or not read, copies of master

[11] Since our axioms implicitly enjoy the universal time quantitifcation this means that no two past operations can have the take place at the same location, at the same location and time and agent.

documents, etc. We have endowed documents with such attributes as unique document identifiers, the location, time and agent of operations performed on documents, the 'kind of operation' ("most recently") performed on documents, document history, etc.

And we have "embedded" documents and agents in a space of such while expressing some, but (we think) far from all relevant properties (axiomatically) . In other words, we have started establishing a domain theory for the kind of documents that are of the kinds and have the operations and properties expressed in this paper.

The formalisation of the domain description has been carried out using the RAISE specification labuage, RSL [6, 7]. Emphasis has been on an algebraic style specification using sorts, observer functions and axioms.

3.2 Shortcomings

Immediate shortcomings are believed to be: uncertainty as to whether all relevant properties (axioms) have been formulated, uncertainty as to whether the axioms that have been expressed are consistent (they are not thought of as being complete). Less immediate shortcomings, we think, have to do with the following: Have we expressed the sorts, function signaturs and the axioms as succinctly as we would desire? A "gut feeling" about issues that we ought to have also covered!

3.3 What Needs be Done?

Obviously we need to now review the paper. And to improve it along the lines outlined in the previous section. This will take time. Hopefully colleagues will wish to study this paper. And hopefully the author — and the paper — can then benefit from resulting discussions.

4 Acknowledgements

The current author acknowledges the support of JAIST, the Graduate School of Information Science — where the writing of the current report was carried out; the very preliminary discussions the author has had with colleagues, in particular Kokichi Futatsugui and Kuzuhiro Ogata; and the IBM TRL for providing a stimulus for getting this paper ready for 9 August 2006.

References

[1] Dines Bjørner. *Software Engineering, Vol. 1: Abstraction and Modelling*. Texts in Theoretical Computer Science, the EATCS Series. Springer, 2006.

[2] Dines Bjørner. *Software Engineering, Vol. 2: Specification of Systems and Languages*. Texts in Theoretical Computer Science, the EATCS Series. Springer, 2006. Chapters 12–14 are primarily authored by Christian Krog Madsen.

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

[4] Johan van Benthem. *The Logic of Time*, volume 156 of *Synthese Library: Studies in Epistemology, Logic, Methhodology, and Philosophy of Science (Editor: Jaakko Hintika)*. Kluwer Academic Publishers, P.O.Box 17, NL 3300 AA Dordrecht, The Netherlands, second edition, 1983, 1991.

© Dines Bjørner 2006, Fredsvej 11, DK–2840 Holte, Denmark

[5] Wayne D. Blizard. A Formal Theory of Objects, Space and Time. *The Journal of Symbolic Logic*, 55(1):74–89, March 1990.

[6] Chris W. George, Anne Elisabeth Haxthausen, Steven Hughes, Robert Milne, Søren Prehn, and Jan Storbank Pedersen. *The RAISE Method*. The BCS Practitioner Series. Prentice-Hall, Hemel Hampstead, England, 1995.

[7] 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.

© Dines Bjørner 2006, Fredsvej 11, DK–2840 Holte, Denmark