Documents – a Domain Description

DINES BJØRNER, Technical University of Denmark – February 22, 2025, Denmark

We analyze and describe a conceptual domain of *documents*. Their *creation, editing (writing, updating), reading (viewing), copying, deletion* and *contents*. The special "features" of the documents, that we have in mind, are that all of these operations and the documents they apply to and/or result in can be traced – including as to who performed (*handlers*) those operations and at which times.

Dines Bjørner. 2025. Documents – a Domain Description. 1, 1 (February 2025), 20 pages.

1 WHAT ARE DOCUMENTS, INFORMALLY ?

The documents that we, foremost, have in mind are typically those of public government, legal documents pertaining to the three branches of government: the legislative branch, the executive branch, and the judicial branch[1]. But our considerations apply also to more mundane documents: those of an writer keeping track of versions of some writing, those of a scientist keeping track of year-long versions of some scientific paper[2].

There are three notions related to documents: (i) *existing documents,* i.e., created, "worked upon", but not [yet] deleted documents. Within such documents there may be copies: copies of a *master document,* i.e., a document which is not itself a copy of some document; or there may be copies of copies of ... documents ! (ii) *deleted documents,* i.e., documents that have "existed", but can no longer be operated upon. (iii) And document *versions:* By a *version* we understand an attribute of a document. By *document version***s** we shall understand a sequence of one or more *version*s. More later !

So documents to us, in this paper, are conceptual entities, first as *endurant*s, then as more specifically *part*s, finally as *perdurant*s, i.e., *behaviours actions* in the sense of [6]. They have *unique identification*, relates to other documents, i.e., have *mereology*, and have *attribute*s: versions, contents; access authorization, i.e., who may perform which operations on documents; history, i.e., *time-stamped* sequence of operations performed on the documents; etc.

We use the plural, 'documents', since it makes little sense to speak of only one document. So there is a "space" of documents. You may think of this space to be "in-the-cloud".

Documents are handled by *document handlers*. They perform the operations of *creating, editing, reading, copying, deleting* and *setting document access authorizations*. Documents are not thought of as physical, in the sense of being printed on paper. In this day and age of computing, of vast memory capacities and of data communication (Internet), we may be allowed to think of "our" documents as being abstract entities "in space – in the cloud". So handlers are allowed to think and talk of any document as it has evolved over time: *"Yes, I recall that this document was handled by*

[1] The term *trias politica* or *separation of powers* was coined by Charles-Louis de Secondat, baron de La Brède et de Montesquieu, an 18th century French social and political philosopher.

[2] My attempts to understand documents in a widest sense started in 1994, when I was the UN Director of the UN University's Intl. Inst. for Software Technology in Macau; then restarted in 2006, during my stay at JAIST in Japan, and is "documented" in [7]; a third attempt was made in 2017, while I prepared for my fall visit to TongJi Univ., Shanghai [2]. The present effort started February 14, 2025 – and appears to be my last "try" !

Author's address: Dines Bjørner, Technical University of Denmark – February 22, 2025, DTU Compute, Fredsvej 11, Holte, 2840, Denmark, bjorner@gmail.com.

'such-and-such' a handler at 'such-and-such' a time and in the following way '...' ". Such recollections are, in a world of paper documents, not always "recordable". In this treatment of documents we shall model such a domain. "Finally"[3] there is *concurrency*: the facts that two or more handlers may handle the same document "simultaneously" !

We shall unfold or story on documents in two stages. In Sect. 2, an informal stage, we start with documents an move on to their handlers. In Sect. 3 on page 5 we then start all over with a formal domain description.

In this report we use many technical terms. Some are from the domain of document handling. Others from the ontology of domain descrition. Yet others are from the formal description of the domain, i.e., Sect. 3. Etcetera. We provide therefore a *glossary* in Appendix A (pages 13–15). There is also three sets of indexes to their use, Appendix B (pages 15–18).

1.1 Document Operations

These are the operations that we have in mind:

- `create document`,
- `begin document display`,
- `edit contents(insert,delete,change)`,
- `version document content`,
- `copy document`,
- `grant document permits`,
- `end document display`, and
- `delete document`.

While *reading/displaying* a *version* of a *document* a *handler* may therefore *edit* that *version*, may *version* it, and henceforth *read, edit,* etc., that latest *version*, may *copy* it, may change its *authorisation*, i.e., *access rights*, and may *delete* it ! In overlapping time intervals that handler may thus handle more than one document.

1.2 Document Handlers

By a document handler we mean an entity, for example a human being. more to come

1.3 Concurrency

This subsection attempts to summaruze some cocurrency aspects of document handlin.

More to come

1.4 Commands

A document handling command is a syntactic quantity ! Document commands are issued by uniquely identified handlers and are directed at uniquely identified documents. Such commands have three elements: a handler identifier, the operation designation, and a document & version identifier.

2 A FIRST TAKE ON DOCUMENTS AND THEIR OPERATIONS

The purpose of this secion is to further encircle a, or "our", concept of document handling. This is in preparation for a "full", formal treatment.

[3] Well, for this introductory section ! Additonal document domain concepts will emerge later.

2.1 Documents

We shall in this section deal only with the endurant aspects of documents: their external and internal qualities. The perdurant aspects, i.e., actions and behaviours will be toched upon only indirectly – with more to come later !

2.1.1 ***External Qualities.*** Documents are entities. First we look at documents as endurants; subsequently as perdurants, i.e., behaviours that interact with document handler behaviours.

Any freshly `created` document becomes an *existing document* of *version # 1*. Subsequent versions of such an existing document becomes *version # 1*, *version # 2*. ..., *version # n*. All these versions have the same unique identifier. The `edit` and `authorisation` operations are wrt. to the most recent version. `read`, `terminate read` and `copy` operations on a uniquely identified document can be with respect to any `version` of that document.

2.1.2 ***Internal Qualities.*** These are some internal qualities of documents:

- `unique identification:` Each document has a unique identification. We need not bother about how that unique identification is represented. The copy of a document identified by a *ui* "receives, "miraculously", an identification distinct from all other documents so far created ! The unique identifiers of *deleted* documents are never reused. The identifiers of a document and its versions are identical ! But those of their copies are distinct.
- `mereology:` The mereology of a document specifies two things: who handles the document, and which other documents it is either created from, and/or whose "text" content it refers to, or which have references to its "text" content. Document mereologies are dynamic in that they change – as we shall see.
- `attributes:` Documents have many attributes:
  - *Document Title:* A document has a title. A title is a sequence of words, i.e., plain text.
  - *Document Abstract:* A document usully has an *abstract*. The astract is a short plain text.
  - *Identification:* By *identification* we mean not only that a document has a unique identifier, but also an *authorship* (one or more handlers), with their *names*, professional titles, institution (affiliation), authority, address, telephone number, e-mail address and URL. Some of these may be optional.
  - *Document References:* A document reference is a se of zero, one or more "references" to other, preceding or expected documents. These references may be in the form of their document titles, authorship, version #, and unique identifier.
  - *Contents:* By contents we shall mean texts, music, videos. More later.
  - *Access Authorization:* A map from document handler names (etc.) to their access rights in terms of which operations (and their restrictions) they may perform on said document.
  - *Operation History:* A time-ordered sequence of time-stamped and handler-identified entries which designate the times at which identified handlers performed designated operations (with their operands) on said document.
- `intentional pull:` Documents are entities "in their own right". But documents are intended to be handled, created, etc., by handlers. At any time a document may be handled by zero, one or more handlers. At the same time handlers are in the process of handling zero, one or more documents. These two relations form an *intentional pull*.

2.2 Document Operations

In Sect. 1.1 on page 2 wefirst mentioned the contemplated document operations:

- `create document`,
- `begin document display`,
- `edit contents` (`insert`,`delete`,`change`),
- `version document content`,
- `copy document`,
- `grant document permits`,
- `end document display`, and
- `delete document`.

These are the operations that we have in mind.

(1) `create`: A handler decides, and we shall not bother why, to create a new document. From scratch, tough that new, thereby existing, document may, from the start be based on, i.e., refer to one or more existing documents. The creation operation is an action of the handler behaviour, and is instantaneous. That is, it can be considered as taking no time. At the time of creation the creator handler may, or may not, endow that new document with specific access rights: the identity of such handlers, including the creator, and the operations they are allowed to perform on the created, new document. We shall later elaborate on the concept of 'access authorization'. The newly created document has its history attribute initialized to the one element sequence of the time of creation and identity of handler.
We take up the formal syntax this commnd in Sect. 3.2.1 on page 12.

(2) `begin display`: (or `view/display`) A handler decides to read/view/display a document, say on the screen of a computer. That document, with or without a full or partial subset of its internal qualities: contents, past history of operations, access rights, is then, somehow, presented to that handler. Two or more handlers may, dependent on access rights, read/view/display the requested document in overlapping time intervals. The action is therefore not instantaneous. The time-interval from requesting the 'read' to its 'termination' can, dependent on access rights, be of any length. The read document has its history attribute augmented to reflect the beginning time of being read and identity of handler.
We take up the formal syntax this commnd in Sect. 3.2.2 on page 12.

(3) `edit`: (or `write/update`) A handler, while reading/viewing a specific document, decides to "add contents", i.e., to edit/write/update a specific document. It is important to maintain that edit actions on an existing document can only take place while that document is being read (viewed). The edit operation take no time, is instantaneous. That is: we abstract from the often time-consuming "affair" of formulating, i.e., figuratively "typing" in the edit-material. The edit document has its history attribute augmented to reflect the time of edit and identity of handler.
We take up the formal syntax this commnd in Sect. 3.2.3 on page 12.

(4) `version`: A handler, while reading/viewing/editing a specific document, decides "to take a break" and "save", not as a copy, i.e., as a "free-standing document copy", but as an "intermediary" *version* of the document being worked upon. ... MORE T COME ...
We take up the formal syntax this commnd in Sect. 3.2.4 on page 12.

(5) `copy`: A handler decides to copy an existing document in one or more copies – dependent on access rights. The result of a copying can be considered instantaneous: One or more new, thereby existing documents are 'created'. The base document, i.e., the document being copied has its document history record the event: time of copying, and number and identity of

the copied new documents. Each of the newly created copies have their history attribute initialized to the one element sequence of the time of being copied and identity of handler.
We take up the formal syntax this commnd in Sect. 3.2.5 on page 12.

(6) `authorization:` (or `permits`) to come
We take up the formal syntax this commnd in Sect. 3.2.6 on page 12.

(7) `end display:` A handler, in the process of displaying an existing document, eventually decides to end that display, i.e., to terminate it. The display "disappears". The action is instantaneous. The read-terminated document has its history attribute augmented to reflect the end time of being displayed and identity of handler.
We take up the formal syntax this command in Sect. 3.2.7 on page 12.

(8) `delete:` A so authorized handler decides to delete a document. The identified document ceases to exist as an existing document and becomes a *deleted document*. Deleted documents retain all their properties, i.e., internal qualities: identity, mereology and attributes. Their history does reflect the 'deletion' event: time and handler. Deleted documents cannot be operated upon by [ordinary] handlers, i.e., the handlers we have dealt with so far.
We take up the formal syntax this commnd in Sect. 3.2.8 on page 12.

While *reading/displaying* a *version* of a *document* a *handler* may therefore *edit* that *version*, may *version* it, and henceforth *read, edit,* etc., that latest *version*, may *copy* it, may change its *authorisation*, i.e., *access rights*, and may *delete* it ! In overlapping time intervals that handler may thus handle more than one document.

2.3 Document Handlers

By a document handler we mean an entity, for example a human being. more to come

2.4 Concurrency

This subsection attempts to summaruze some cocurrency aspects of document handlin.

More to come

2.5 Commands

A document handling command is a syntactic quantity ! Document commands are issued by uniquely identified handlers and are directed at uniquely identified documents. Such commands have three elements: a handler identifier, the operation designation, and a document & version identifier.

3 A FORMAL DOCUMENT DOMAIN DESCRIPTION

We shall subject a concept of *document handling domains* according to the analysis ontology of Fig. 1 on the next page.

3.1 Endurants

3.1.1 ***External Qualities***.

(9) There is a domain of document handling, $dh : DH$[4].

(10) From $dh : DH$ we can observe
  (a) a collection of documents, $cd : CD$, and
  (b) a group of handlers, $gh : GH$.

[4]By $t : T$ we mean an instance of a value, t, of type T.

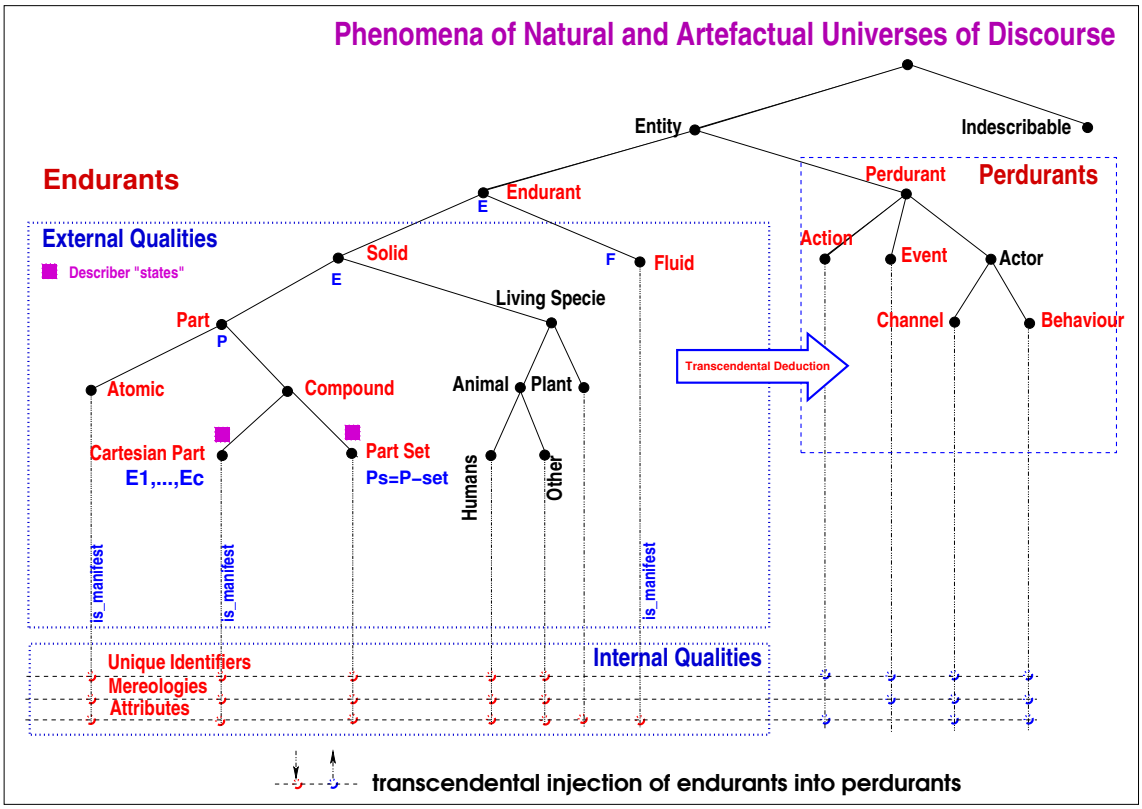

Fig. 1. An Ontology for Analyzing Manifest Domains

(11) From a collection of documents we can observe two sets of documents:
  (a) created, but not [yet] deleted, *ed* : *ED*,
  (b) created, but deleted, *dd* : *DD*, documents.
(11) From a group of handlers we can observe its set, *hs* : *HS*, of handlers.
(12) From documents we can observe a sequence of one or more versions, *vl*:*VL*.
(13) Versions are atomic.
(14) Handlers are atomic.

```
type
9.  DH
10a.  CD
10b.  GH
10a.  ED = D-set
10b.  DD = D-set
11.   HS = H-set
13.   D
14.   H
value
10a.  obs_CD: DH → CD
10b.  obs_GH: DH → GH
10a.  obs_ED: CD → ED
10b.  obs_DD: CD → DD
11.   obs_HS: GH → HS
```

(15) The two sets, *de* : *ED* and *dd* : *DD* are disjoint.

```
axiom [Unique Identification]
```

```
15.  ∀ cd:CD • obs_ED(cd) ∩ obs_DD(cd) = {}
```

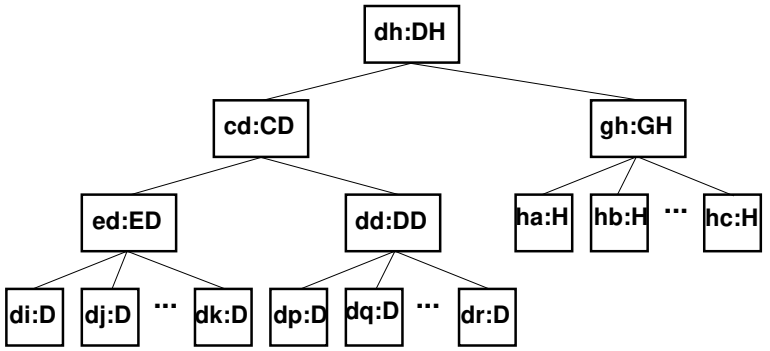

Fig. 2. A Document Handling Domain Taxonomy

An Endurant State.

(16) Technically speaking, i.e., through the use f the formal specification language RSL [11], ad for convenience we can refer to the endurants of our interest as *parts*.

type

```
16.  P = DH | CD | GH | ED | DD | HS | C | H
```

We postulate an abstract, immaterial construct: a document domain state. That state consists of

(17) the document domain,
(18) its collection of documents with
  (a) their set of existing documents,
  (b) their set of deleted documents;
(19) its group of handlers, with
  (a) their set of handlers.

value

```
17.  dh:DH
18.  cd:CD = obs_CD(dh)
18a.  ed:ED = obs_ED(obs_CD(dh))
18b.  dd:DD = obs_DD(obs_CD(dh))
19.  gh:GH = obs_GH(dh)
19a.  hs:HS = obs_HS(gh)
```

(20) We can therefore define an endurant part state "collection" function:

value

```
20.  σ:P-set = {dh}∪{cd}∪{ed}∪{d|d:D•d ∈ ed}∪{dd}∪{d|d:D•d ∈ dd}∪{gh}∪{h|h:H•h ∈ gh}
```

3.1.2 ***Internal Qualities.***

Unique Identification.

(21) Each of the endurant part sorts ahve their unique identifiers.
(22) We "unite" all unique identifiers into one type: *ui*:*UI*.
(23) All endurants, i.e., the document handling domain, the collection of documents, the documents, the group of handlers and the handlers, all have distinct, i.e., unique identifiers.
(24) . We can also express this in terms of parts *p*:*P*.

```
type
21. DHI,CDII,GHI,EDI,DDI,DDI,HSI,DI,HI
22. UI = DHI|CDI|GHI|EDI|DDI|DI|HI
value
23. uid_DH: DH → DHI
23. uid_CD: CD → CDI
23. uid_GH: GH → GHI
23. uid_ED: ED → EDI
23. uid_DD: DD → DDI
23. uid_HS: HS → HSI
23. uid_D: D → DI
23. uid_H: H → HI
24. uid_P: P → UI
value
23. σui:UI-set =
23.   {uid_DH(dh)}
23. ∪ {uid_CD(cd)}
23. ∪ {uid_ED(ed)}∪{uid_ED(d)|d:ED•d∈ed}
23. ∪ {uid_DD(dd)}∪ { uid_DD(d)|d:DD•d∈dd}
23. ∪ {uid_GH(gh)}∪{uid_H(h)|h:H•h∈hs}
24.     = {uid_P(p)|p:P•p∈ σ}
```

axiom [Unique Identification]
23. **card** σ = **card** σ_{ui}

Mereology. Mereology, as introduced by the Polish mathematician *Stanisław Leśniewski (1886–1939)* [1, 10], is the study and knowledge of parts and part relations.

(25) The mereology of a document is a, or the, set of the unique identifiers of the handlers that may access that document.
(26) The mereology of a handler is triplet of (i) the set of the unique identities of the documents that the handler may access, (ii) the identity of the defined document endurant, *dd*:*DD*, the identity of the group of handlers, *gh*:*GH*, and (iii) the identity of the defined documents, *dd*:*DD*. Perhaps more ! (...)
(27) The mereology of a defined document part is a triplet of (i) a set of defined document identifiers, (ii) the set of handler identifiers, and (iii) the identity of the collection of document part.
(28) The mereology of a collection of documents is a triplet of the defined documents part, the handlers, and the document handler.
(29) The mereology of the group of handlers is a pair of all the handler identifiers and the document handler identifier.
(30) The mereology of the document handler identifier. is a pair of te identifiers of the collection of documents and the group of handlers.

```
type
25. DM = HI-set
26. HM = DI-set × GHI × DDI × ...
27. DDM = DI-set × HI-set × CDI
28. CDM = DDI × HI-set × DHI
29. GHM = HI-set × DHI
30. DHM = CDI × GHI
value
25. mereo_D: D → DM
26. mereo_H: H → HM
27. mereo_DD: DD → DDM
28. mereo_CD: CD → CDM
29. mereo_GH: GH → GHM
30. mereo_DH: DH → DHM
```

Attributes.

- **Documents:**

(31) A document has a list, a sequence, of one or more *version*s. The time-wise most recent Version is a programmable attribute. [5]
(32) A version has *time of being "versioned"*.
(33) A version has *contents*. We shall define the concept of 'content' later, see Items 37, 38, etc., below.

π

[5] Margin σs and πs shall alert the developer as to the attribute category.

(34) A document has access limitations: Some, but perhaps not all, can read, and/or edit, ad/or copy, and/or delete, and/or prescribe access rights. We shall later define, more precisely, what access right can be formulated. DocAccessRight is a programmable attribute. π

(35) A document has a history. A document history records, as a time-ordered sequence of "events" the actions that has been performed on the document: the time of action, the kind of action/operation performed, by whom, "et cetera". We shall later explain the "et cetera". DocHistory is a programmable attribute. π

(36) Et cetera.

That is:

type
```
31. Versions = Version*
32. VersionTime = TIME
33. VersionContent
34. DocAccessRight
35. DocHistory = Event*
35. DocEvent = (Date×TIME) × HI × Operation × ...
35. Operation [see Item 2.2 on page 4]
36. etc.
```
value
```
31. attr_Versions: D → Versions
32. attr_DocDateTime: D → Date × TIME
34. attr_DocAccessRight: D → AccessRight
35. attr_DocHistory; D → DHistory
36. ...
```
axiom [Time-ordered Document History]
```
35. ∀ dh:DocHistory • ∀ {i,i+1}⊆inds(dh) ⇒
35.     let (τ,...)=dh[i], (τ′,...)=dh[i+1] in τ<τ′ end
```

Contents

(37) By Contents (plural) we shall presently[6] Contents is a programmable attribute. understand an indexed collection of cont:Content (singular).

(38) By a Content we shall presently understand either a txt:Text, or a fig:Figure, or a tbl:Table, or an img:Image (say a photo), or a vid:Video, or a piece of mus:Music !

(39) By a txt:Text we shall understand a sequence of identified Segments such that no two segments have the same segment identifier.

(40) By a Segment we shall, recursively, understand a PlainText, or an itemized (•'ed) Text, or an enumerated (1, 2, ...) Text, or a Text, such that this embedding of [itemized or enumerated, or ...] Texts is finite.

(41) By PlainText we shall understand a more to come

(42)

(43)

type
```
37. Contents = Idx →m Content
38. Content = Text | Figure | Table | Image | Video | Music
```

[6] We use term 'presently' to alert the reader to the situation that one, he, she, I, might wish to define Contents and the below Content, Text, etc., differently.

```
39. Text = (Sid×Segment)*
39. Sid = ...
39. Segment = PlainText | Text
38. Figure = ...
38. Table = (RowId →m (ColId →m Text))
38. RowId = ...
38. ColId = ...
38. Image = ...
38. Video = ...
38. Music = ...
41. PlainText = ...
value
37. attr_Contents: D → Contents
```

Access Rights

(44) By AccessRights we mean a set of zero, one or more non-conflicting AccessRights ! AccessRights is a programmable attribute.
(45) By an AccessRight we mean either a right to (i) edit some of that documents contents indexed contents, (ii) and if that content is a text, then some (or all) of that text's indexed segments, or (iii) replace a figure, a table, an image, a video or a music !
(46) By a TextPermit we shall mean ...
(47) By a ReplacePermit we shall mean ...

```
type
44. AccesRights = AccessRight-set
45. AccessRight = TextPermit | ReplacePermit
46. TextPermit = ...
47. ReplacePermit = ...
value
44. attr_AccessRights: D → AccesRights
```

- **Handlers:**

(48)
(49)
(50)
(51)
(52)

```
type                                value
48.                                 48. attr_
49.                                 49. attr_
50.                                 50. attr_
51.                                 51. attr_
52.                                 52. attr_
```

- **Existing Documents:**

(53)
(54)
(55)

(56)
(57)

type	value
53.	53. **attr_**
54.	54. **attr_**
55.	55. **attr_**
56.	56. **attr_**
57.	57. **attr_**

Intentional Pull.

To be written

3.2 Commands

3.2.1 ***Create.*** [Cf. Item 1 on page 4]
See also Sect. 3.3.3.

3.2.2 ***Begin Display.*** (read) [Cf. Item 2 on page 4]
See also Sect. 3.3.3.

3.2.3 ***Edit.*** (or `write/update`) [Cf. Item 3 on page 4]
See also Sect. 3.3.3.

3.2.4 ***Version.*** [Cf. Item 4 on page 4]
See also Sect. 3.3.3.

3.2.5 ***Copy.*** [Cf. Item 5 on page 4]
See also Sect. 3.3.3.

3.2.6 ***Set Permits.*** [Cf. Item 6 on page 5]
See also Sect. 3.3.3.

3.2.7 ***End Display.*** [Cf. Item 7 on page 5]
See also Sect. 3.3.3.

3.2.8 ***Delete.*** [Cf. Item 8 on page 5]
See also Sect. 3.3.3.

3.3 Perdurants

Manifest [endurant] parts can be transcendentally deduced into perdurant behaviours, some pro-active, some re-active. For example: handlers into pro-active behaviours, documents into re-active behaviours.

3.3.1 ***An Interaction Space.*** In previous writings, [6, 9], the CSP [13] notions of channels, ch[{i,j}], and channel actions: ch[{i,j}] ! value ("output") and ch[{i,j}] ? ("input") were use to describe communication between domain behaviours. In this paper we shall use the term *communication medium*:

(58) The document handlng **comm**unication medium is modelled in terms of a CSP channel array indexed by a "pair", really set, of two part identifiers:

```
58. channel { comm[ {i,j} ]|{i,j}:UI•{i,j}⊆σui }
```

3.3.2 ***Behaviours.***

3.3.3 ***Actions.***

Create. [Cf. Item 1 on page 4 and Sect. 3.2.1.]

Begin Display. (read) [Cf. Item 2 on page 4 and Sect. 3.2.2.]

Edit. (or `write/update`) [Cf. Item 3 on page 4 and Sect. 3.2.3.]

Version. [Cf. Item 4 on page 4 and Sect. 3.2.4.]

Copy. [Cf. Item 5 on page 4 and Sect. 3.2.5.]

Set Permits. [Cf. Item 6 on page 5 and Sect. 3.2.6.]

End Display. [Cf. Item 7 on page 5 and Sect. 3.2.7.]

Delete. [Cf. Item 8 on page 5 and Sect. 3.2.8.]

3.3.4 ***Domain Initialization.***

4 CONCLUSION

5 BIBLIOGRAPHY

REFERENCES

[1] Dines Bjørner. A Rôle for Mereology in Domain Science and Engineering. In *Mereology and the Sciences*, Synthese Library (eds. Claudio Calosi and Pierluigi Graziani), pages 323–357, Amsterdam, The Netherlands, October 2014. Springer. https://www.imm.dtu.dk/~dibj/2011/urbino/urbino-colour.pdf.

[2] Dines Bjørner. What are Documents ? www.imm.dtu.dk/~dibj/2017/docs/docs.pdf. Research Note, Technical University of Denmark, Fredsvej 11, DK-2840 Holte, Denmark, July 2017.

[3] Dines Bjørner. *Domain Science & Engineering – A Foundation for Software Development.* EATCS Monographs in Theoretical Computer Science. Springer, Heidelberg, Germany, 2021. A revised version of this book is [5].

[4] Dines Bjørner. Domain Modelling – A Primer. A short and significantly revised version of [3]. xii+202 pages[7], May 2023.

[5] Dines Bjørner. Domain Science & Engineering – A Foundation for Software Development. Revised edition of [3]. xii+346 pages[8], January 2023.

[6] Dines Bjørner. Domain Modelling. *Submitted to ACM FAC*, page 18, February 2025. Institute of Mathematics and Computer Science. Technical University of Denmark.

[7] Dines Bjørner. *[8] Chap. 8: Public Government – A Rough Sketch Domain Analysis*, pages 201–222. JAIST Press, March 2009.

[8] Dines Bjørner. *Domain Engineering: Technology Management, Research and Engineering.* A JAIST Press Research Monograph # 4, 536 pages, March 2009.

[9] Dines Bjørner and Yang ShaoFa. Domain Modelling. Technical University of Denmark. Revised edition of [3]. xii+208 pages. https://www.imm.dtu.dk/~dibj/2023/dommod/dommod.pdf, May 2023.

[10] Roberto Casati and Achille C. Varzi. *Parts and Places: the structures of spatial representation.* MIT Press, 1999.

[11] 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.

[12] Charles Anthony Richard Hoare. *Communicating Sequential Processes.* C.A.R. Hoare Series in Computer Science. Prentice-Hall International, 1985.

[13] Charles Anthony Richard Hoare. Communicating Sequential Processes. Published electronically: `usingcsp.com/-cspbook.pdf`, 2004. Second edition of [12]. See also `usingcsp.com/`.

A GLOSSARY

In every [construction] project it is important
to define all technical terms precisely,
and to adhere to these definitions.

(1) **Abstract:**
(2) **Access Authority:**
(3) **Action:**
(4) **Address:**
(5) **Affiliation:**
(6) **Attribute:**
(7) **Authority:**
(8) **Authorsip:**
(9) **Behaviour:**

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

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

(10) **Cartesian Part:**
(11) **Collection of Documents:**
(12) **Column of a Table:**
(13) **Communication:**
(14) **Compound Part:**
(15) **Concurrency:**
(16) **Content:**
(17) **Contents:**
(18) **Deleted Document:**
(19) **Document:**
(20) **Document Access Right:**
(21) **Document Domain:**
(22) **Document Figure:**
(23) **Document Handler:**
(24) **Document History:**
(25) **Document Image:**
(26) **Document Music:**
(27) **Document Permit:** – same as item 20.
(28) **Document Plain Text:**
(29) **Document Table:**
(30) **Document Text:**
(31) **Document Title:**
(32) **Document Video:**
(33) **Domain:**
(34) **Endurant:**
(35) **Endurant State:**
(36) **Entity:**
(37) **Event:**
(38) **Existing Document:**
(39) **External Quality:**
(40) **Figure:**
(41) **Glossary:**
(42) **Group of Handlers:**
(43) **Handler:**
(44) **Handler Group:**
(45) **Human:**
(46) **Identification:**
(47) **Image:**
(48) **Internal Quality:**
(49) **Living Species:**
(50) **Master Document:**
(51) **Mereology:**
(52) **Mereology of Document:**
(53) **Mereology of Document Collection:**
(54) **Mereology of Document Handler Domain:**
(55) **Mereology of Existing Document (Collection):**
(56) **Mereology of Group of Handlers:**
(57) **Mereology of Handler:**

(58) **Music:**
(59) **Part:**
(60) **Part Set:**
(61) **Perdurant:**
(62) **Phenomenon:**
(63) **Plain Text:**
(64) **Professional Title:**
(65) **Row of a Table:**
(66) **Solid Endurant:**
(67) **State:**
(68) **Table:**
(69) **Text:**
(70) **Time:** – TIME
(71) **Title:** There are two kinds of titles, see Items 31 on the facing page and 64.
(72) **Transcendence:**
(73) **Transcendental Deduction:**
(74) **Unique Identification:**
(75) **Unique Identifier State:**
(76) **Version:**
(77) **Version Content:**
(78) **Version Date:**
(79) **Version :**
(80) **Video:**
(81) **:**
(82) **:**
(83) **:**

B INDEXES

B.1 Document Domain Concepts

B.2 Doman Modelling Ontology

B.3 Formal Entities

mereo_GHM, 7
mereo_H, 7
uid_CD, 6
uid_DD, 6
uid_DH, 6
uid_D, 6
uid_ED, 6
uid_GH, 6
uid_HS, 6
uid_H, 6
uid_P, 6
obs_CD, 5
obs_DD, 5
obs_ED, 5
obs_GH, 5
obs_HS, 5

There are 76 formal RSL entities.

Contents