Business Object Reference Ontology

BORO Analysis Tools

C-FORS Summer School in Foundational Ontology
(C-FORS 2025)

23 May 2025, University of Oslo, Norway

Chris Partridge
Andrew Mitchell
Oscar Xiberta Soto
Diego Alberto Zabala Betancur
Jonathan Eyre

Schedule

Morning Sessions		
	9:00 - 9:05	Session 0 – Introduction
	9:05 - 9:45	Session 1 – Context
	10:00 - 10:45	Session 2 – BORO Ontology
	11:00 - 12:00	**Session 3 – Analysis Tools**
Afternoon Sessions		
	1:15 - 3:30	Session 1 – Practical Examples
	3:30 - 5:00	Session 2 – Examples Discussion / Presentation

Business Object Reference Ontology

Structure

- Overview
- LOAD: Structured Data Table Migration
  - BORO Top Level Tables
- EVOLVE
  - Support Tools
    - Space Time Maps
    - Ontological Euler Diagrams
    - BORO UML
    - BORO eXcel Table (Manual) Pipeline
  - BORO KNIME Data Pipeline
- Project planning

Overview

Overall goal

- To provide some ‘practice’ with ontologization interoperability pipelines
  - by recreating the first stage of an ontologization interoperability pipeline
    - With (simple) examples
  - directly experience some of the challenges (in an attenuated form)
- Probably significantly more ‘practice’ needed before one becomes expert

Two practical problems

Practical problems
lump of clay
ISO Countries

The focus this afternoon will be on the first problem.
If you manage to finish the first problem, the second problem is available.

Six BORO Summer School tools

Abbreviation	Name	Stage
SDTM	Structured Data Table Migration	LOAD
STM	Space Time Maps	EVOLVE
OED	Ontological Euler Diagrams	EVOLVE
BUML	BORO UML	EVOLVE
BXTP	BORO eXcel Table (Manual) Pipeline	EVOLVE
BKnDP	BORO KNIME Data Pipeline	EVOLVE

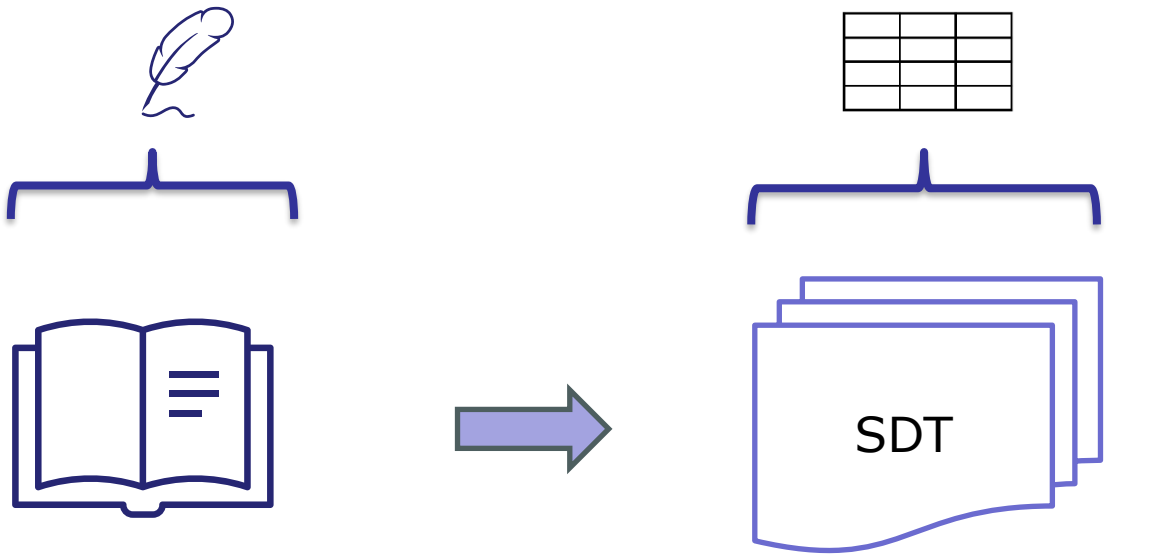
NB: This is a list developed for the Summer School. There is a large range of evolving tools, which we have selected from.
Note also, our default language/tool is Python not KNIME. KNIME has been selected for its no-code credentials – making it more appropriate here.

LOAD: Structured Data Table Migration

Principle: shift (data) structure left

- A goal of the bCLEARer pipeline process is to shift left, as far a possible, the move to structured data
  - enable us to work with machines talking to machines as soon as possible
- If one starts with unstructured text
  - one needs to structure it
  - with the minimum intervention
  - preserving as much of the explicit structure as possible
- A good first step is a shift to simple tables
  - the simplicity of their structure helps to keep the structuring clear
- One goal of Load is to ensure one is working with structured data

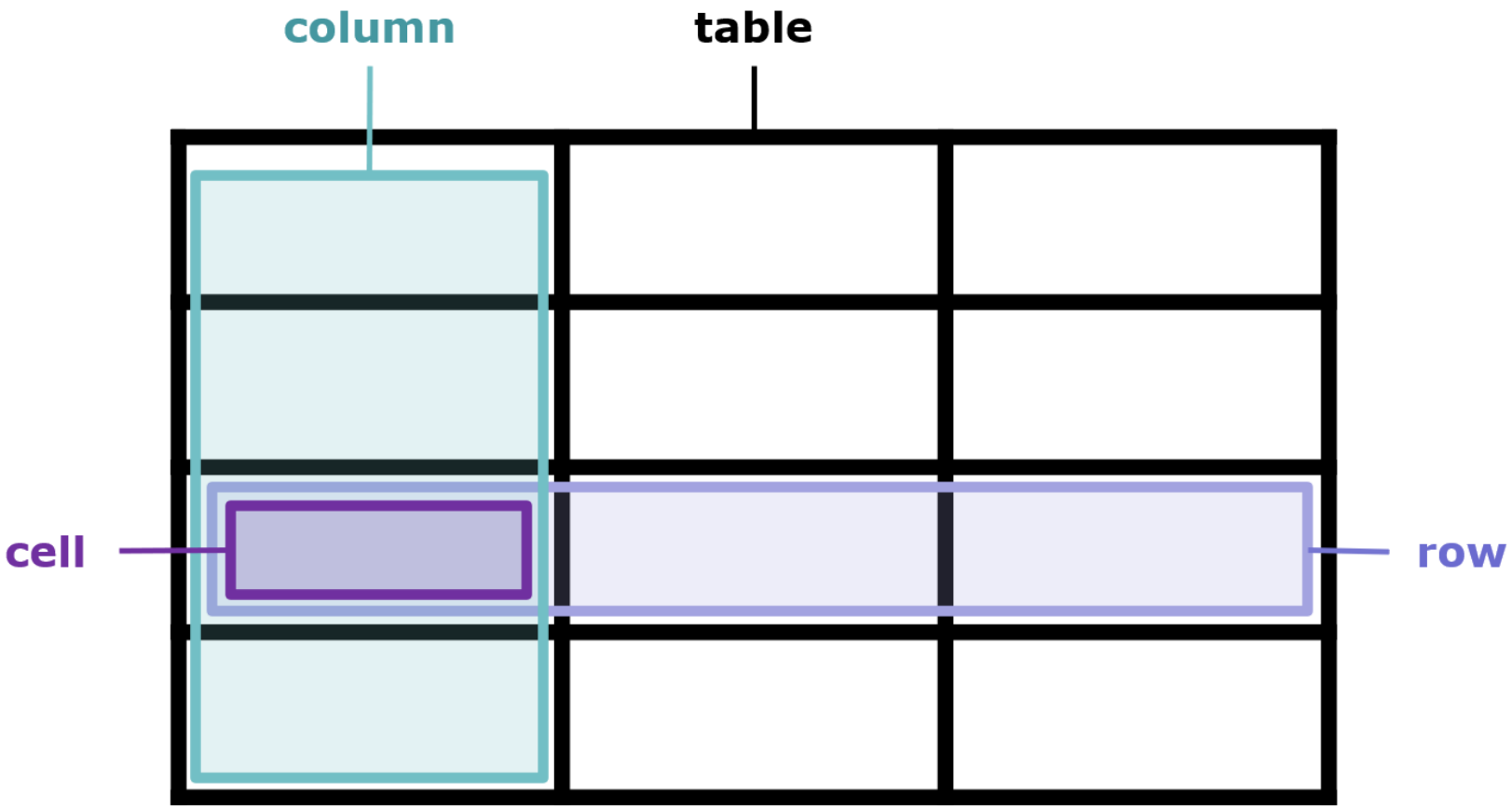
Structured data table migration



unstructured text

structured data table

Base data table structure

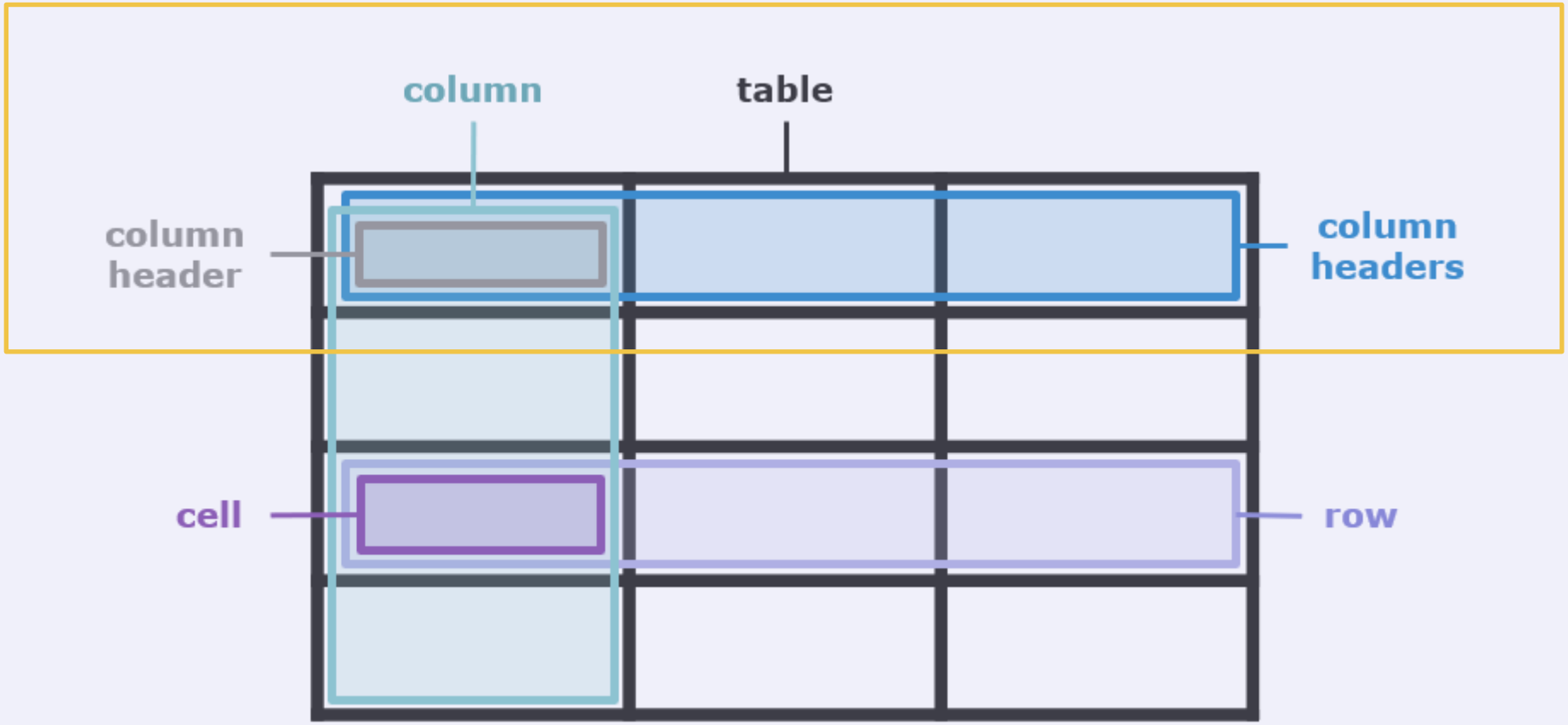


Base table composition structure

base table composition structure

data type	structure type	table structure element	description
data item	table	row	composed of cells organised by column
data column	table	column	composed of cells organised by row - has a single column heading cell.
data cell	table	cell	the intersection of columns and rows - has content

Extended data table structure



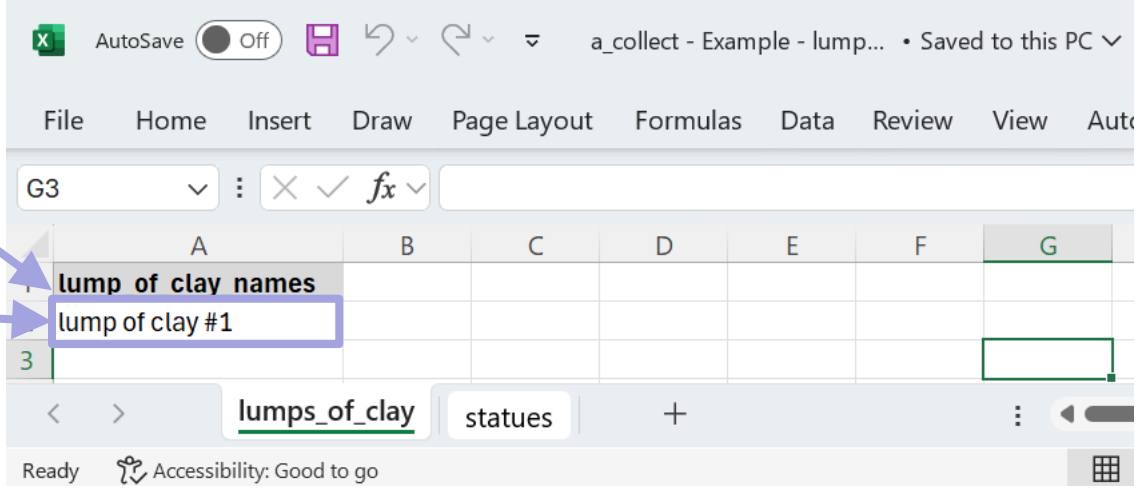
Lump of clay - starting point

- We start with unstructured text:
  - “There is a lump of clay that, at time t1, is used to make an aesthetically valuable statue. At time t2, the statue is destroyed. At time t3, the same lump of clay is reshaped to make a different statue. This statue is aesthetically valuable too.”

- NOTE: We do this here
  - and provide you with the results for the practical example.

Lump of clay – migration to table 1

"There is a lump of clay that,
at time t1, is used to make an
aesthetically valuable statue.
At time t2,
the statue is destroyed.
At time t3,
the same lump of clay is
reshaped to make a
different statue.
This statue is aesthetically
valuable too."



Lump of clay – migration to table 2

"There is a lump of clay that, at time t1, is used to make an aesthetically valuable statue.

At time t2,

the statue is destroyed.

At time t3,

the same lump of clay is reshaped to make a

different statue.

This statue is aesthetically valuable too."

F2

	A	B	C	D	E
1	statue_names	aesthetically valuable	made from	made on	destroyed on
2	statue #1	Yes	lump of clay #1	t1	t2
3	statue #2	Yes	lump of clay #1	t3	
4					

lumps_of_clay statues

Lump of clay – migration to table 3

"There is a lump of clay that, at time t1, is used to make an aesthetically valuable statue.

At time t2,
the statue is destroyed.

At time t3,
the same lump of clay is reshaped to make a different statue.

This statue is aesthetically valuable too."

F2

	A	B	C	D	E
1	statue_names	aesthetically valuable	made from	made on	destroyed on
2	statue #1	Yes	lump of clay #1	t1	t2
3	statue #2	Yes	lump of clay #1	t3	
4					

lumps_of_clay statues

Lump of clay – migration to table 4

"There is a lump of clay that, at time t1, is used to make an aesthetically valuable statue.

At time t2,

the statue is destroyed.

At time t3,

the same lump of clay is reshaped to make a different statue.

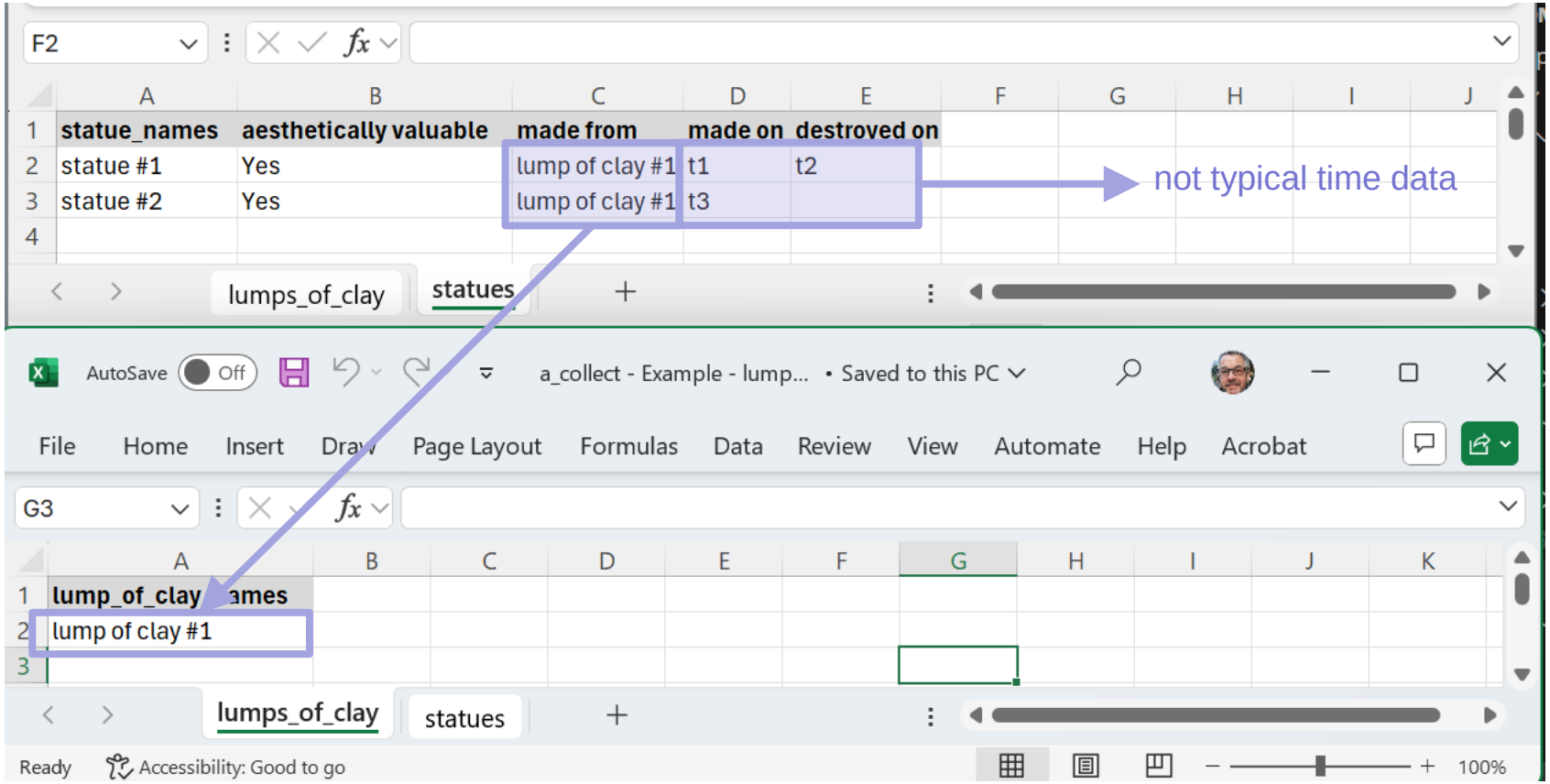
This statue is aesthetically valuable too."

F2

	A	B	C	D	E
1	statue_names	aesthetically valuable	made from	made on	destroyed on
2	statue #1	Yes	lump of clay #1	t1	t2
3	statue #2	Yes	lump of clay #1	t3	
4					

lumps_of_clay | statues

Lump of clay - structured data tables – proto time



a_collect - Example - lump of clay - stage 1 - proto time.xlsx

Lump of clay - structured data tables – dataise time

F2

	A	B	C	D	E
1	statue_names	aesthetically valuable	made from	made on	destroyed on
2	statue #1	Yes	lump of clay #1	t1	t2
3	statue #2	Yes	lump of clay #1	t3	
4					

lumps_of_clay | statues

a_collect - Example - lump of clay - stage 1 - proto time.xlsx

	A
1	lump_of_clay_names
2	lump of clay #1
3	

lumps_of_clay | statues | times

Ready Accessibility: Good to go

A1 statue_names

	A	B	C	D	E
1	statue_names	aesthetically valuable	made from	made on	destroyed on
2	statue #1	Yes	lump of clay #1	2023-06-20	2023-10-24
3	statue #2	Yes	lump of clay #1	2024-07-06	

lumps_of_clay | statues | times

Ready Accessibility: Good to go

A1 time_names

	A	B
1	time_names	YYYY-MM-DD
2	t1	2023-06-20
3	t2	2023-10-24
4	t3	2024-07-06
5		

lumps_of_clay | statues | times

Ready Accessibility: Good to go

NB: added domain knowledge about time granularity

a_collect - Example - lump of clay - stage 2 - dataise time.xlsx

ISO Countries

Already structured data (table)

	A	B	C	D	E	F
1	**English short name**	**French short name**	**Alpha-2 code**	**Alpha-3 code**	**Numeric**	
2	Afghanistan	Afghanistan (l')	AF	AFG	4	
3	Albania	Albanie (l')	AL	ALB	8	
4	Algeria	Algérie (l')	DZ	DZA	12	
5	American Samoa	Samoa américaines (les)	AS	ASM	16	
6	Andorra	Andorre (l')	AD	AND	20	
7	Angola	Angola (l')	AO	AGO	24	
8	Anguilla	Anguilla	AI	AIA	660	
9	Antarctica	Antarctique (l')	AQ	ATA	10	
10	Antigua and Barbuda	Antigua-et-Barbuda	AG	ATG	28	
11	Argentina	Argentine (l')	AR	ARG	32	
12	Armenia	Arménie (l')	AM	ARM	51	
13	Aruba	Aruba	AW	ABW	533	
14	Australia	Australie (l')	AU	AUS	36	
15	Austria	Autriche (l')	AT	AUT	40	
16	Azerbaijan	Azerbaïdjan (l')	AZ	AZE	31	
17	Bahamas (the)	Bahamas (les)	BS	BHS	44	

with formatting | without formatting

a_collect - Example - ISO 3166 - Part 1 -- Country code.xlsx

Business Object Reference Ontology

BORO Top Level Tables

LOAD: Structured Data Table Migration

BORO top level tables

- The top level needs to be introduced to the pipeline
- Various ways of doing this
- Here, we add it in directly at LOAD
  - this is done for you in the starter pack
- We add a simplified table version
  - suitable for the worked examples

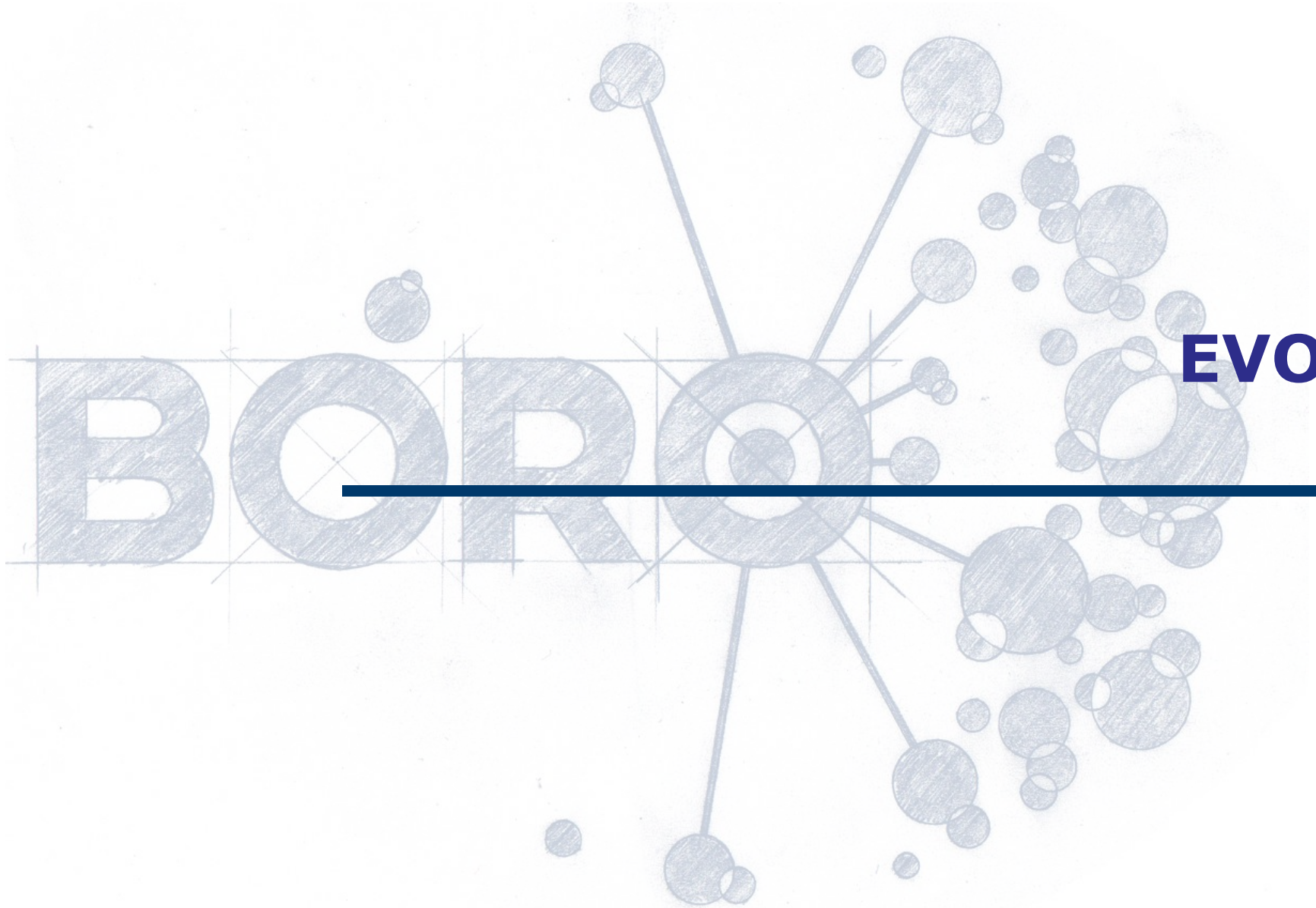
BORO top level tables - pure

table_names	column_1_names	column_2_names	column_3_names	column_4_names
Sets	bie_ids	names		
Individuals	bie_ids	names		
Tuples	bie_ids			
elements-sets	bie_ids	element_bie_ids	set_bie_ids	
parts-wholes	bie_ids	part_bie_ids	whole_bie_ids	
sub-super-sets	bie_ids	sub_set_bie_ids	super_set_bie_ids	
tuple-places	bie_ids	placing_bie_ids	placed_bie_ids	place_numbers

BORO top level tables - pragmatic

table_names	column_1_names	column_2_names	column_3_names	column_4_names
Sets	bie_ids	names		
Individuals	bie_ids	names		
Tuples	bie_ids	place1_bie_ids	place2_bie_ids	place3_bie_ids
grounding-relation-places	bie_ids	relation-place-type_names	placing_bie_ids	placed_bie_ids
elements-sets	bie_ids	element_bie_ids	set_bie_ids	
parts-wholes	bie_ids	part_bie_ids	whole_bie_ids	
sub-super-sets	bie_ids	sub_set_bie_ids	super_set_bie_ids	

Business Object Reference Ontology

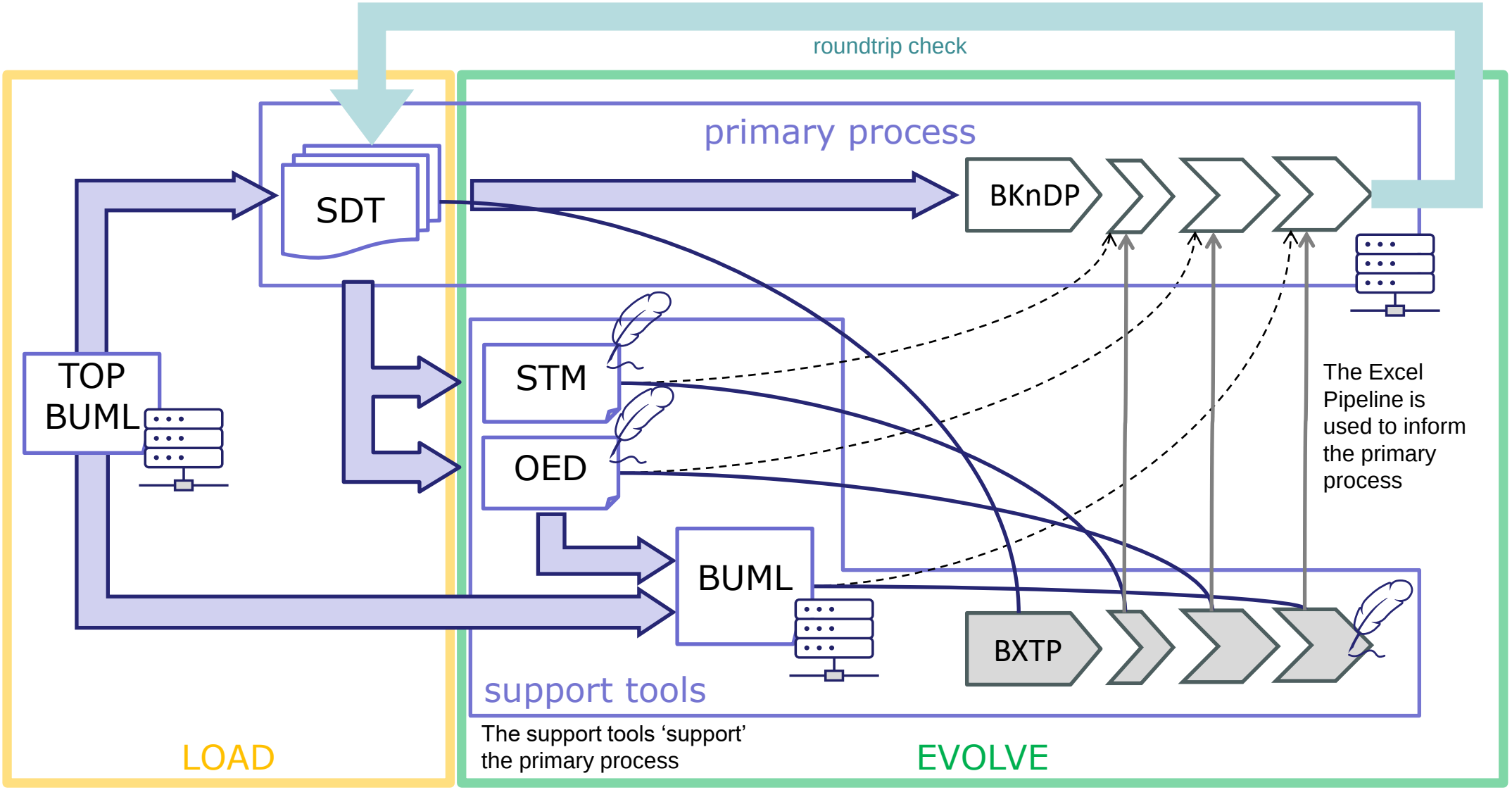


EVOLVE

Broad outline

- Same process stages for both practical problems
- Broadly,
  - there is
    - a support process that ‘designs’ the pipeline
      - this has human aspects
    - a primary process that IS the pipeline
      - this is a ‘pure’ machine process
        - the process is executable, repeatable and inspectable
- Hopefully,
  - building the primary process will help you experience the practice on ontologising a pipeline

EVOLVE - practical problem process



Support Tools

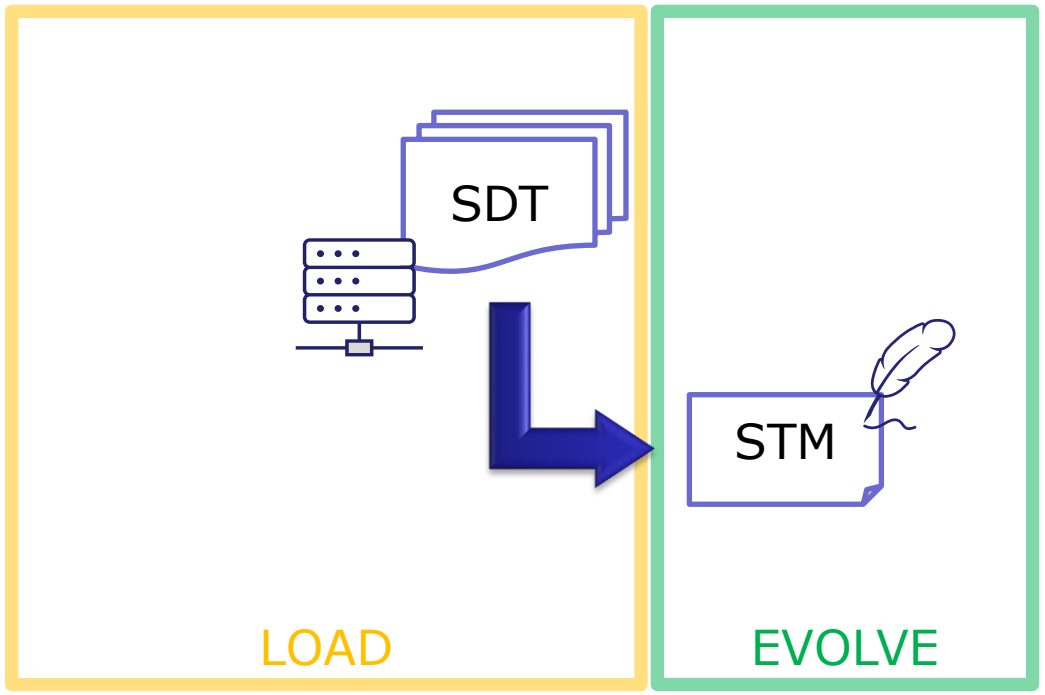
Four BORO Summer School support tools

Abbreviation	Name
STM	Space Time Maps
OED	Ontological Euler Diagrams
BUML	BORO UML
BXTP	BORO eXcel Table (Manual) Pipeline

These Summer School ‘tools’ are part of a much wider, richer toolkit
They are useful pedagogical devices
As often, with practices, once one gains competence with these, one deploys them in innovative radically different ways

Space Time Maps

Practical problem process



STM Segment

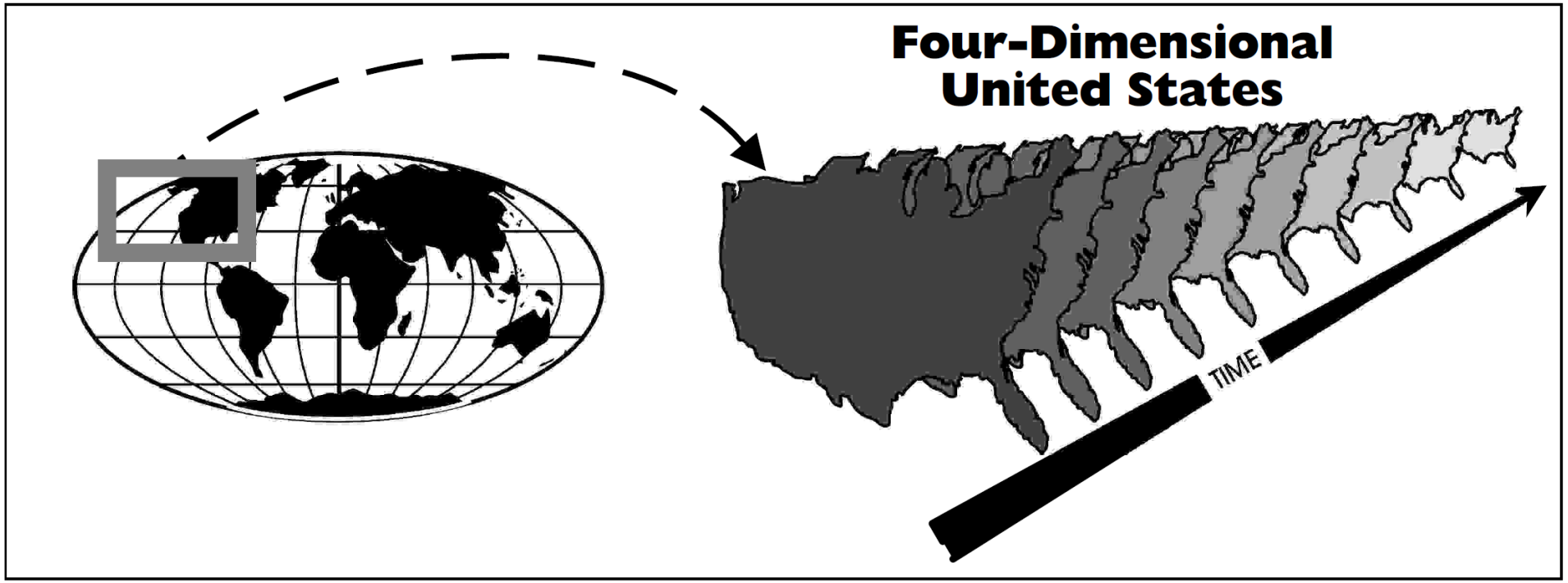
- STMs Focus on Individuals
  - other BORO objects are constructed from (grounded in) Individuals
- STMs are an intuitive way of visualising the mereology of Individuals
  - in particular the overlap and part hood relations
- As Individuals (in BORO) have a mereological extensional criteria of identity
  - the ‘extensional’ two-dimensional surface of a page (or screen) has sufficiently similar characteristics to represent ‘directly’ (visually) the mereology
  - humans find this a useful tool

Purpose of STMs

- STMs are typically used as a first stage of analysis, where in the second stage these are translated into a BORO model
- STMs provide an easy way to visualise 4D elements in a domain
  - and as such are a useful way to start understanding them
- The STM visualisation aims to abstract the individuals to their mereological essentials
- Too time consuming to draw for the whole domain
  - so, collect representative examples of important ‘typical’ mereological relations
  - these are candidates for STMs

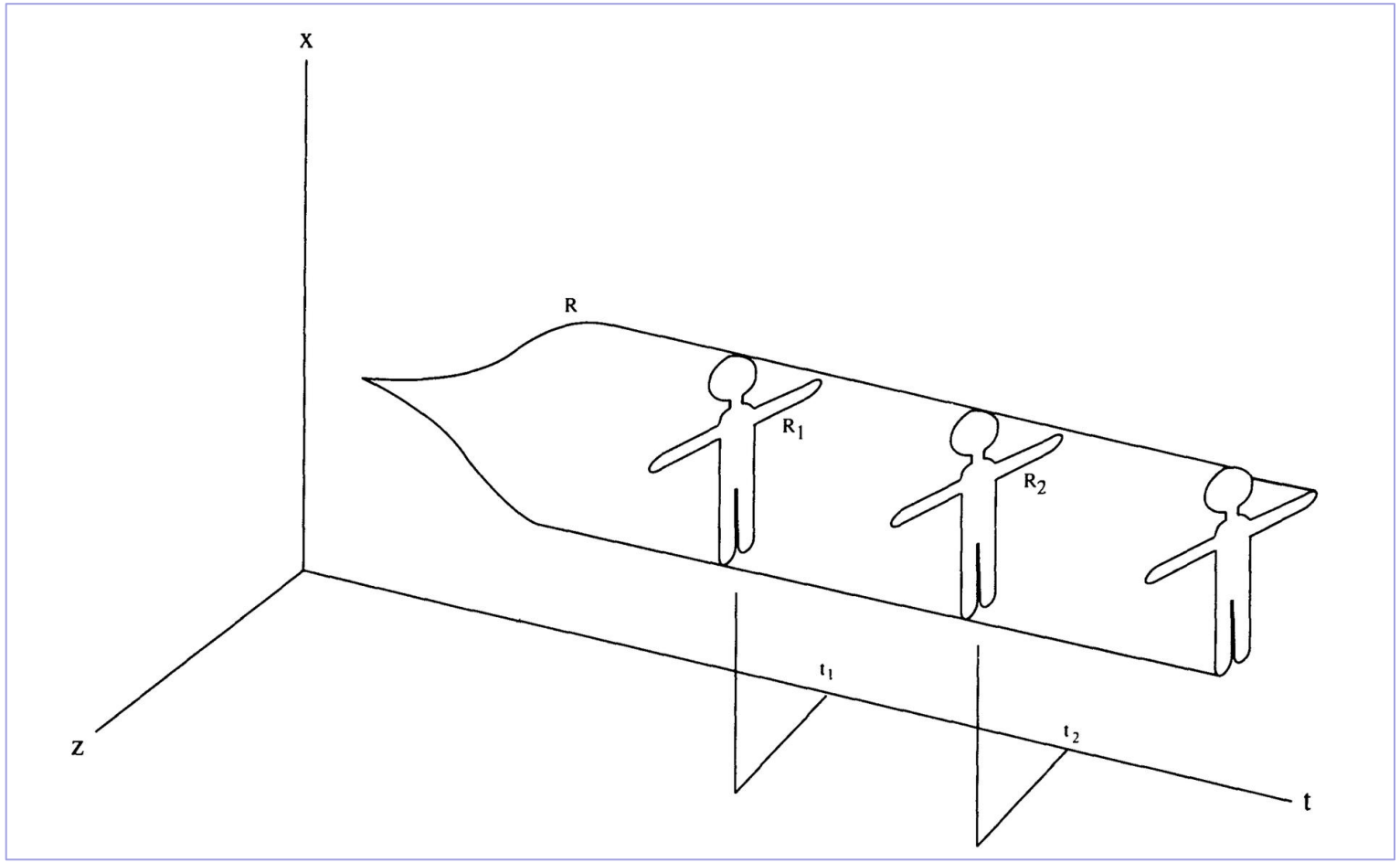
Individuals in BORO

- BORO is an extensional ontology
  - the criterion of identity for Individuals is mereological (spatio-temporal) extension
  - or more colloquially four-dimensional (4D)



Van Inwagen's space-time diagrams

Not that usual, but STMs appear in various places.
In this case, not quite the same, less focus on exact mereology.



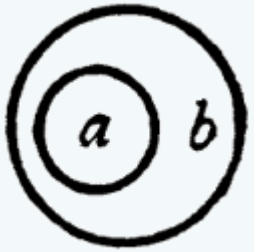
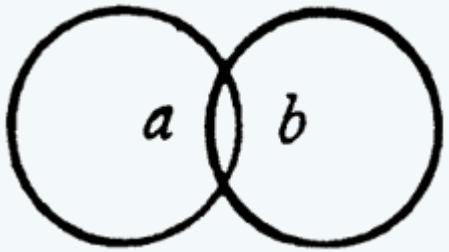
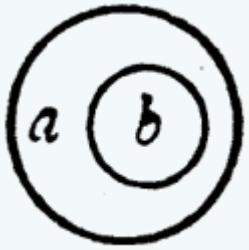
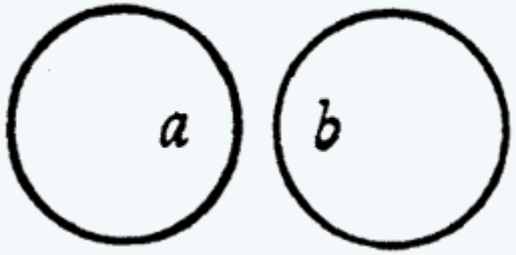
Van Inwagen, P. (1990). Four-Dimensional Objects. Noûs, 24(2), 245.

Three components of an STM Individual

A region representing a spatio-temporal extent	A border around the region – the *contour* – marking the region as representing an Individual	A string of letters labelling the Individual
		A
		A

The Gergonne five possible extensional relations

- The Gergonne relations give the five possible ways in which two objects can simply share extension

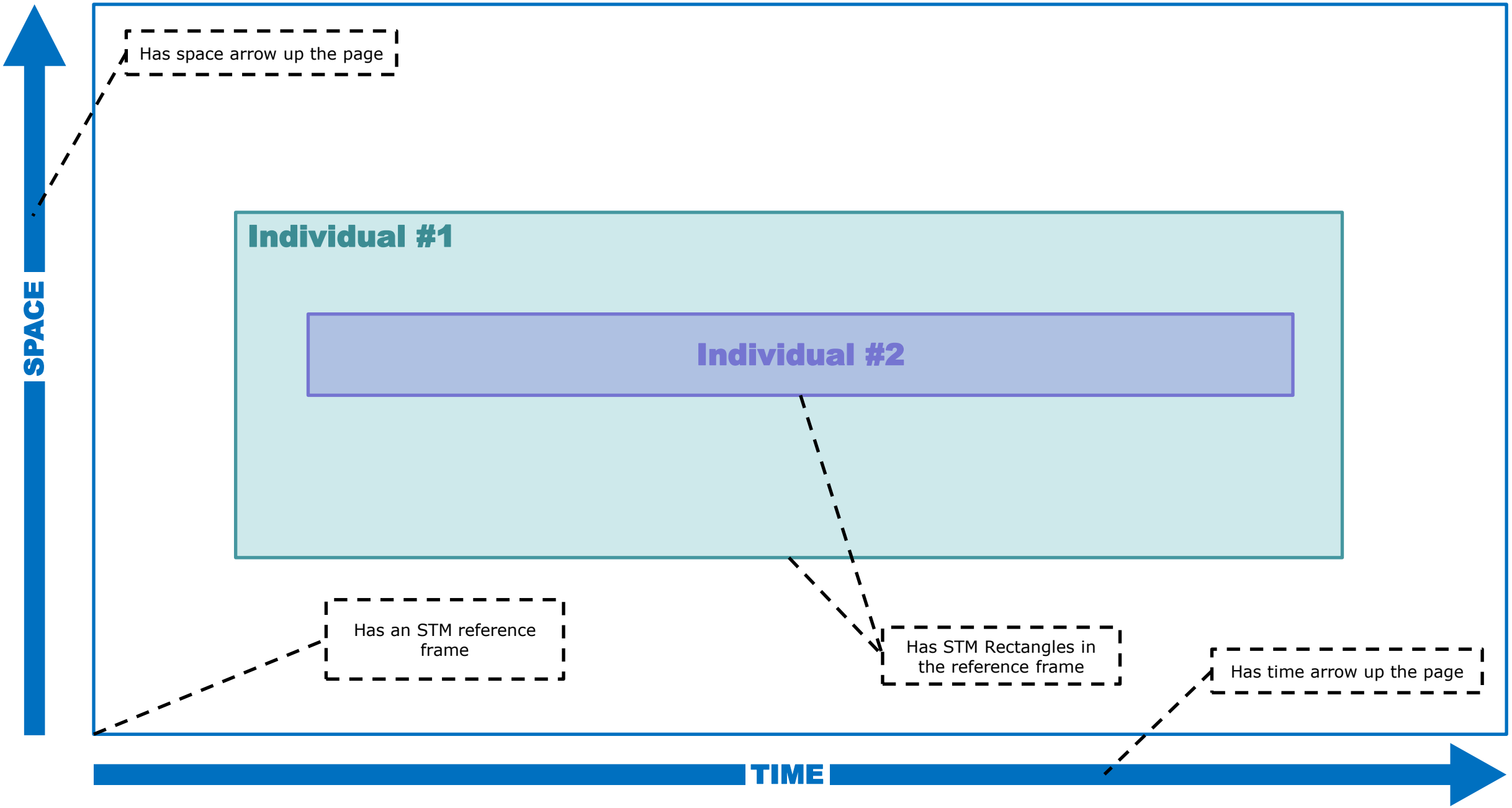
coincident	contained	overlap	contains	disjoint
"if and only if every a is a b and every b is an a"	"if and only if every a is a b and not every b is an a"	"if and only if it is not the case that either every a is a b or every b is an a or no a is a b"	"if and only if every b is an a and not every a is a b"	"if and only if no a is a b"
				

Based upon: J. A. Faris - The Gergonne Relations - The Journal of Symbolic Logic, Vol. 20, No. 3 (Sep., 1955), pp. 207-231
based upon: J. D. Gergonne, Essai de dialectique rationelle, Annales des mathematiques pures et appliquees, Vol. 7, (1817).

How to recognise STMs

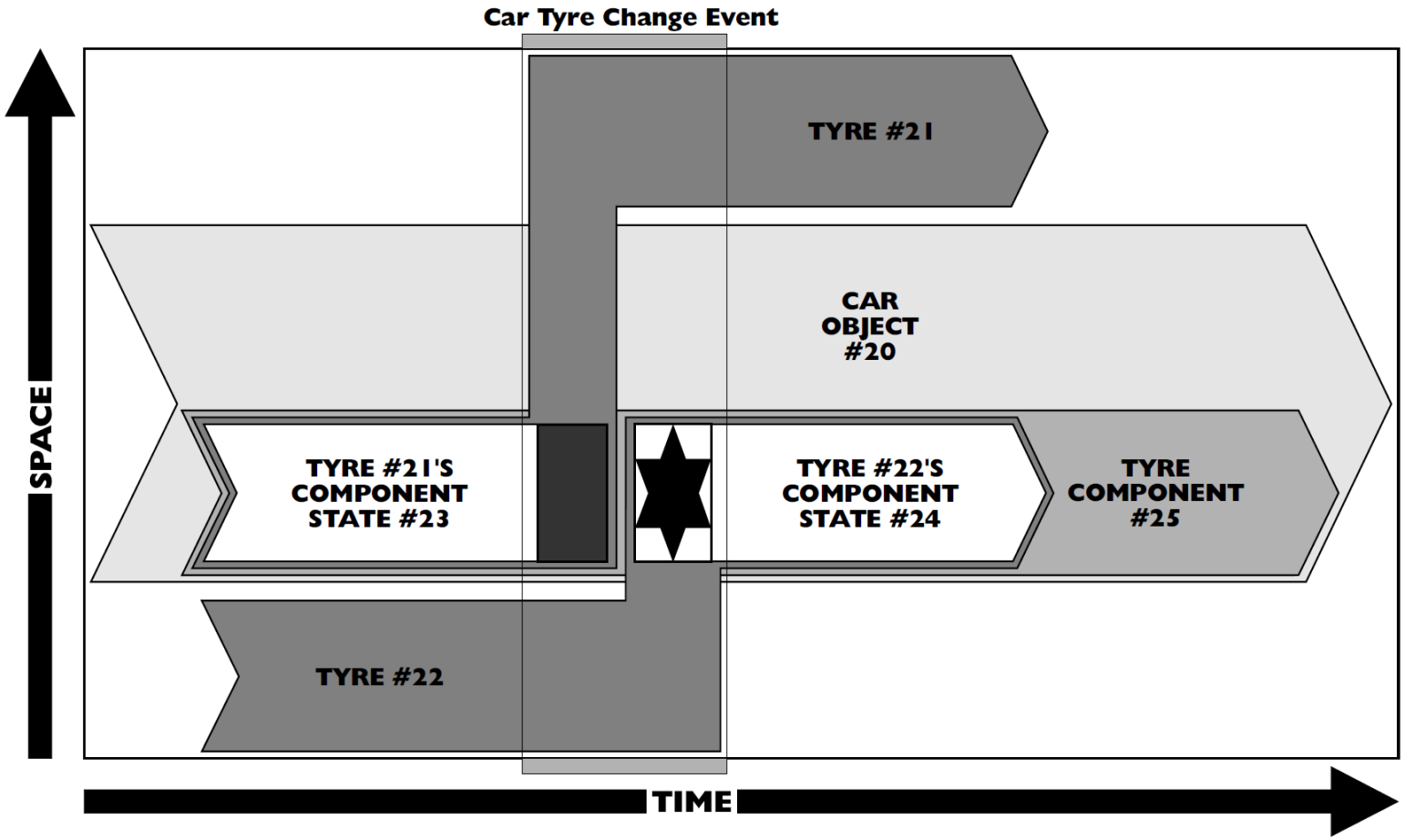
- Clues to look for
  - space axis along the left side of the page
  - time axis across the bottom of the page
  - reference frame contained within the axes
  - Individual rectangles (shapes) in the reference frame

Lore ipsum


SPACE
Has space arrow up the page
Individual #1
Individual #2
Has an STM reference frame
Has STM Rectangles in the reference frame
Has time arrow up the page
TIME

Patterns: overlap

Figure 8.14:
Car tyre change space-time map



From *Business Objects: Re-engineering for Reuse*

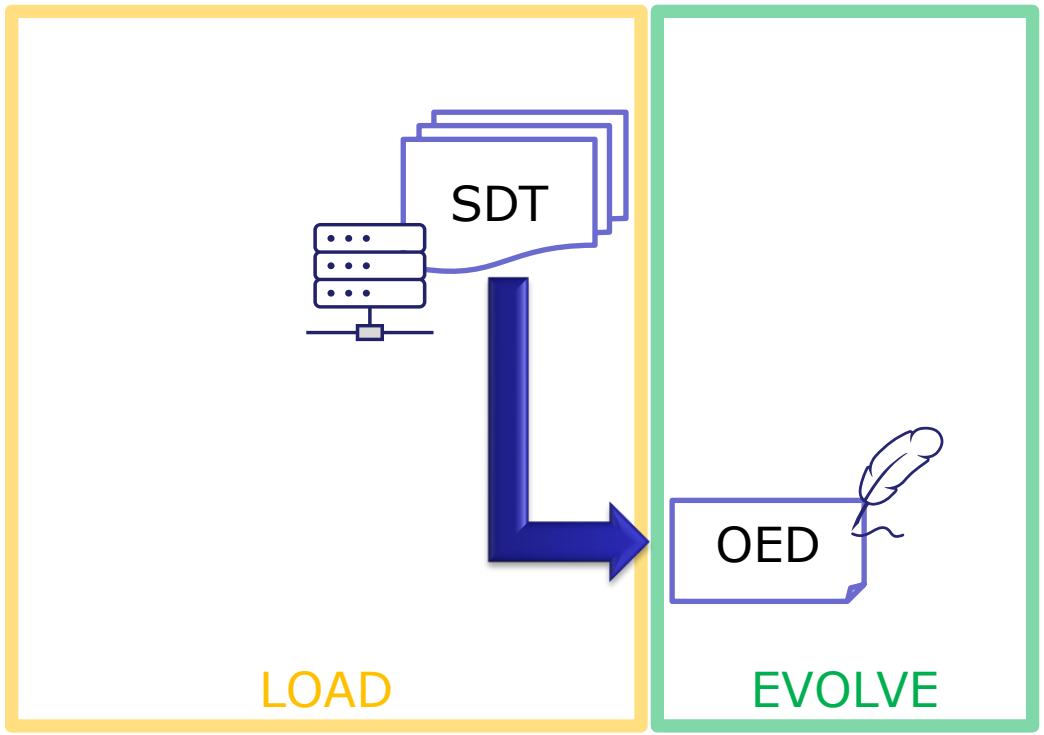
How to read STMs

- The 2D reference frame picks out the relevant fragment of the 4D domain
- The domain's objects are shown as boxes
  - a box's boundary represents the Individual's real boundary
  - boxes' containment, overlap, etc. show the mereological relationships between Individuals
  - a box's name helps to further identify the Individual

Business Object Reference Ontology

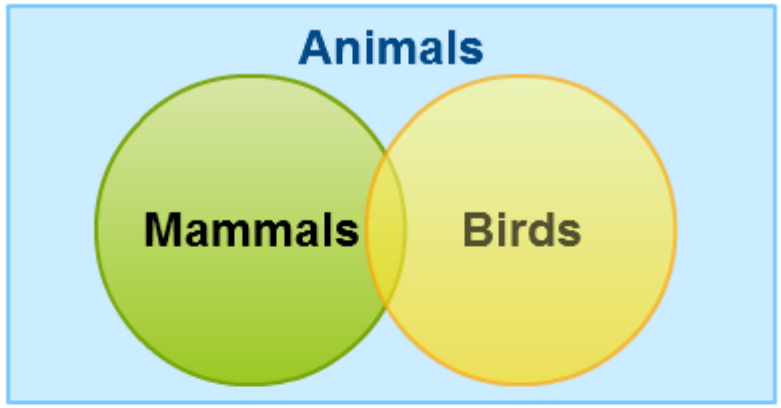
Ontological Euler Diagrams

Practical problem process

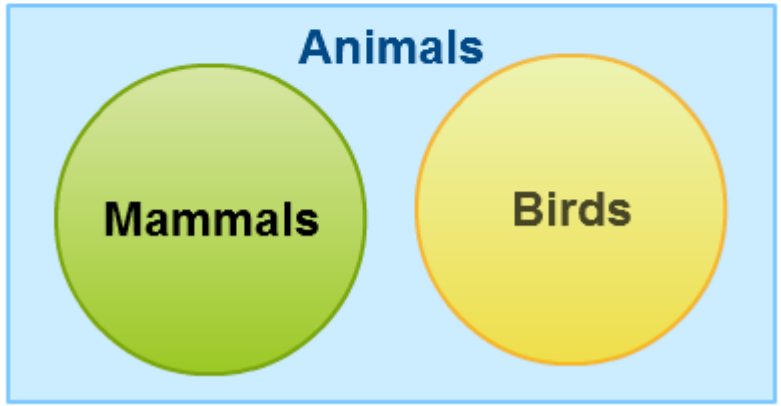


Euler not Venn diagrams

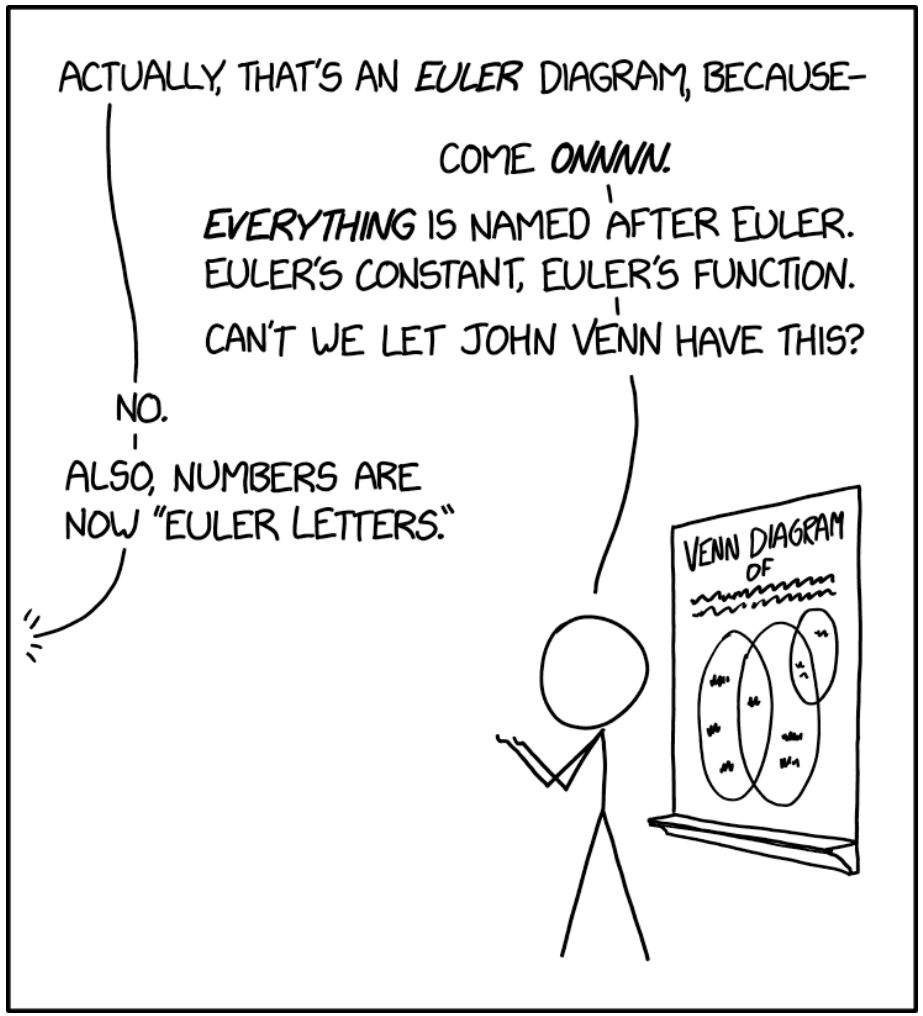
VENN



EULER



A Venn diagrams show all combinations (even impossible ones).



https://xkcd.com/2721/

Types in a BORO foundational ontology

- Sets' identity relations
  - from an ontological perspective, these will exist simpliciter
  - from an epistemic perspective, we may or may not know whether these exist
  - an ontological picture should not contain epistemological elements
  - to handle this issue, for the topic of the Euler diagram
    - we make a set identity relation omniscience assumption
      - in other words, all the identity relations between sets are known

- With this assumption, it makes sense to adopt
  - the 'Existential Import Convention'
    - where adding a region to the Euler diagram implies that it exists
      - in other words, it has members

Sets in a BORO foundational ontology

- In BOROs' extensional ontology
  - given a Set's identity is grounded in its members
  - then the way in which two Sets' identity can be related are broadly
    - with the same members are identical
    - with different members are not identical
    - that share members are partially identical
    - with no members in common are disjoint

Three basic extensional set patterns – for two sets

- There are only three ways two sets can be ‘identity’ related

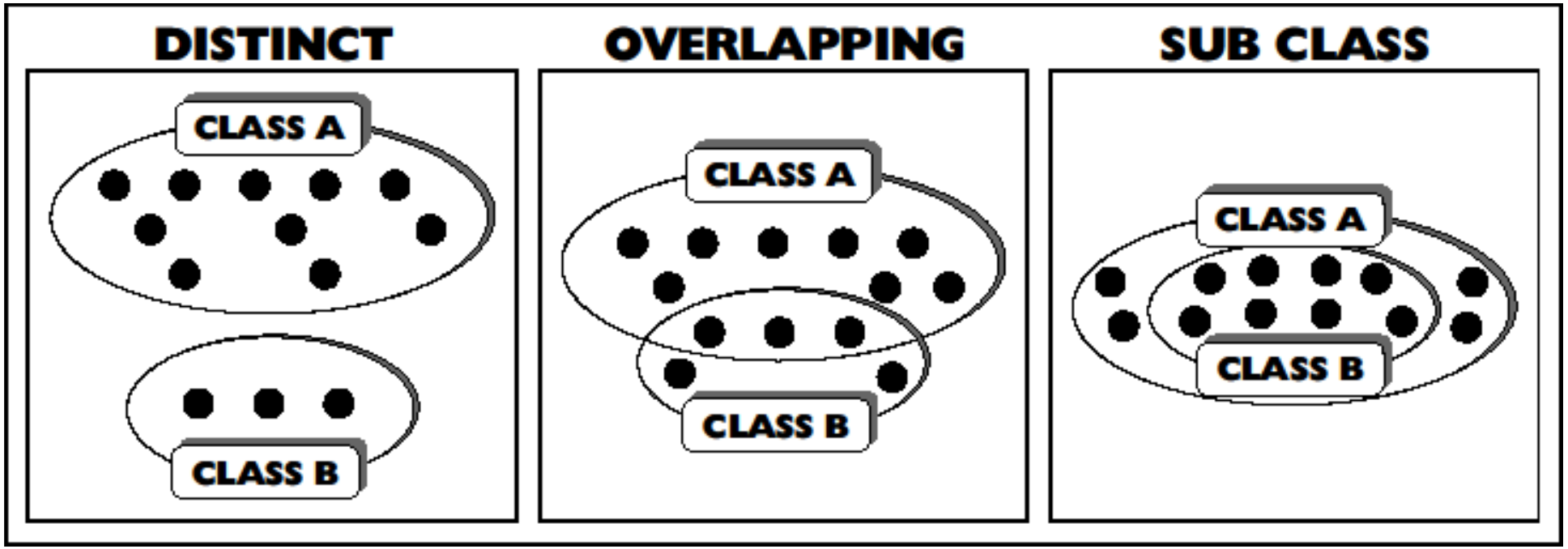


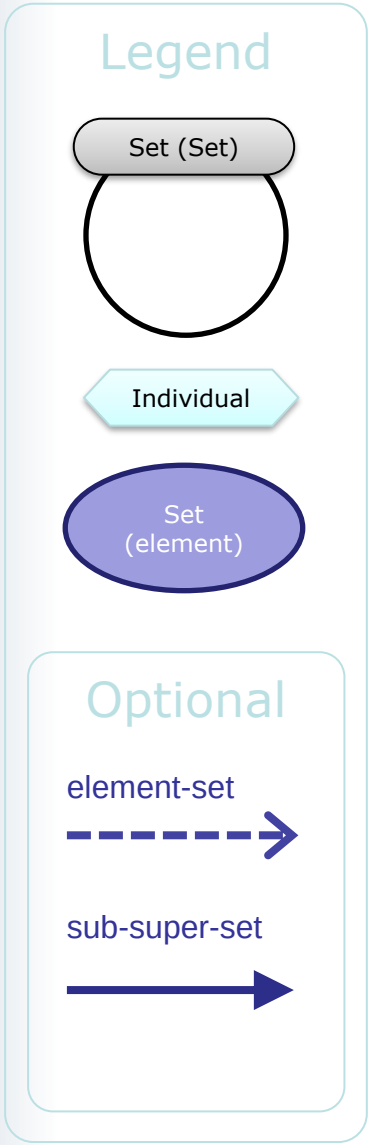
Figure 10.15 - Pattern for classes

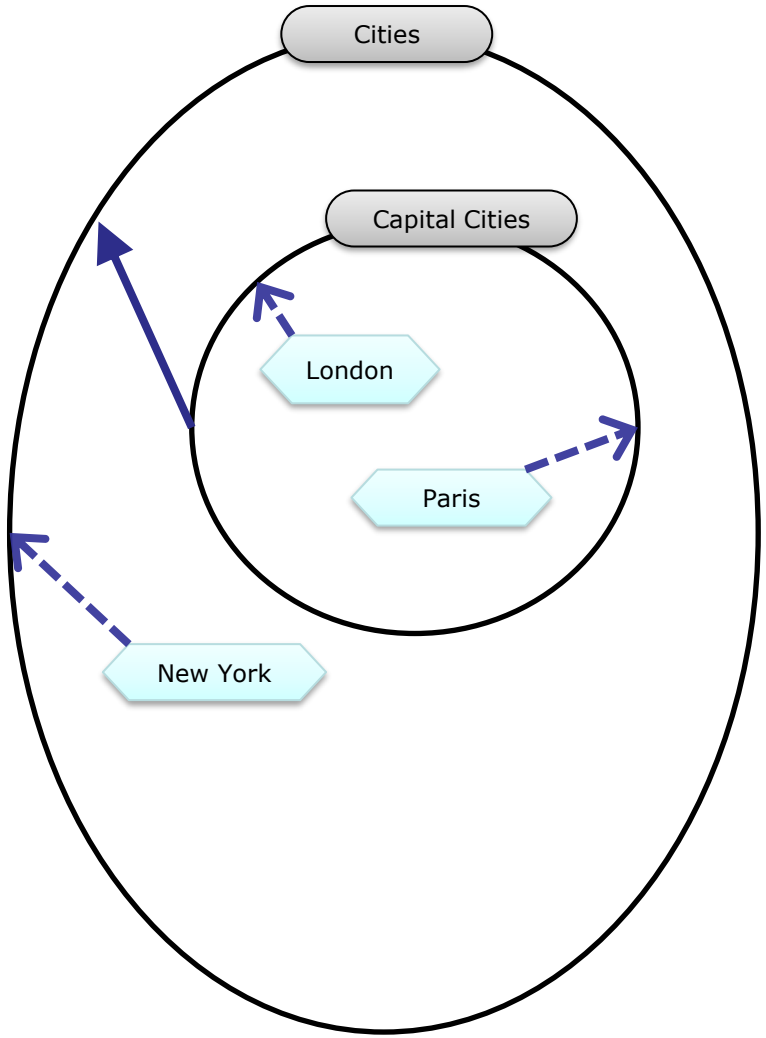
From *Business Objects: Re-engineering for Reuse*

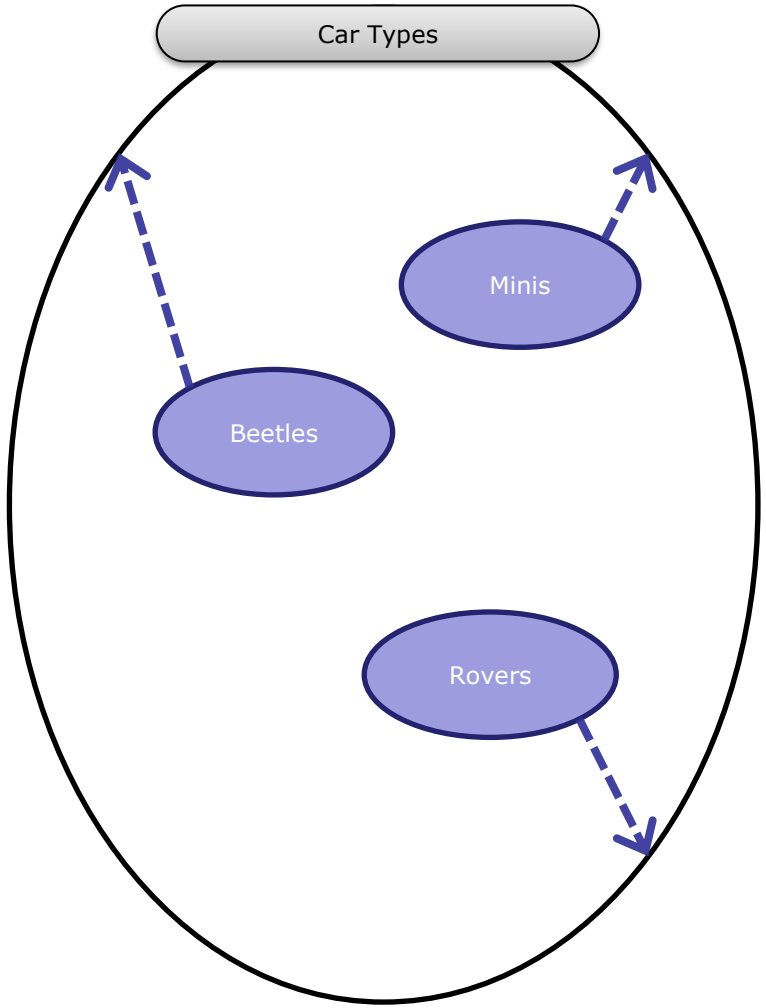
Three components of an Euler set

A region representing a collection of objects	A circle around the region marking the region as representing a type – the *contour*	A string of letters labelling the type
		A

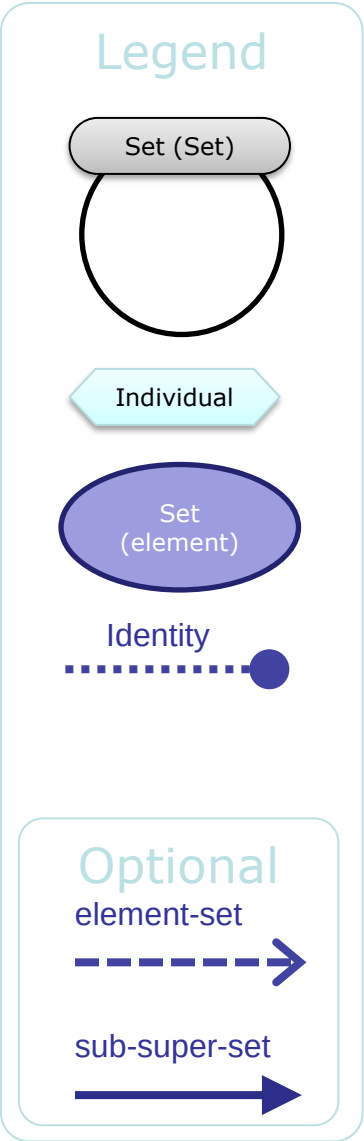
Example simple order type OEDs

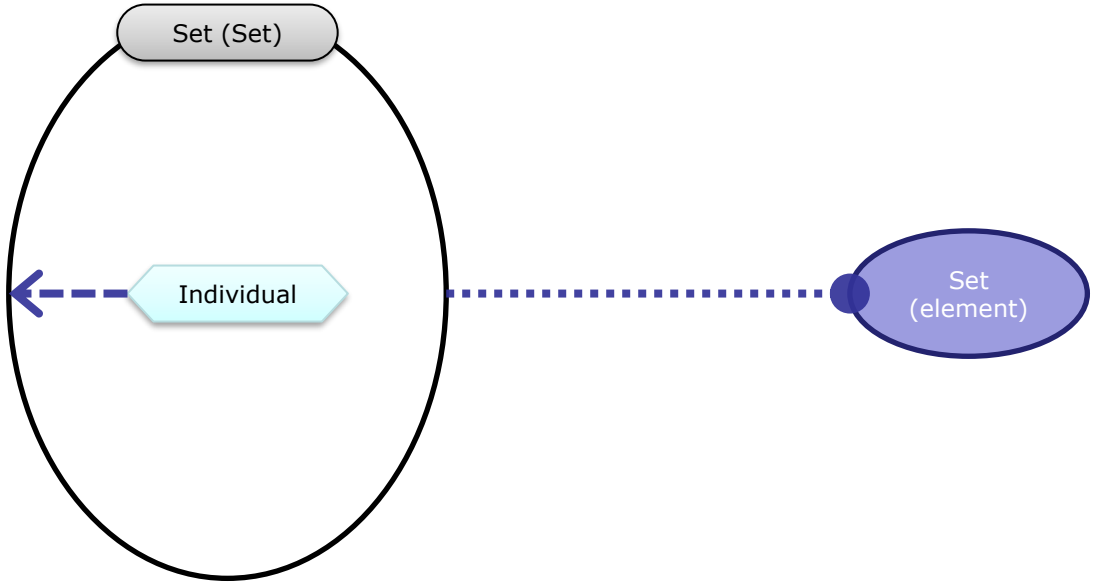




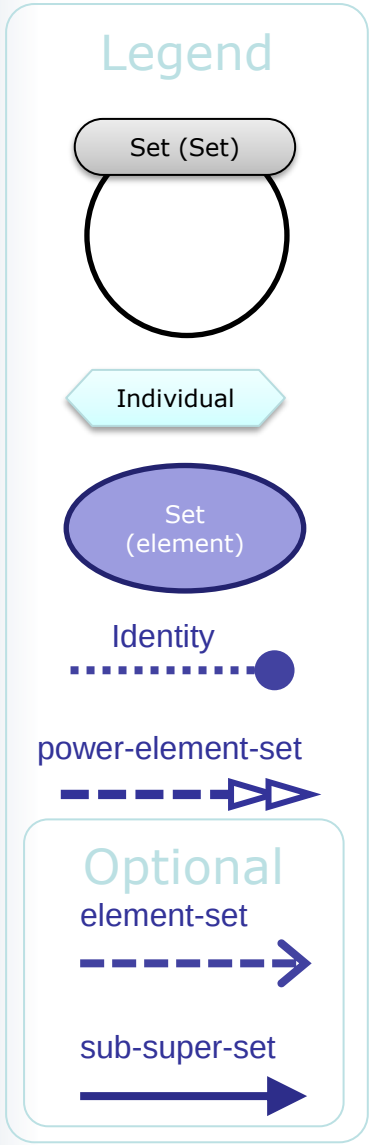


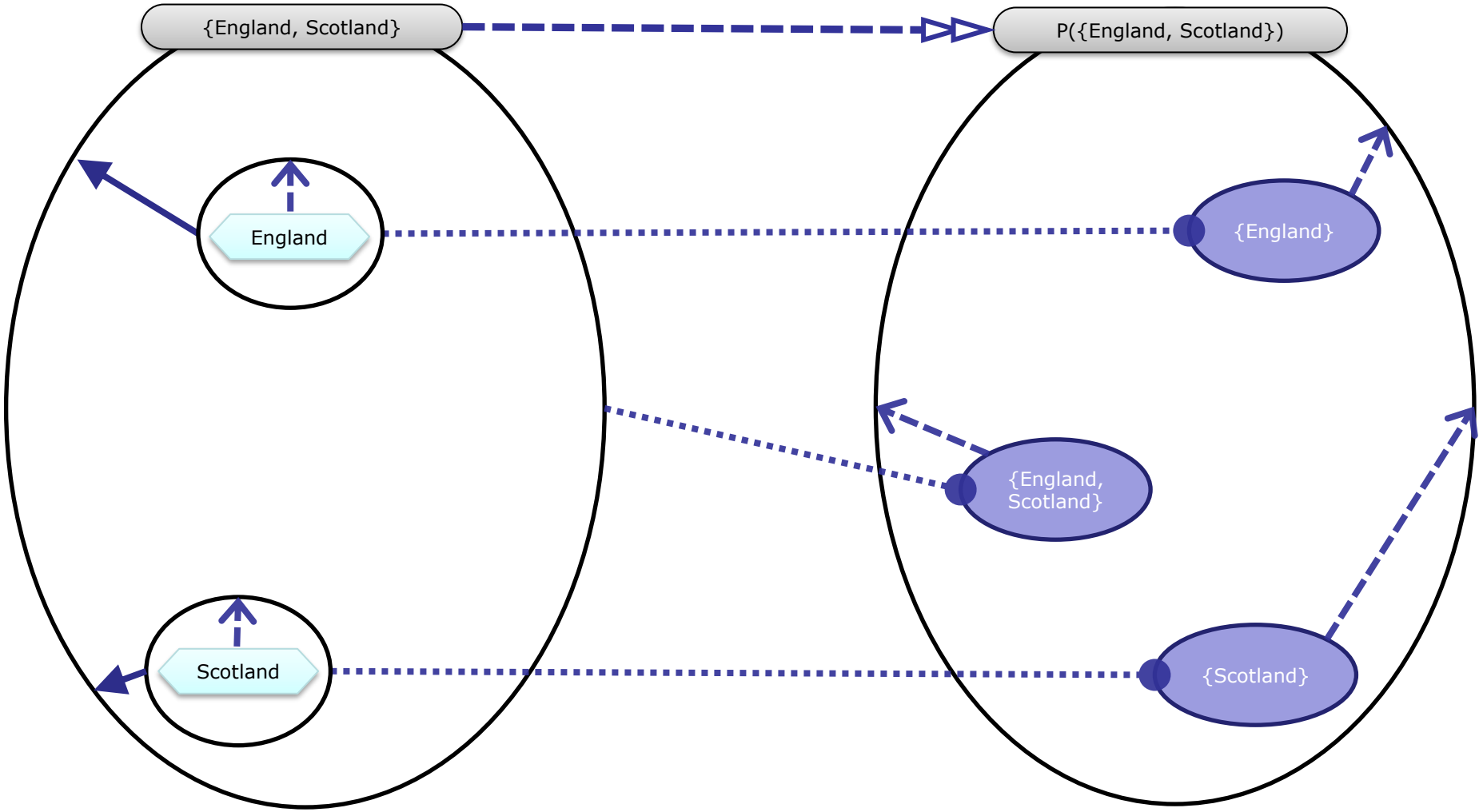
Type ascending



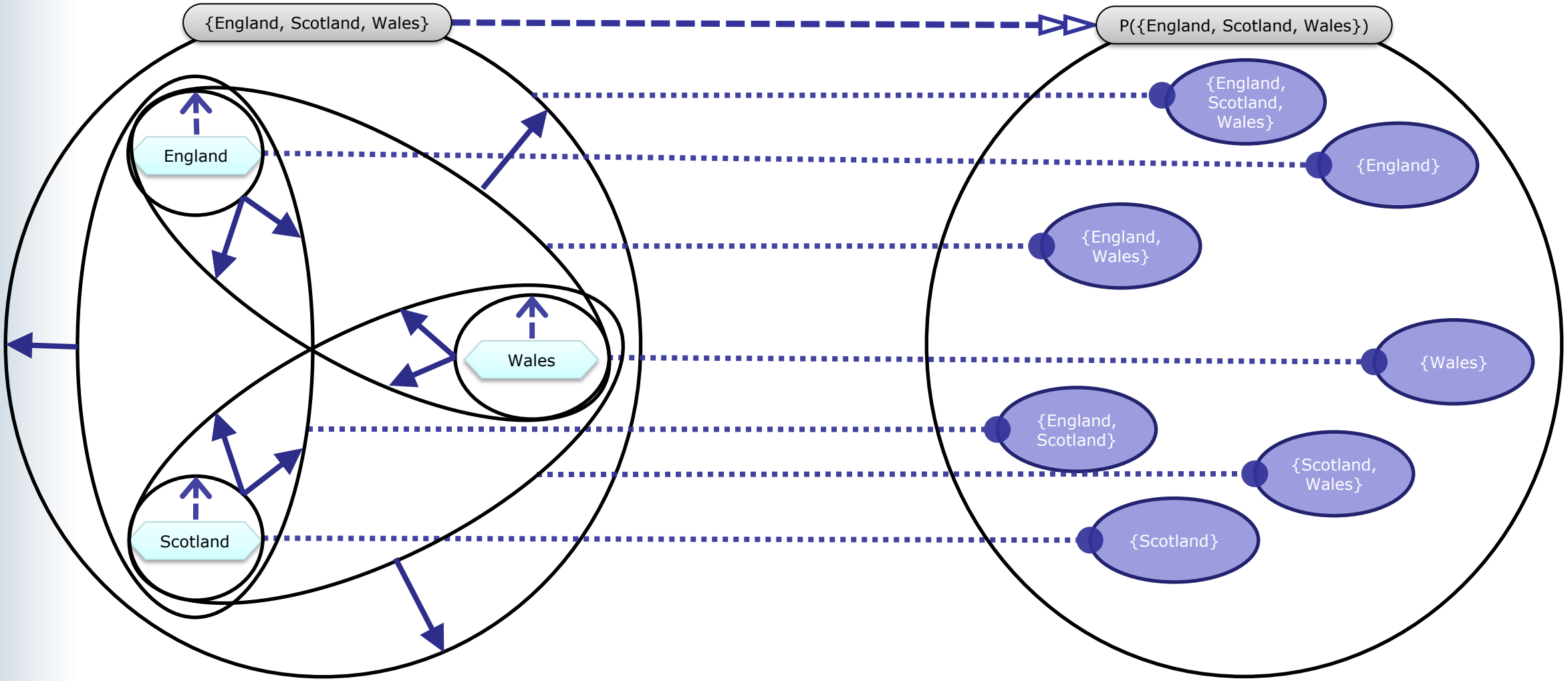


OED powerset pattern: two member types



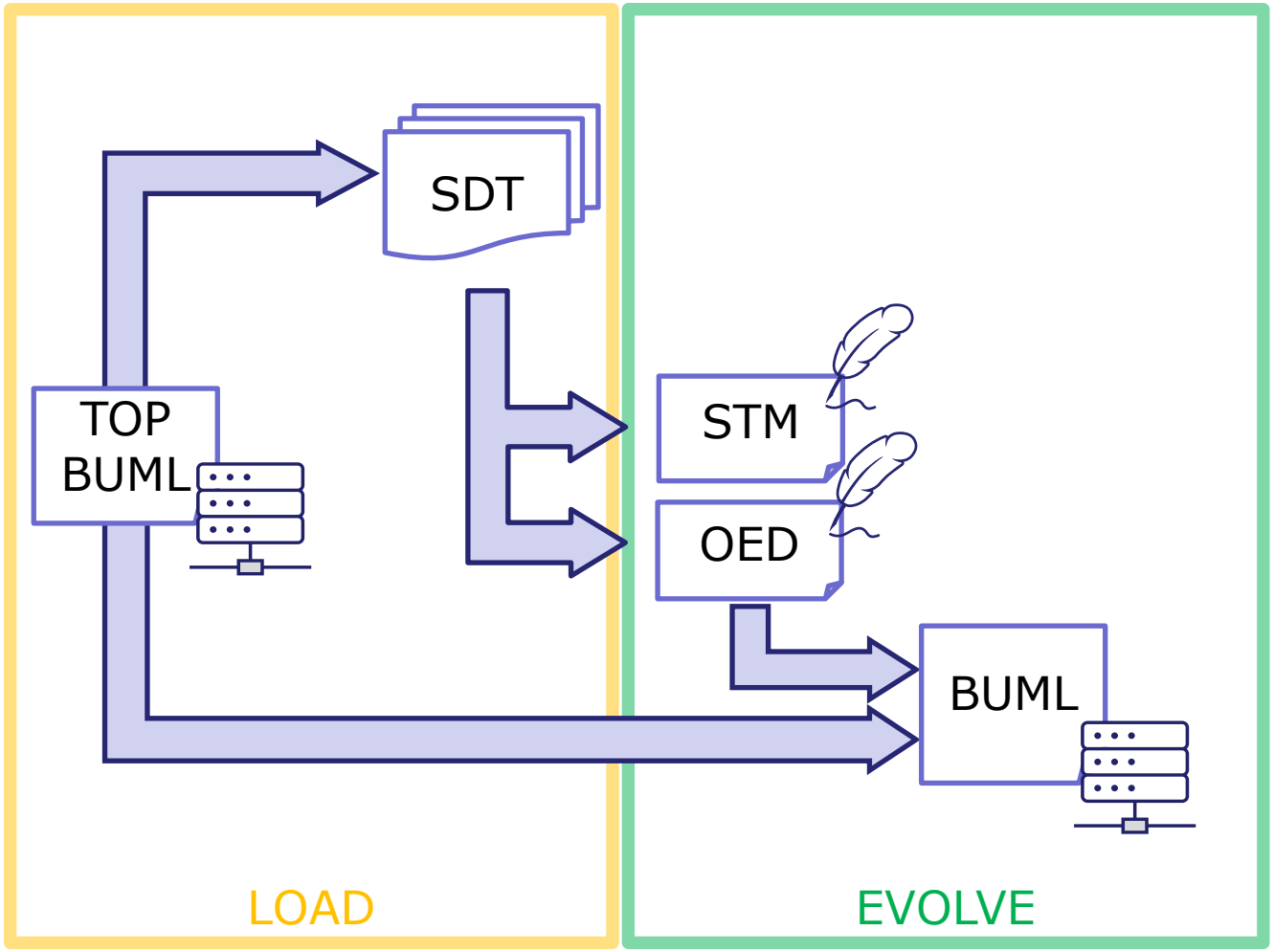


Multiple order patterns: OED powerset pattern



BORO UML

Practical problem process

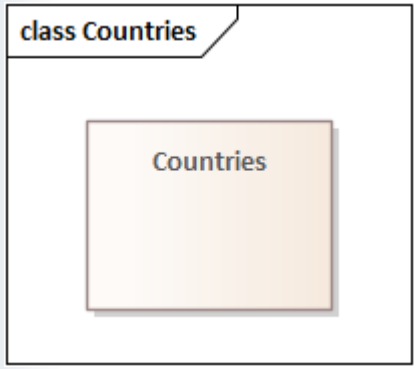


Background

- BUML is the BORO variant of UML
- We use UML notation to visualise BORO ontological structure
  - UML classes
  - UML generalizations, dependencies, n-ary associations and associations
- In addition, we make use of stereotypes, for various purposes, e.g., to indicate the ontological category of the represented objects
- All of these are used with a specific sense, which sometimes may deviate from the UML conventions. The BUML sense is defined by the underlying constructional semantics
- We use Enterprise Architect to draw diagrams, but other software options are available

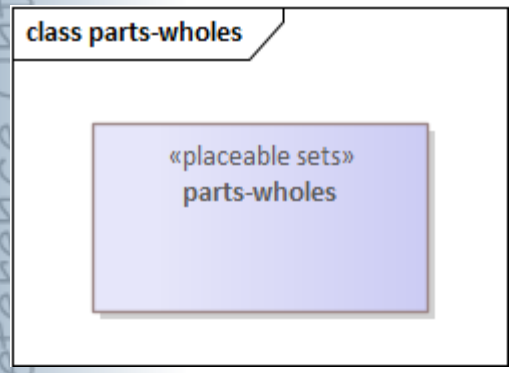
Use of UML notation

Use of UML classes



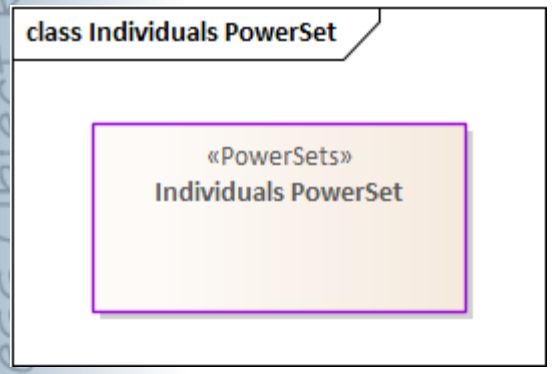
We use UML classes to represent sets, e.g., sets of Countries.

Use of UML classes



Sometimes, sets contain higher-order objects, like tuples.

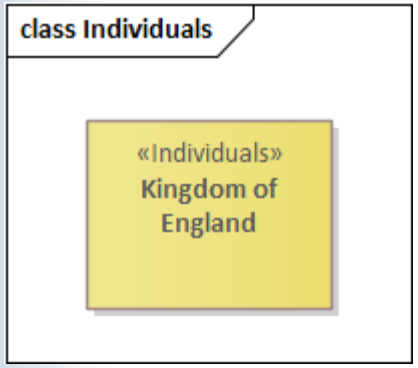
Stereotype <<placeable sets>> refers to the ontic category of the represented objects,
But we do not expect you to follow this convention in your work study.



Sets may also contain sets.

Stereotype <<PowerSets>> refers to the ontic category of the represented objects,
But we do not expect you to follow this convention in your work study.

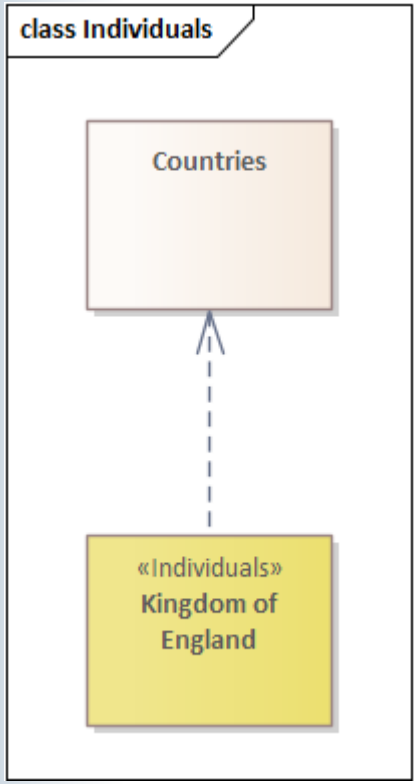
Use of UML classes



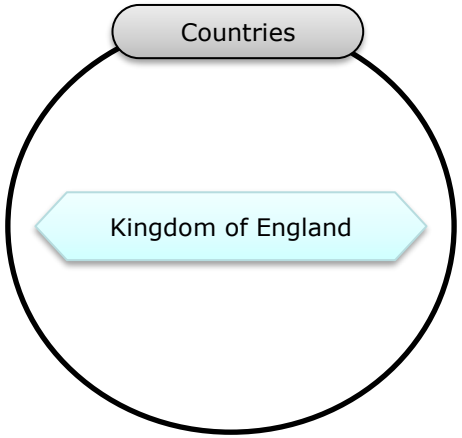
Also we use UML classes to represent Individuals.

Stereotype <<Individuals>> shows the ontic category of the represented object, we do not expect you to follow this convention in your work study.

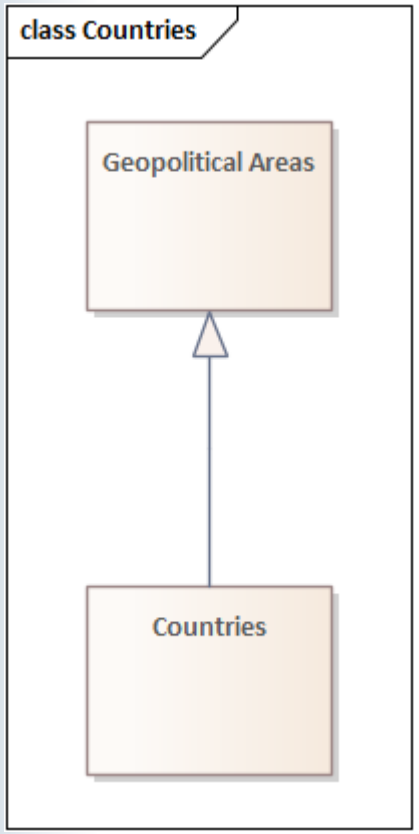
Use of UML dependencies



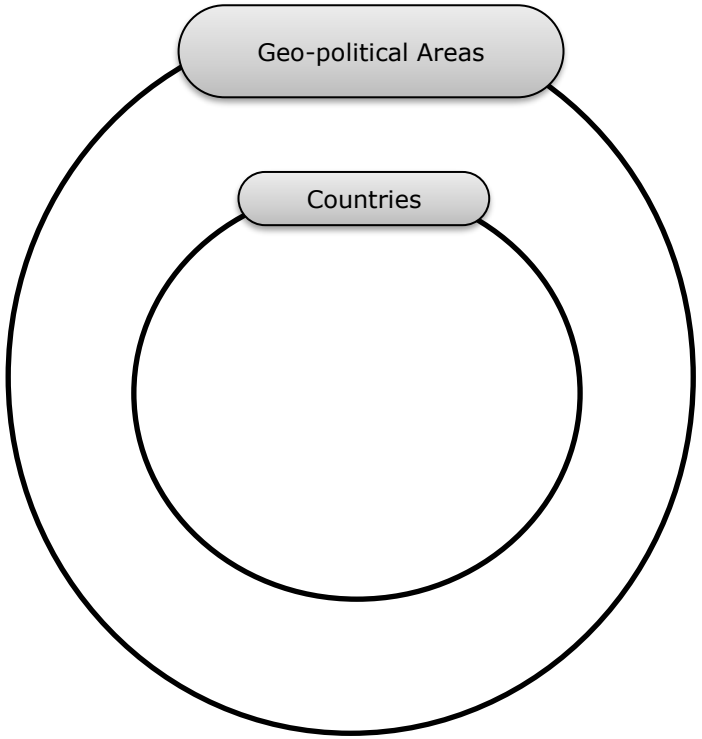
Dependency link shows element-sets (tuple).



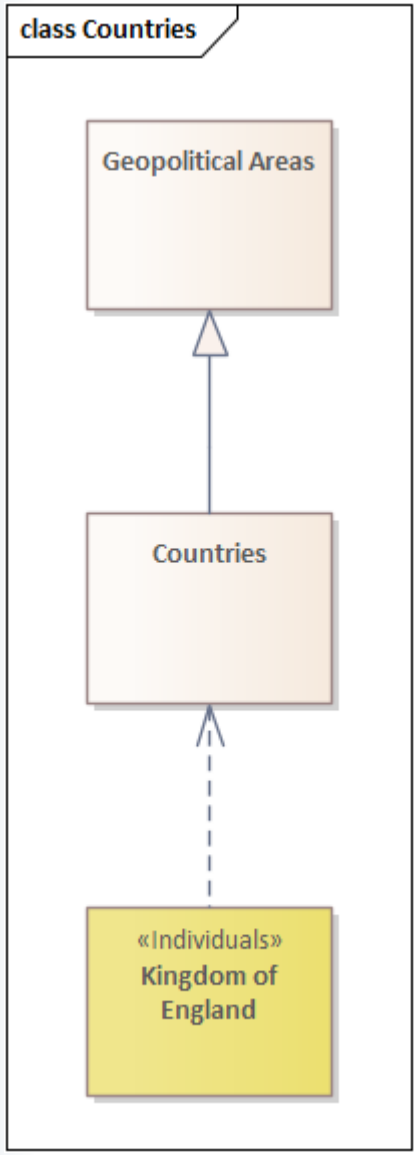
Use of UML generalizations

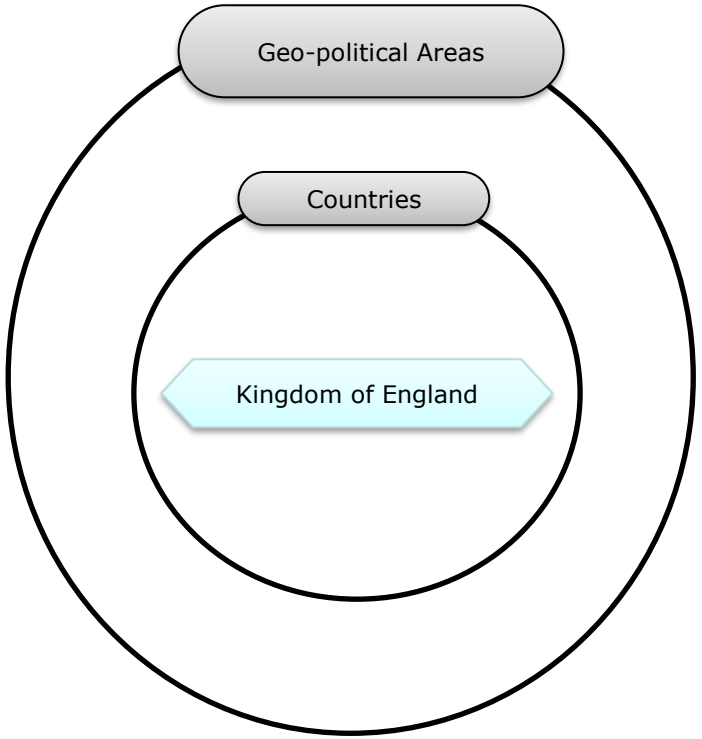


Generalization link shows sub-super-sets (tuple).



Use of UML dependencies and generalizations



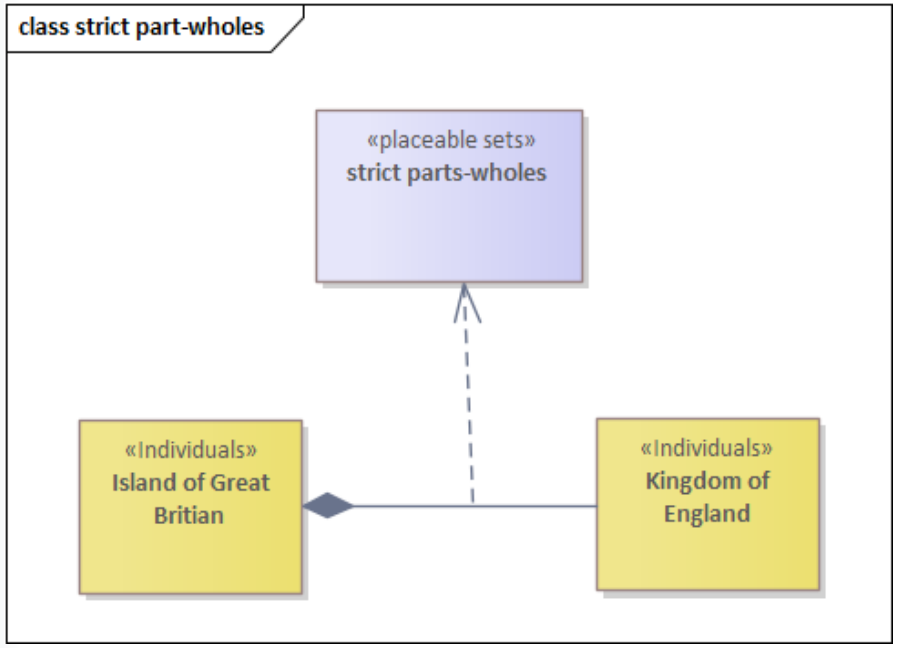


Business Object Reference Ontology

Patterns

Parts-wholes - basic pattern

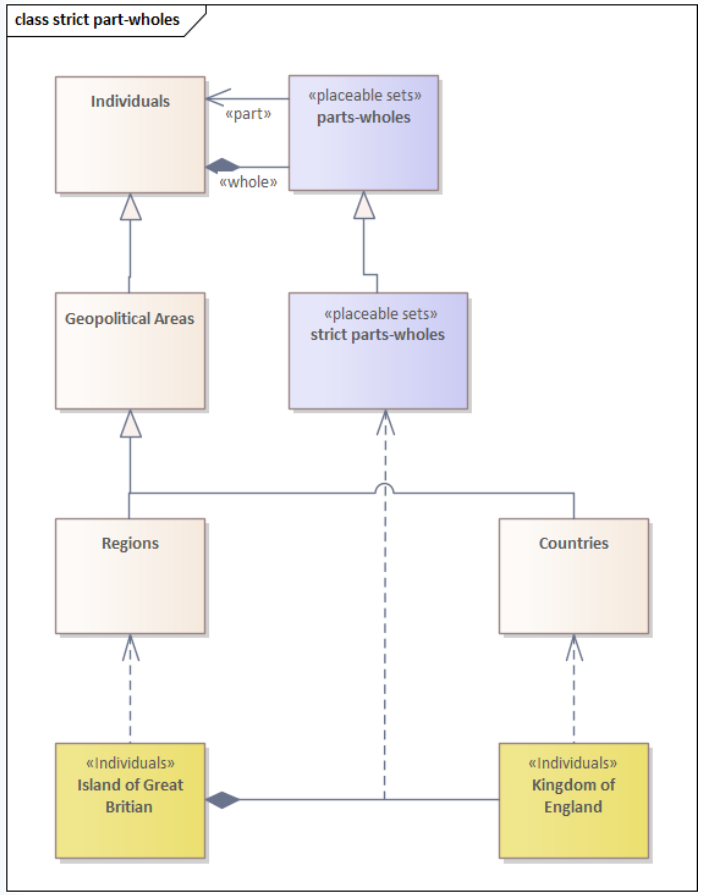
strict = proper (excl. identity)



UML composition shows part-whole (tuple).

<<place1>> and <<place2>> association stereotypes show, respectively, the whole and the part.

Parts-wholes - full pattern

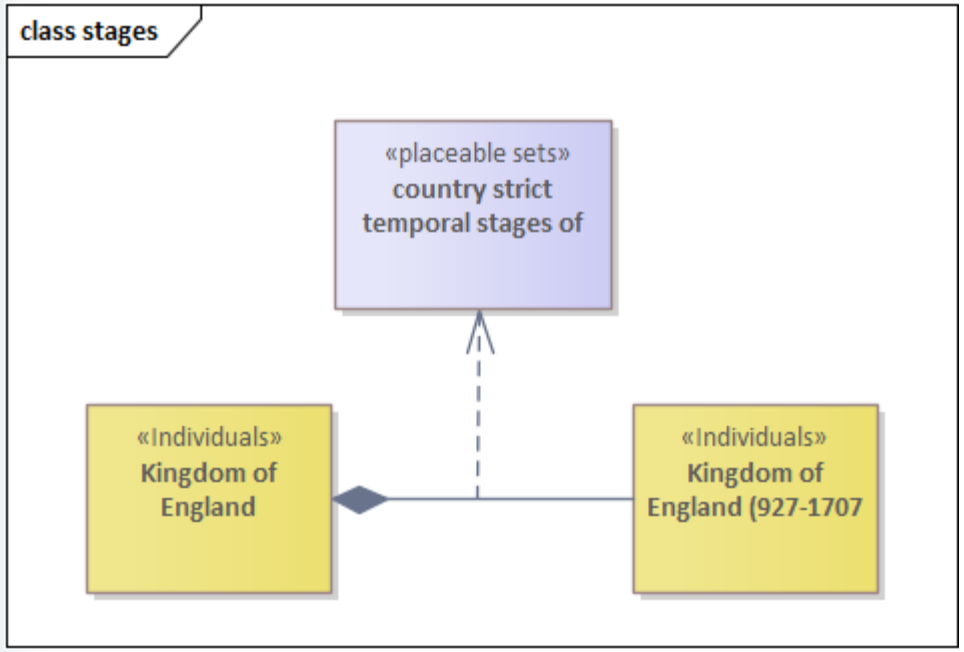


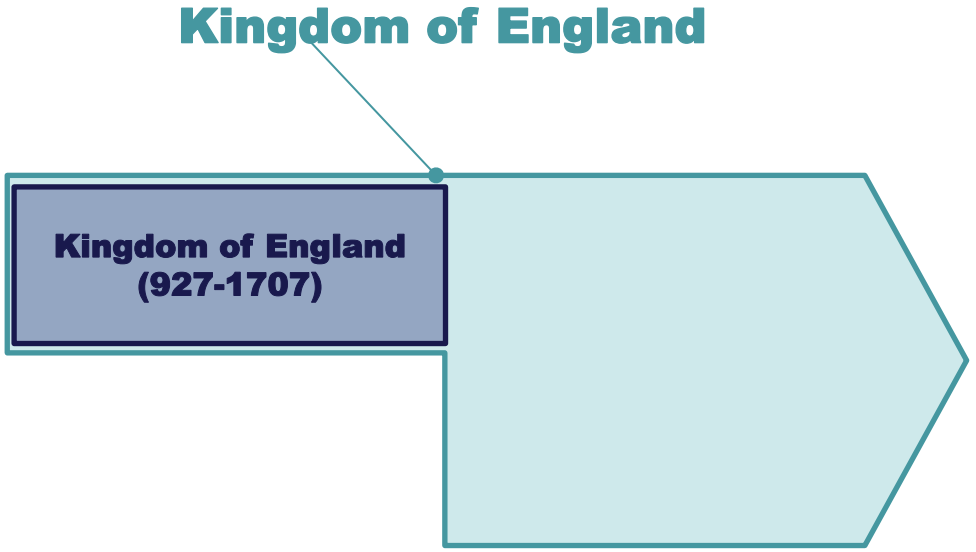
parts-wholes includes identity tuples,
e.g., <London, London>.

strict parts-wholes is a **set** of tuples, so it can have supersets (and subsets).

<<part>> and <<whole>> association stereotypes show, respectively, the whole and the part.

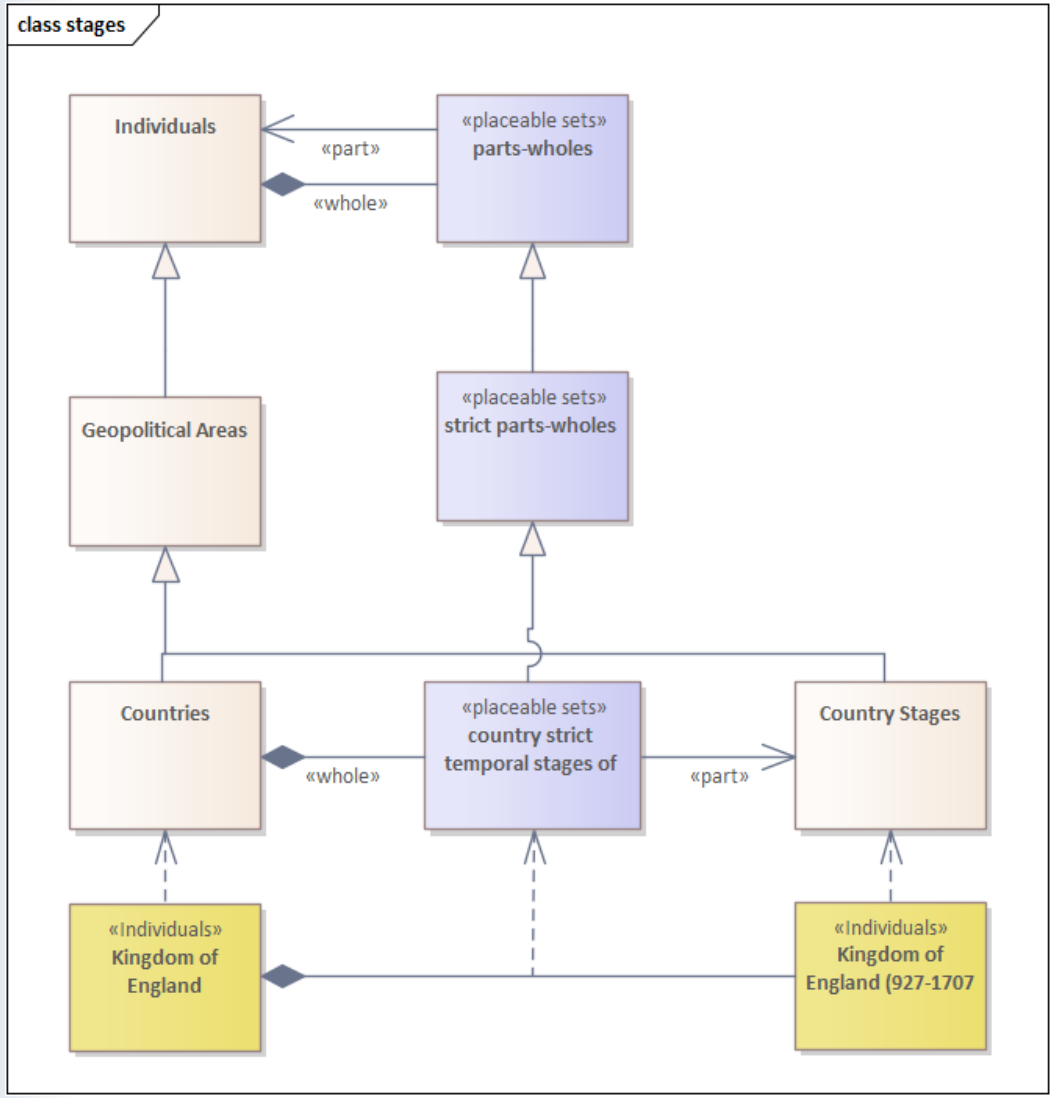
Temporal stages – basic pattern

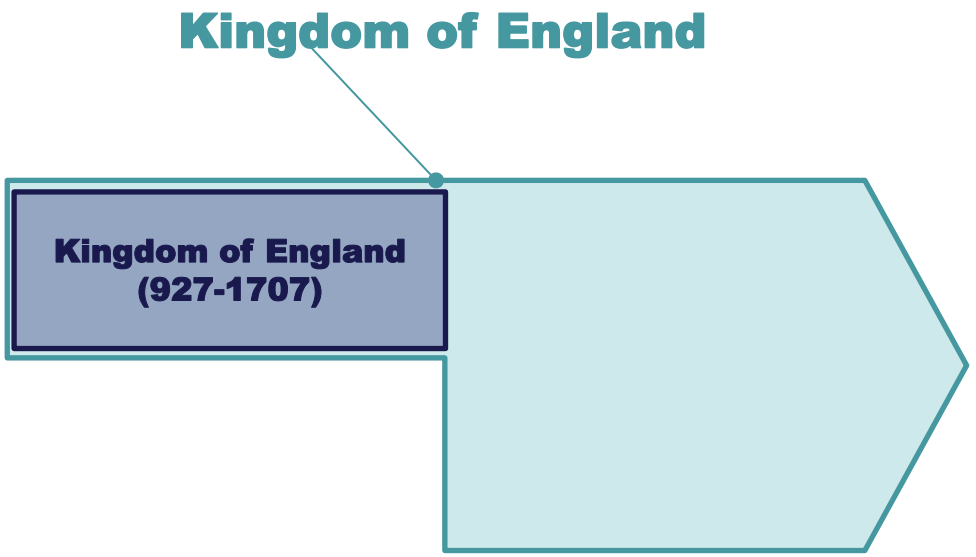




UML composition shows the temporal-stages (tuple).

Temporal stages – full pattern



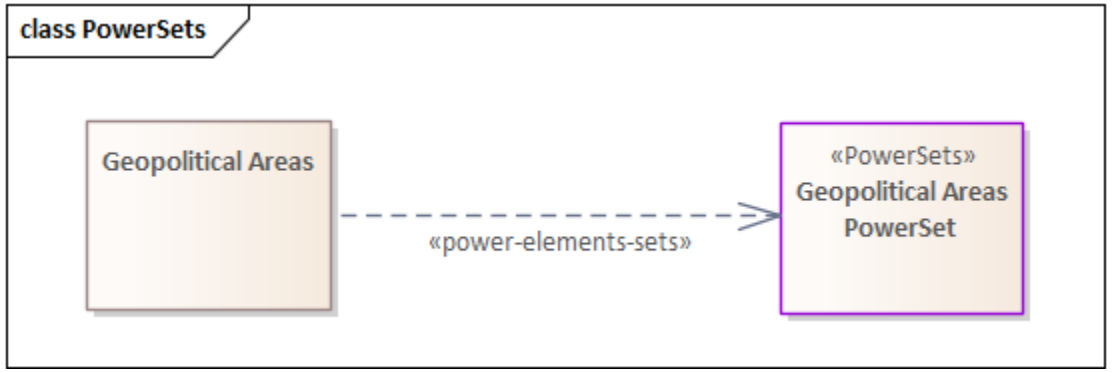


country strict temporal stages of is a **set** of tuples, so it can have supersets (and subsets)

Powerset pattern

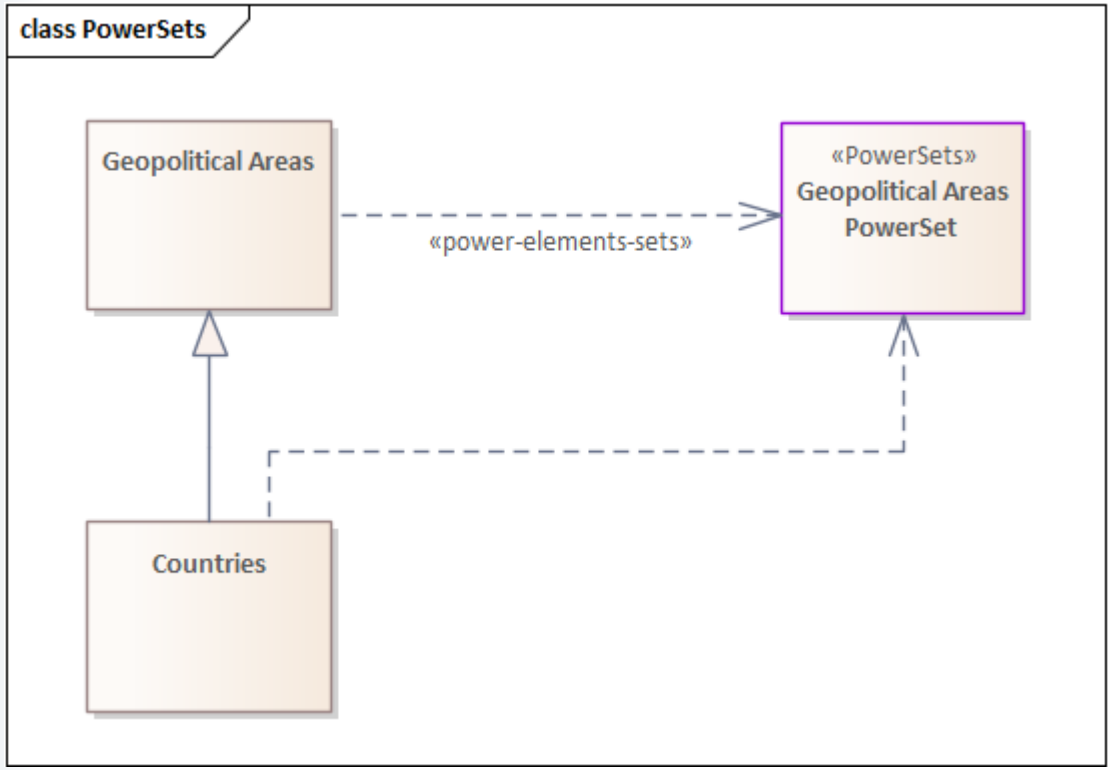
set of
geopolitical
areas

set of
all subsets of
set of
geopolitical
areas



dependency stereotype shows the set and its powerset

Powerset pattern – implicit tuples

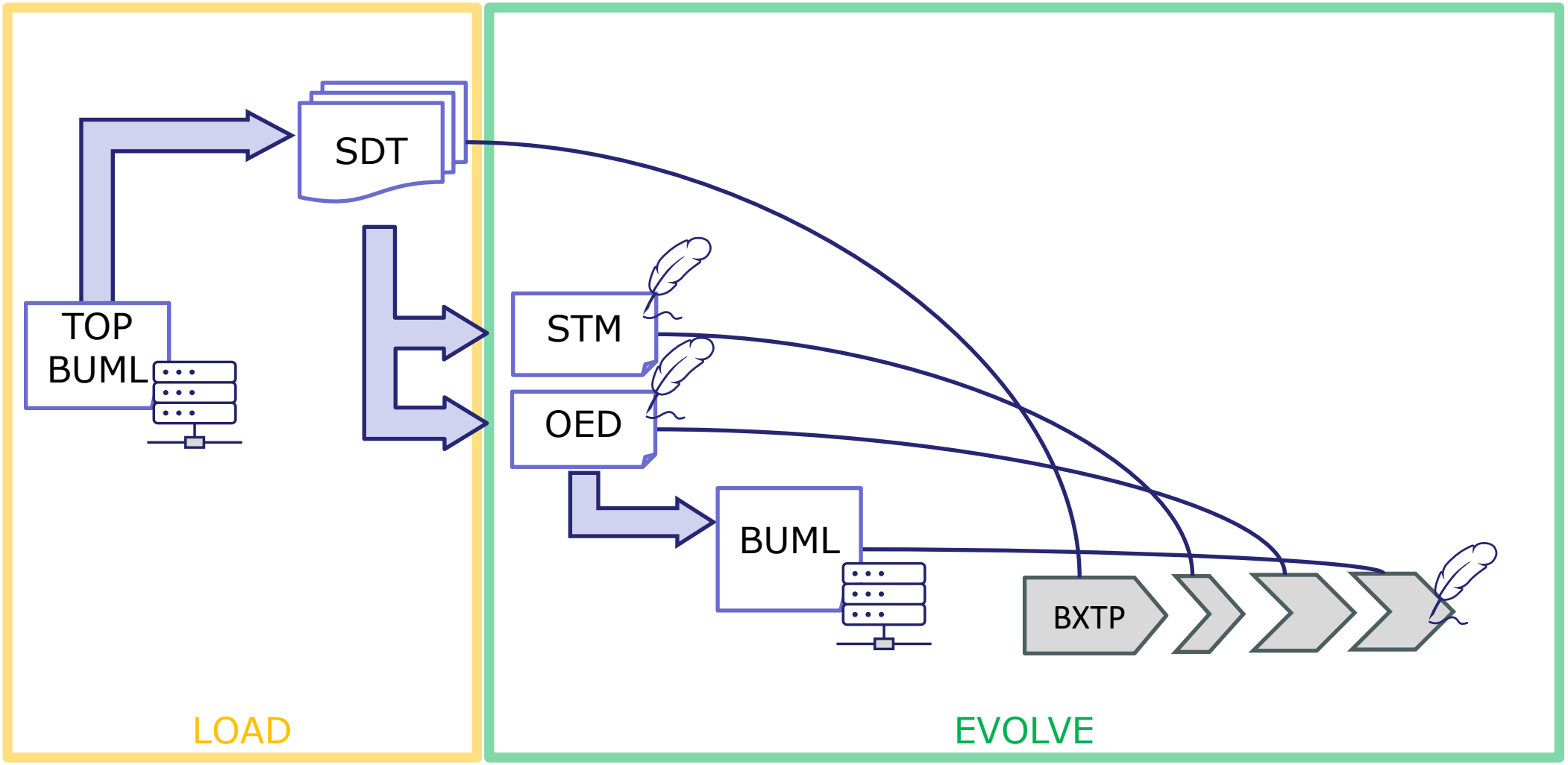


Dependency shows implicit element-set (tuple).

Business Object Reference Ontology

BORO eXcel Table (Manual) Pipeline

Practical problem process



Micro transformation pipeline architecture

- Define by contrast
  - not a single monolithic structure
    - for example, a single complex transformation between two schemas
- Pipeline consists of a network of simple micro transformations
  - the micro transformations are:
    - algorithmic – they can be coded
    - simple to allow for maximal inspectability
    - are repeatable
    - aim to be reusable
      - there is an ‘art’ to finding high levels of reusability

‘Formal’ transformation patterns

- Repeatable, reusable ‘formal’ patterns
  - can use same computer code
    - with enough work

LOAD – bie-ise identifier pattern

	A	B
1	path_names	fso_types
2	D:\folder_1	folder
3	D:\folder_1\file1.txt	file
4	D:\folder_1\file2.txt	file
5	D:\folder_1\file3.txt	file
6	D:\folder_2	folder
7	D:\folder_2\file4.txt	file
8	D:\folder_3	folder
9	D:\folder_3\file5.txt	file
10	D:\folder_3\file6.txt	file
11		

a_file_system_objects

	A	B	C
1	bie_ids	path_names	fso_types
2	K7TGBEHQT763L	D:\folder_1	folder
3	NIPIIQHJWF46B	D:\folder_1\file1.txt	file
4	G3MKPAJIN4E5O	D:\folder_1\file2.txt	file
5	2XIEV7LZZPSVZ	D:\folder_1\file3.txt	file
6	LUT4KTDY4Q3AE	D:\folder_2	folder
7	7BV66YAOOAPSA	D:\folder_2\file4.txt	file
8	YMQMY6AZ4A3SO	D:\folder_3	folder
9	3RIIKMPJMO2W4	D:\folder_3\file5.txt	file
10	X6KVOFQU63LAQ	D:\folder_3\file6.txt	file
11			

file_system_objects | files | folders

A column is added to the beginning of the table

- Its name is ‘bie_ids’
- The cells uniquely identify the row of the table

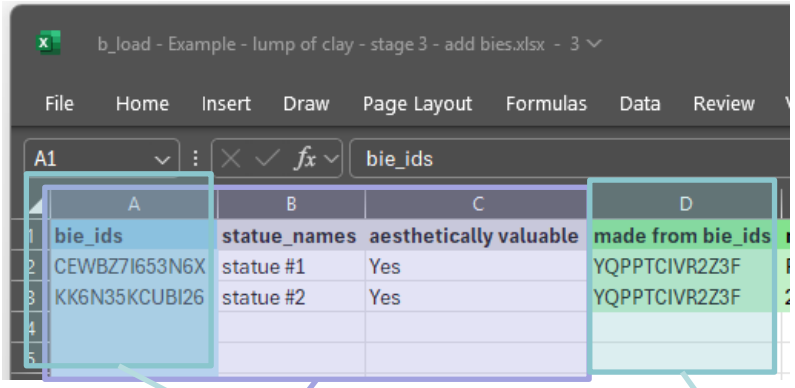
No changes are made to the other columns

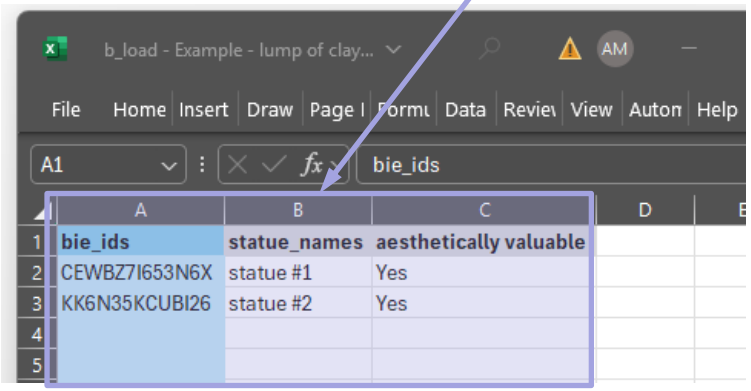
Column split pattern

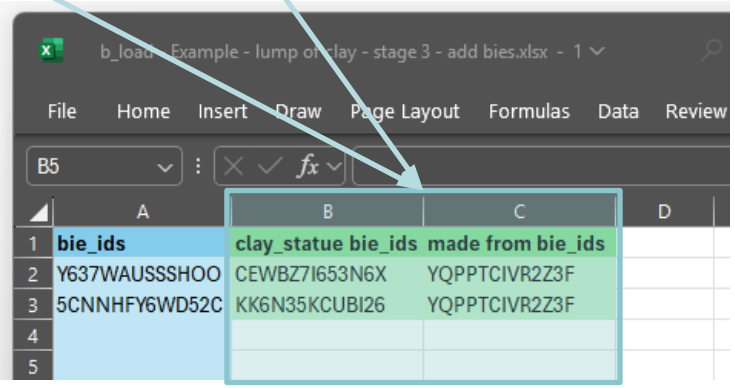
Note: the identifying bie_ids column is ‘inherited’ by both split tables.
The pattern of inheritance can be of two types.

1. Identifier preserving
2. Identifier introducing.

This is driven by the intuition of the intended identity.







Row split pattern

	A	B	C
1	bie_ids	path_names	fso_types
2	K7TGBEHQT763L	D:\folder_1	folder
3	NIPIIQHJWF46B	D:\folder_1\file1.txt	file
4	G3MKPAJIN4E5O	D:\folder_1\file2.txt	file
5	2XIEV7LZZPSVZ	D:\folder_1\file3.txt	file
6	LUT4KTDY4Q3AE	D:\folder_2	folder
7	7BV66YAOOAPSA	D:\folder_2\file4.txt	file
8	YMOMY6AZ4A3SO	D:\folder_3	folder
9	3RIIKMPJMO2W4	D:\folder_3\file5.txt	file
10	X6KVOFQU63LAQ	D:\folder_3\file6.txt	file
11			

file_system_objects | files | folders

	A	B	C
1	bie_ids	path_names	fso_types
2	NIPIIQHJWF46B	D:\folder_1\file1.txt	file
3	G3MKPAJIN4E5O	D:\folder_1\file2.txt	file
4	2XIEV7LZZPSVZ	D:\folder_1\file3.txt	file
5	7BV66YAOOAPSA	D:\folder_2\file4.txt	file
6	3RIIKMPJMO2W4	D:\folder_3\file5.txt	file
7	X6KVOFQU63LAQ	D:\folder_3\file6.txt	file
8			

file_system_objects | files | folders

	A	B	C
1	bie_ids	path_names	fso_types
2	K7TGBEHQT763L	D:\folder_1	folder
3	LUT4KTDY4Q3AE	D:\folder_2	folder
4	YMQMY6AZ4A3SO	D:\folder_3	folder
5			

file_system_objects | files | folders

Typically:

- The column heading row is in ALL output tables.
  - The output tables all have the same format
- All rows are in one or other of the split tables.

Implicit foreign key bie-ise pattern – stage 1 – add bie_ids

bie_ids	statue_names	aesthetically valuable	made from	made on	destroyed on
CEWBZ7I653N6X	statue #1	Yes	lump of clay #1	2023-06-20	2023-10-24
KK6N35KCUBI26	statue #2	Yes	lump of clay #1	2024-07-06	

b1_clay_statues | b1_times | c1_clay_statues | c2_clay_statues | c3_c

Match the value in the Foreign table

bie_ids	time_names	YYYY-MM-DD
R7E3FV2GKUN5F	t1	2023-06-20
Y2IXUKCPJUS6Y	t2	2023-10-24
24J67PH4E27UG	t3	2024-07-06

b1_times | c1_clay_statues

Some columns will be foreign keys
This needs to be made explicit

Add bie id to new column

bie_ids	statue_names	aesthetically valuable	made from	made from bie_ids	made on	made on bie_ids	destroyed on	destroyed on bie_ids
CEWBZ7I653N6X	statue #1	Yes	lump of clay #1	YQPPTCIVR2Z3F	2023-06-20	R7E3FV2GKUN5F	2023-10-24	Y2IXUKCPJUS6Y
KK6N35KCUBI26	statue #2	Yes	lump of clay #1	YQPPTCIVR2Z3F	2024-07-06	24J67PH4E27UG		

b1_clay_statues | b1_times | c1_clay_statues | c2_clay_statues | c3_c

Implicit foreign key bie-ise pattern – stage 2 – remove original columns

	A	B	C	D	E	F	G	H	I
1	bie_ids	statue_names	aesthetically valuable	made from	made from bie_ids	made on	made on bie_ids	destroyed on	destroyed on bie_ids
2	CEWBZ7I653N6X	statue #1	Yes	lump of clay #1	YQPPTCIVR2Z3F	2023-06-20	R7E3FV2GKUN5F	2023-10-24	Y2IXUKCPJUS6Y
3	KK6N35KCUBI26	statue #2	Yes	lump of clay #1	YQPPTCIVR2Z3F	2024-07-06	24J67PH4E27UG		
4									
5									

b1_clay_statues | b1_times | c1_clay_statues | c2_clay_statues | c3_c

	A	B	C	D	E	F
1	bie_ids	statue_names	aesthetically valuable	made from bie_ids	made on bie_ids	destroyed on bie_ids
2	CEWBZ7I653N6X	Clay Statue #1	Yes	YQPPTCIVR2Z3F	R7E3FV2GKUN5F	Y2IXUKCPJUS6Y
3	KK6N35KCUBI26	Clay Statue #2	Yes	YQPPTCIVR2Z3F	24J67PH4E27UG	
4						

b_sub-super-sets | c1_clay_statues | c2_clay_statues | c3_clay_statues | c3_statues_temporally

Table-Column heading pushdown pattern

	A	B
1	bie_ids	table_names
2	M5HLU3UUJ727D	lumps_of_clay
3	CBFGCPVTIUQ7R	clay_statues
4		
5		
6		
7		

stages | tables

Add table to Objects

	A	B
1	bie_ids	object names
2	M5HLU3UUJ727D	lumps_of_clay
3	YQPPTCIVR2Z3F	
4	CBFGCPVTIUQ7R	clay_statues
5	CEWBZ7I653N6X	
6	KK6N35KCUBI26	
7		

c5_objects | c5_ta

Add row to Objects

	A	B	C
1	bie_ids		
2	CEWBZ7I653N6X		
3	KK6N35KCUBI26		
4			

c4_clay_statues

And add the relations

	A	B	C
1	table_bie_ids	row_bie_ids	
2	M5HLU3UUJ727D	YQPPTCIVR2Z3F	
3	CBFGCPVTIUQ7R	CEWBZ7I653N6X	
4	CBFGCPVTIUQ7R	KK6N35KCUBI26	
5			
6			
7			

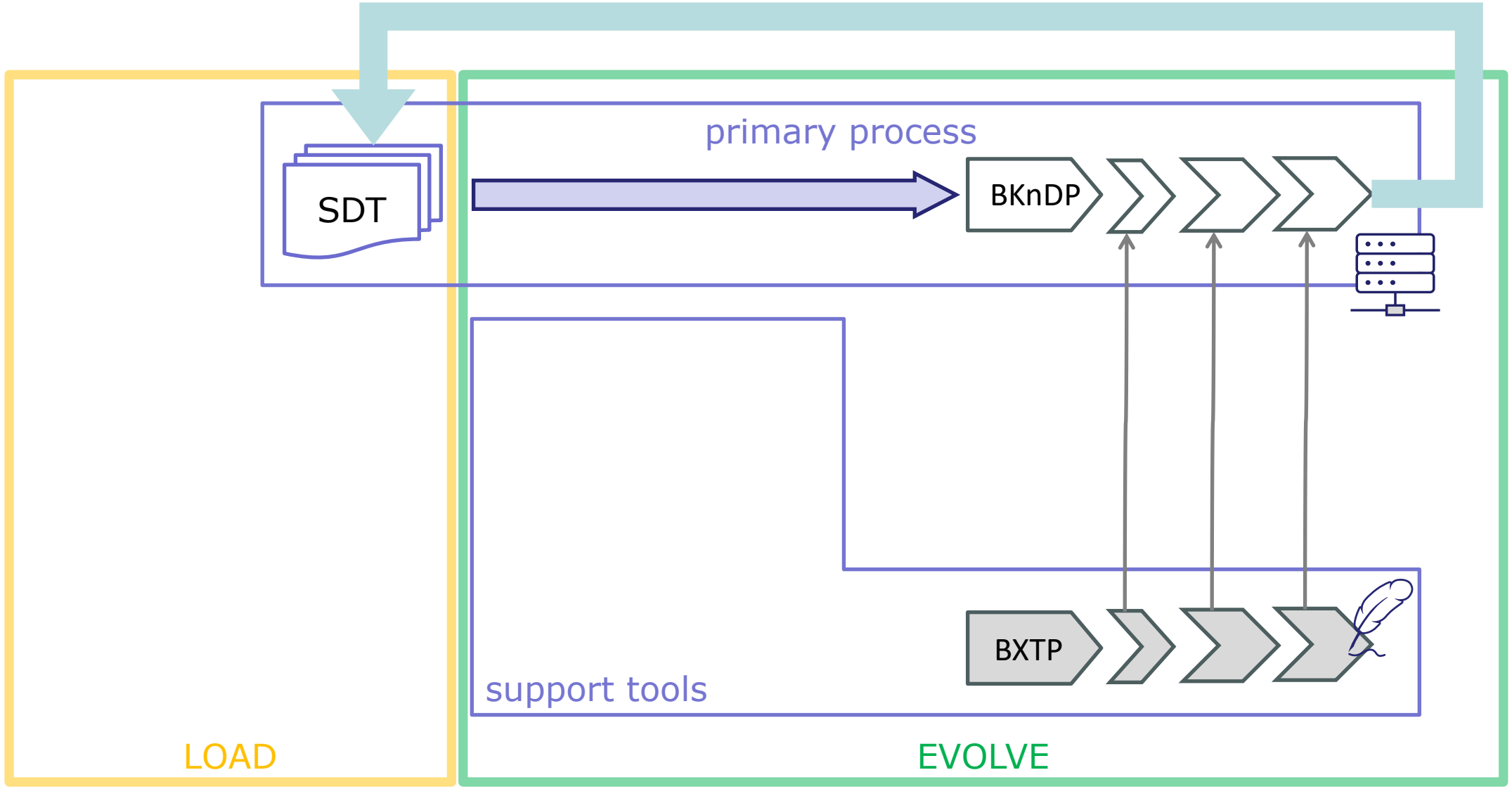
c5_table-row_relations

Tables and column headings can represent objects
Useful to have these as rows in the domain.

Business Object Reference Ontology

BORO KNIME Data Pipeline

Practical problem process



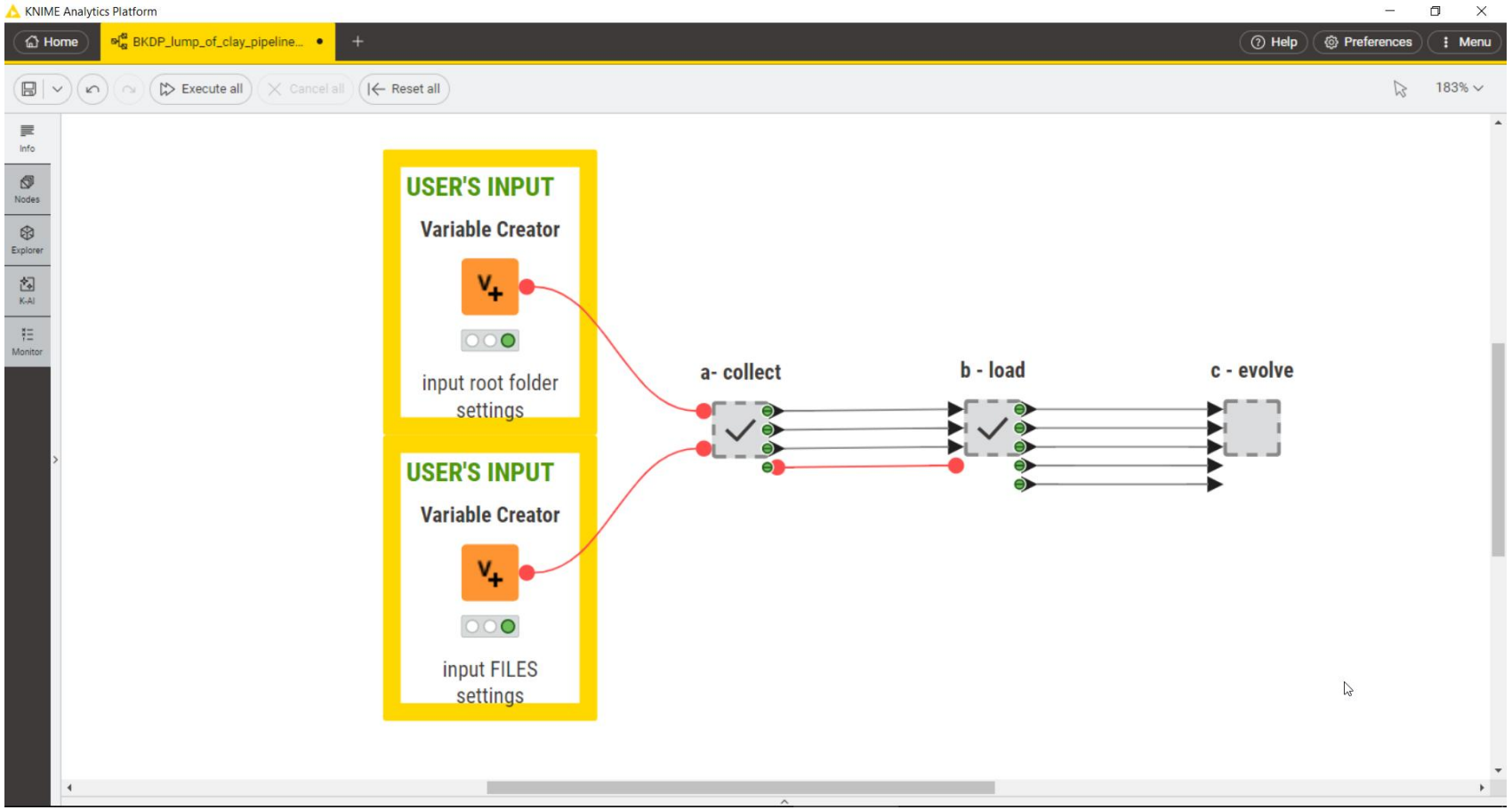
Goal

- To attempt to automatically (using a machine/computer) transform the input into the designed output
  - acquire a feel for
    - the constraints upon these kinds of machine pipeline transformations
    - the nature of these kinds of machine pipeline transformations
- To attempt to build reusable micro transformations
  - acquire a feel for the reuse economics of transformation patterns

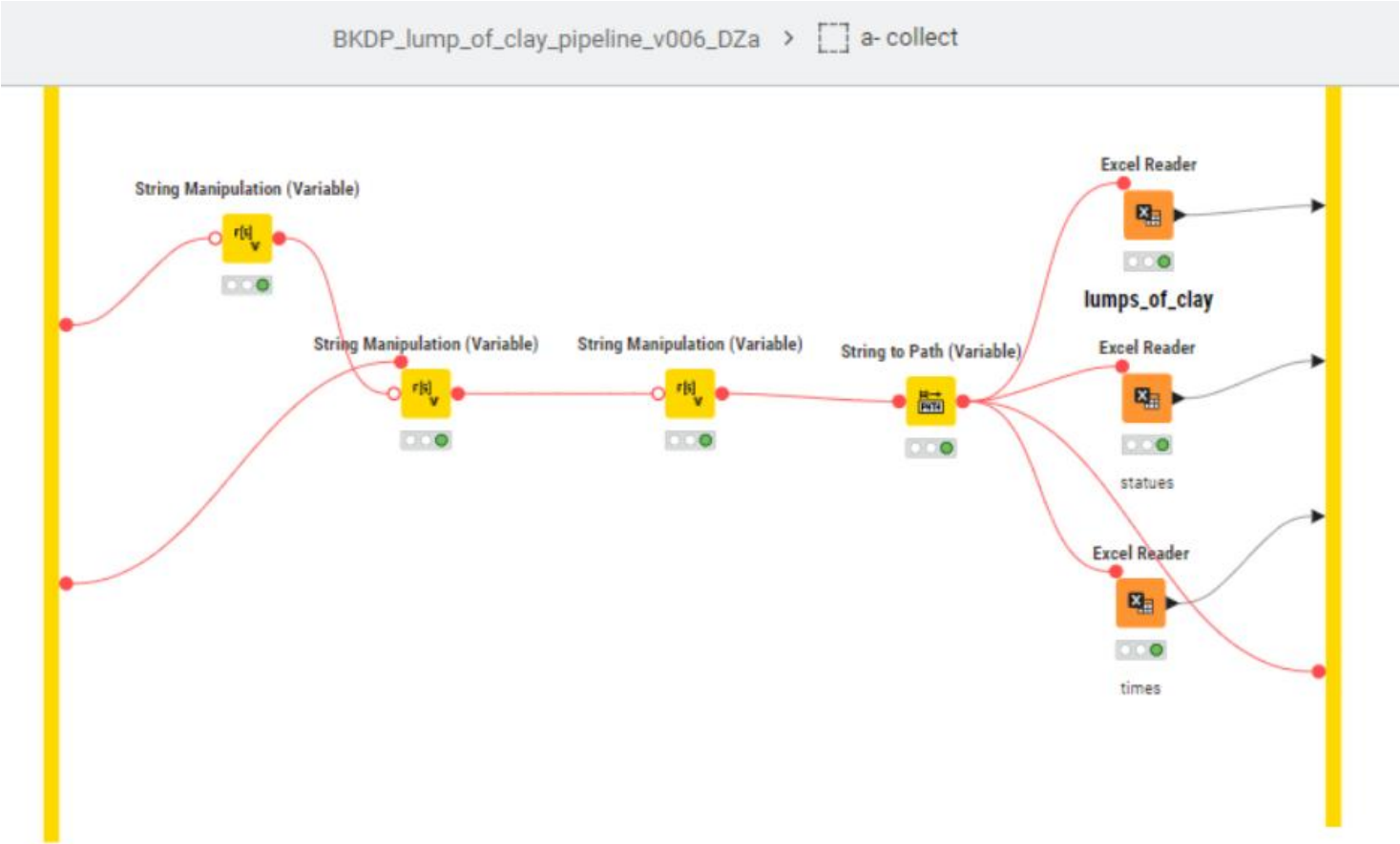
BORO KNIME Data Pipeline

- The ‘BORO eXcel Table (Manual) Pipeline’ should provide a sufficiently detailed specification of most of the micro transformations
  - the KNIME is an implementation of the specification
  - Developing the code to execute the transformation automatically should clarify the formal moves that need to be made

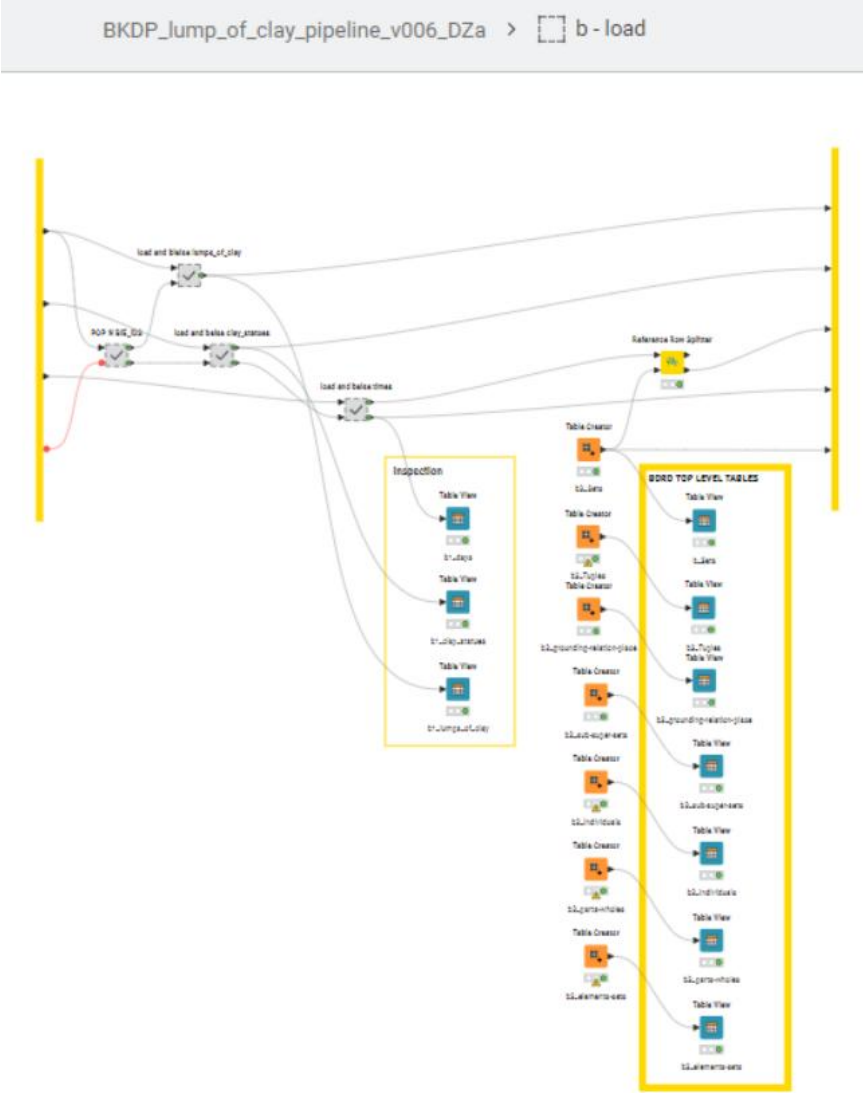
Knime top level – bCLEARer stages



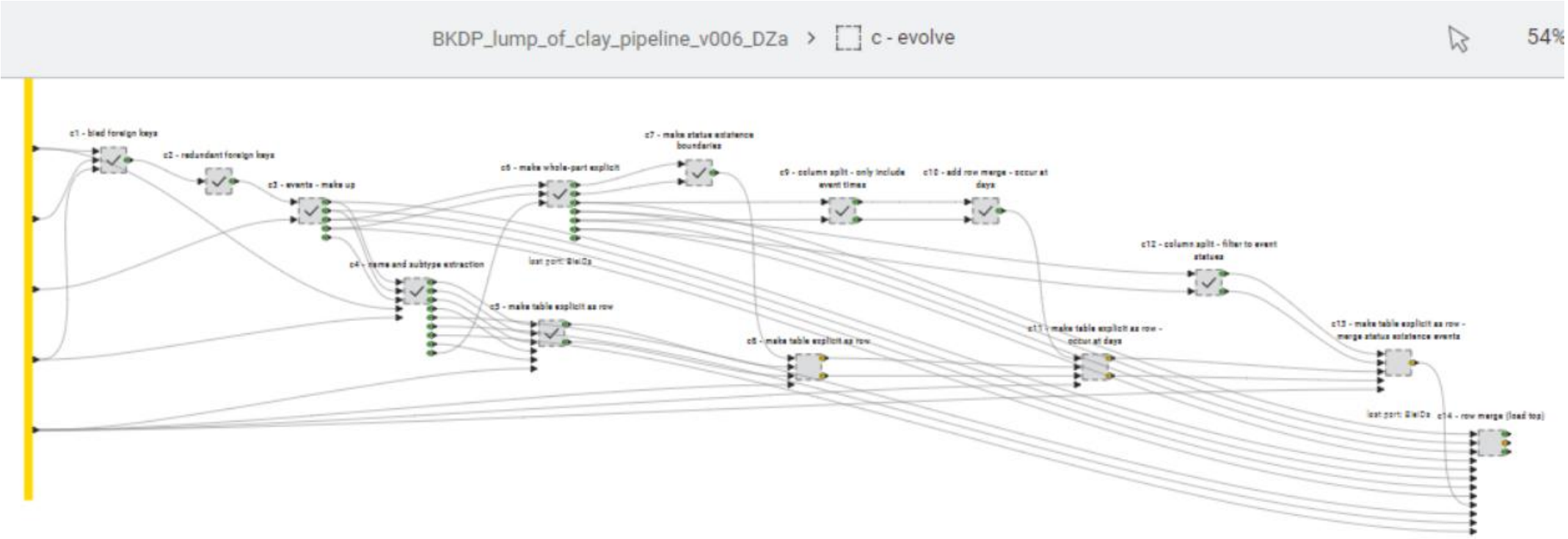
Knime COLLECT – done



Knime LOAD – done



Knime Evolve – work in here



Business Object Reference Ontology

Project planning

Suggested project planning

- Remember everything is on GitHub
- Suggest you plan your ‘project’
  - probably better to do a little of each task,
    - rather than just complete the first few
- Task order – earlier tasks feed into later tasks
  1. STM
  2. OED
  3. BUML
  4. BXTP
  5. BKnDP
- Try to do at least 1, 2 and 4
  - if you can, try a little 5 (block out the last half hour to try)
- Remember task 5 has starter code

Questions

THE END.