

Taking advantage of Conceptual Schemas (i.e. Ontologies)

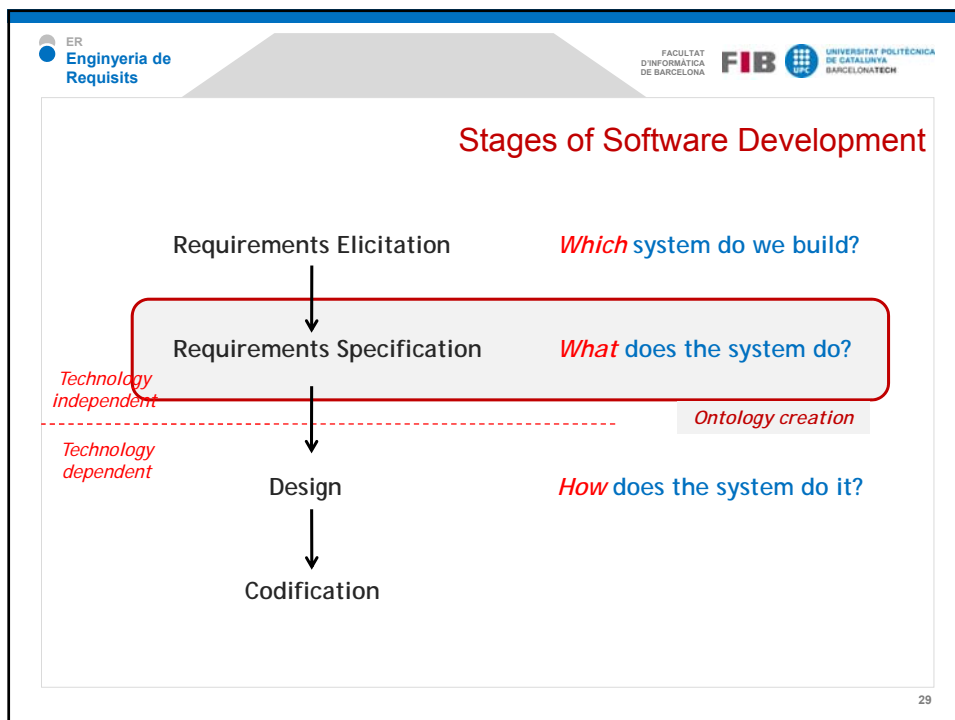
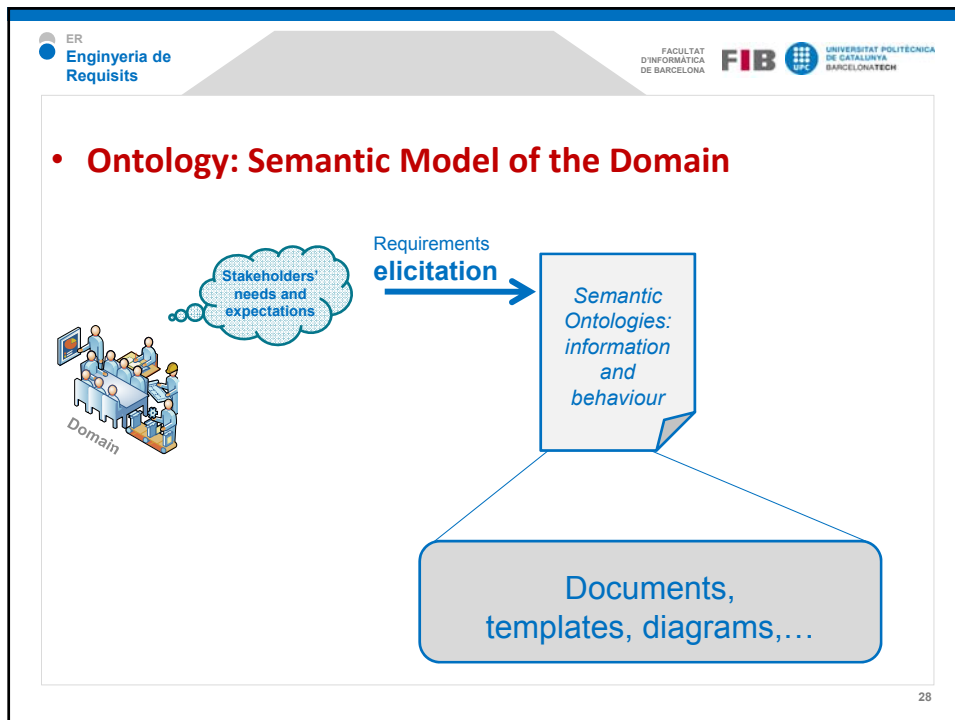
26

Ontology (information science)

"an ontology encompasses a representation, formal naming and definition of the categories, properties and relations between the concepts, data and entities that substantiate one, many or all domains (wikipedia)"

- Every field creates ontologies to limit complexity and organize information into data and knowledge.
- Having a common vocabulary facilitates problem understanding and solution reuse.
- The term **knowledge graph** it is (sometimes) used as synonym for ontology. A knowledge graph represents a collection of interlinked descriptions of entities - real-world objects, events, situations or abstract concepts. Other synonyms exist.
- There are several languages for specifying ontologies:
 - Description Logics, OWL
 - RDF, RDFS, Jason
 - HL7
 - Schema.org
 - UML, OCL

27

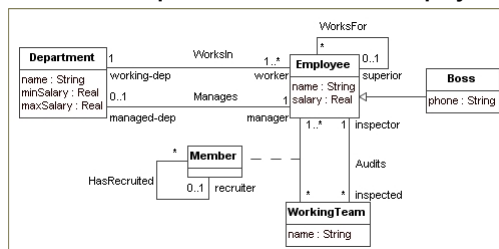


Taking advantage of Ontologies

30

1. Generating Test Data

Give me a sample database where an Employee works for himself



Integrity constraints

```

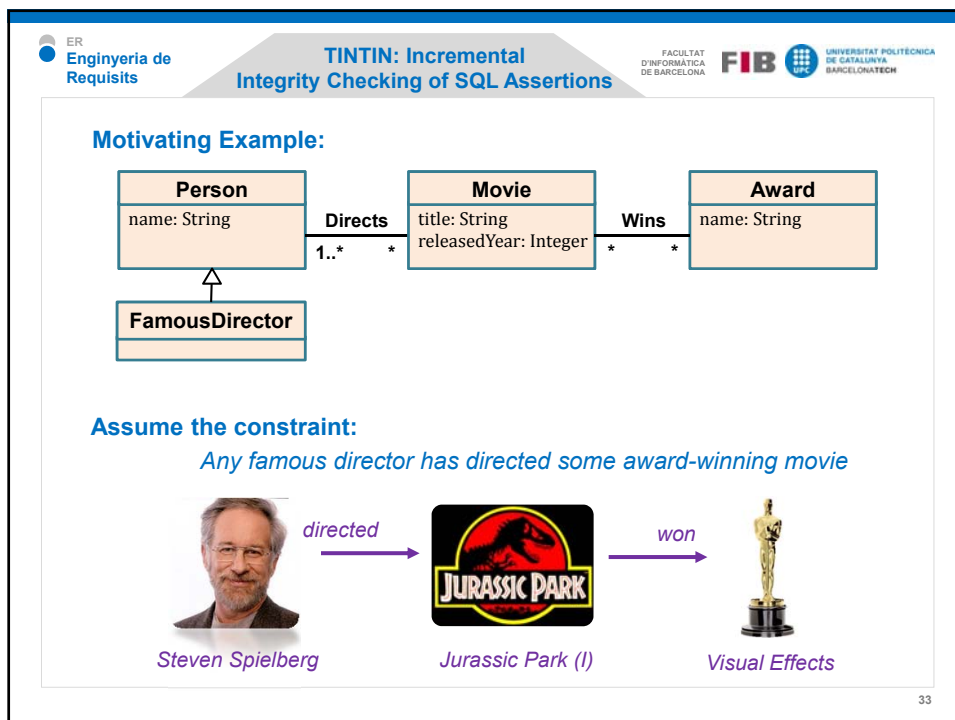
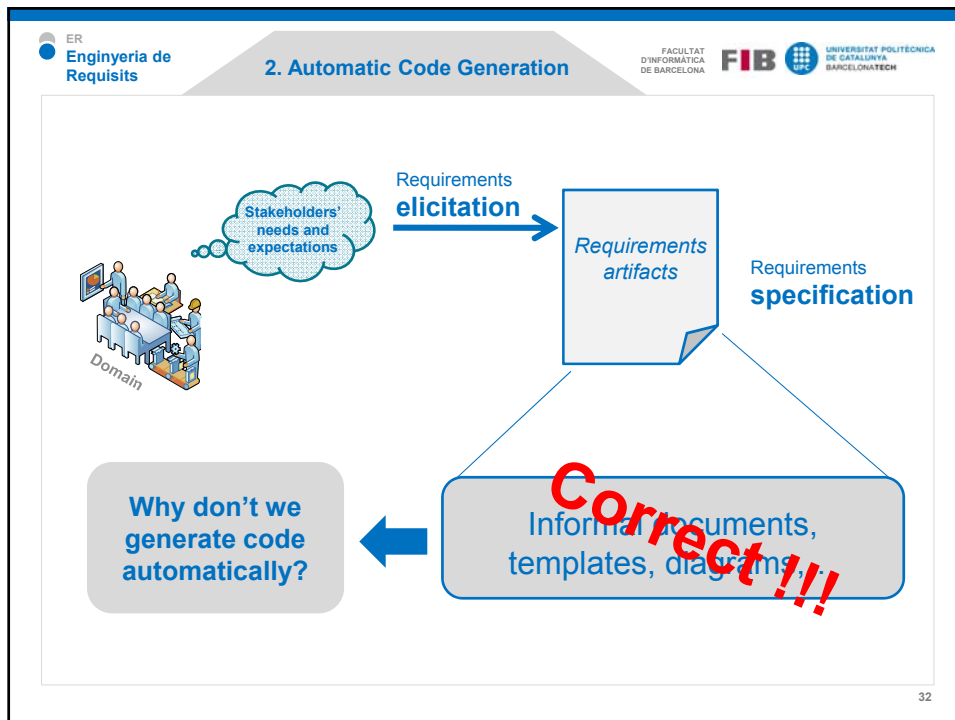
6.context Department inv ManagerIsWorker:
self.worker->includes(self.manager)
7.context Department inv MgrHasNoSuperior:
self.manager.superior->isEmpty()

```

WorksFor	worksFor(#e1, #e1)
Employee	employee(#e1, mary)
WorksIn	worksIn(#e1, #s1)
Department	department(#s1, sales)
Manages	manages(#e1, #s1)

We can define a set of conditions over the data we want to obtain to be able to test a software application (and obtain this data automatically)

31



Running a query looking for the violations

Writing a query returning any famous director who has not directed an award-winning movie. **Empty query = constraint satisfaction**

```
Select * from FamousDirector as FD
where not exists (Select * from Directs as D
                  join Wins as W on (D.movie_id = W.movie_id)
                  where D.person_id = FD.id)
```

Problem: bad performance

Running the query = checking all the data



If we delete 'Jurassic Park' from DB, and run the query, it will search for award-winning movies for all famous directors...

... but the unique relevant one to check is Spielberg!

This is probably why no commercial DBMS implements checking of SQL assertions (SQL92 standard)

34

Manually programming an efficient solution is difficult:

are you sure you are taking all cases into account?



Deleting an award-winning movie from DB causes a violation if...

Which cases must we check to ensure that this is true?

35

Manually programming an efficient solution is difficult:
are you sure you are taking all cases into account?



Deleting an award-winning movie from DB causes a violation...

... unless in the DB there is another award-winning movie directed by the same director...

... such that it is not being deleted in the same transaction too...

... or there is an insertion of a new movie ...

... which should be award-winning and by the same director...

... or the director is being deleted as a famous director



36

We need an automatic method for

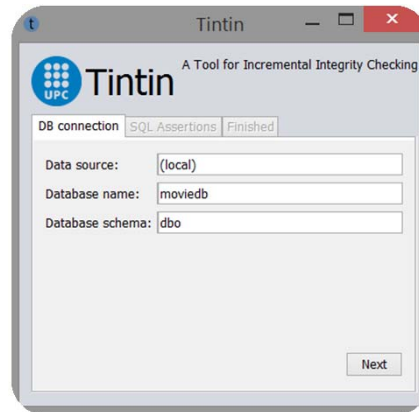
Checking only those parts of the **data** that might violate our defined **constraints** taking in account the **update** being applied

In other words, we need
an **Incremental method for consistency checking**

This is exactly what we provide with **TINTIN**

37

1. Connect TINTIN to your SQL Server DB



38

TINTIN now automatically captures all the insertions/deletions the user wants to apply



If a user sends

delete from directs **where** movie_id = 1
insert into movie **values** (2, 'War Horse', 2012)



The update is not applied, but the tuples the user wants to insert/delete are internally stored in auxiliary SQL tables

DIRECTS	
person_id	movie_id
1	1

Current table

Auxiliary tables

del_DIRECTS	
person_id	movie_id
1	1

ins_MOVIE		
Id	Title	Year
2	War Horse	2012

39

ER
Enginyeria de Requisits

FACULTAT D'INFORMÀTICA DE BARCELONA
FIB
UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

TINTIN: an overview of the tool

1. Connect TINTIN to your SQL Server DB
2. Write your assertions into TINTIN

40

ER
Enginyeria de Requisits

FACULTAT D'INFORMÀTICA DE BARCELONA
FIB
UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

TINTIN: what has happened?

The **safeCommit()** procedure has been created. This procedure looks for ins/deletions of tuples violating your defined assertion/s

The *safeCommit* procedure inspects the auxiliary tables storing the modifications to be applied. If it finds an insertion/deletion causing a violation, the updates are discarded, otherwise, they are committed

FamousDirector	
Id	Name
1	Steven Spielberg

ins_MOVIE		
Id	Title	Year
2	War Horse	2012

del_DIRECTS	
movie_id	person_id
1	1

This is going to violate our assertion!

41

ER

Enginyeria de Requisits


TINTIN: an overview of the tool

FACULTAT D'INFORMÀTICA DE BARCELONA

FIB

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

1. **Connect TINTIN to your DB**
2. **Write your assertions into TINTIN**
3. **Use your DB normally. Just recall to call *safeCommit()* at the end of your transactions**



42

ER

Enginyeria de Requisits

TINTIN: an overview of the tool

FACULTAT D'INFORMÀTICA DE BARCELONA

FIB

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Two simple use case examples:


```

delete from directs
  where movie_id = 1;

insert into movie
  values (2, 'War Horse', 2012);

insert into directs
  values(1, 2)

safeCommit()
```



*1 violation was found.
The update is rejected.*

```


delete from directs
  where movie_id = 1;

insert into movie
  values (2, 'War Horse', 2012);

insert into directs
  values(1, 2)

insert into wins(2, 1)

safeCommit()
```



*No violations were found.
The update is committed.*

43

We have used the TPC-H benchmark, a benchmark for illustrating decision support systems that examine large volumes of data.

- Current data = 1GB * SF
- Data updates = 1MB * SF

Table 1: Time in seconds for checking assertions

	SF = 1		SF = 3		SF = 5	
	Nim	Tintin	Nim	Tintin	Nim	Tintin
#1 Assert.	1.23	0.01	3.82	0.03	69.46	0.06
#2 Assert.	0.89	0.01	2.67	0.04	106.51	0.04
#3 Assert.	0.04	0.07	0.33	0.11	1.23	0.15
#4 Assert.	4.75	1.27	14.64	3.17	88.46	1.29

*Nim = Non Incremental Approach

44

How does it work?

It works with a *logic based core*

SQL is translated into **Logic Denials**,
Logic Denials into **Event Dependency Constraints**,
Event Dependency Constraints into **SQL**

45

Assume the following SQL assertion:

Important managers (i.e. those with a high bonus) work in their departments

```
create assertion "ImportantManagersWorkInTheirDepts" as check (  
not exists (  
  select * from Manages as M  
  where M.bonusSalary > 500 and  
    not exists (select * from WorksIn as W  
      where M.emp = W.emp and M.dept = W.dept)))
```

We assume the database contains the following tables:

Employee (emp, name, ...)

Department (dept, name, ...)

Manages (emp, dept, bonusSalary)

WorksIn (emp, dept)

How can we translate the SQL assertion into logics?

46

Logic Denial

$$\text{manages}(e, d, s) \wedge s > 500 \wedge \neg \text{worksIn}(e, d) \rightarrow \perp$$

How can we check this efficiently (assuming the database is consistent)?

manages will be true in the new (updated) state of the database if:

$$\text{ins_Manages}(e, d, s) \vee (\text{Manages}(e, d, s) \wedge \neg \text{del_Manages}(e, d, s))$$

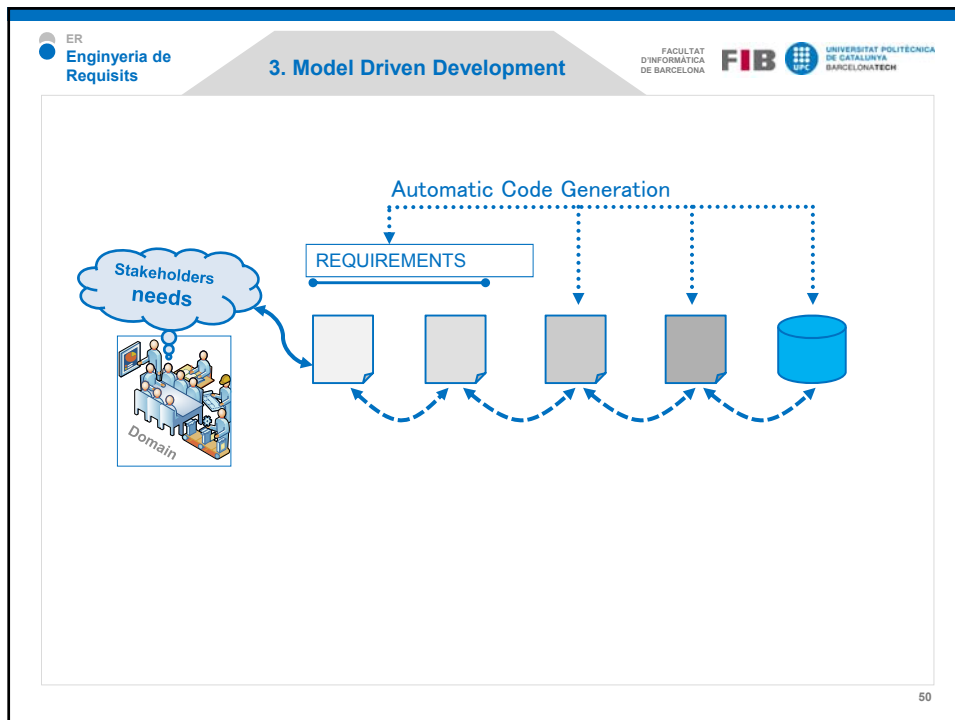
worksIn will be false in the new (updated) state of the database if:

$$\text{del_WorksIn}(e, d) \vee (\neg \text{WorksIn}(e, d) \wedge \neg \text{ins_WorksIn}(e, d))$$

Event Dependency Constraints

$$\text{ins_manages}(e, d, s) \wedge s > 500 \wedge \text{del_worksIn}(e, d) \rightarrow \perp$$
$$\text{ins_manages}(e, d, s) \wedge s > 500 \wedge \neg \text{worksIn}(e, d) \wedge \neg \text{ins_worksIn}(e, d) \rightarrow \perp$$
$$\text{manages}(e, d, s) \wedge \neg \text{del_manages}(e, d, s) \wedge s > 500 \wedge \text{del_worksIn}(e, d) \rightarrow \perp$$

47



ER
Enginyeria de Requisits

4. Metamodeling

FACULTAT D'INFORMÀTICA DE BARCELONA FIB UPC UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Deductive Rules:

$$\text{edm}(e,d,m) \leftarrow \text{worksIn}(e,d) \wedge \text{managedBy}(d,m)$$

$$\text{works}(e) \leftarrow \text{worksIn}(e,d)$$

$$\text{unemployed}(e) \leftarrow \text{labourAge}(e) \wedge \neg \text{works}(e)$$

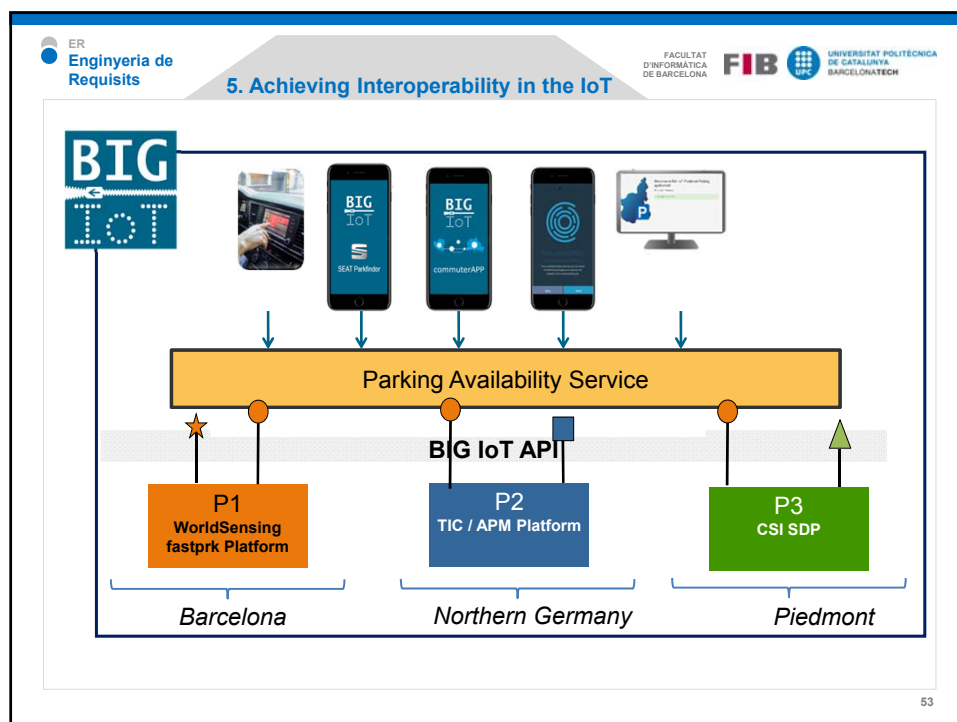
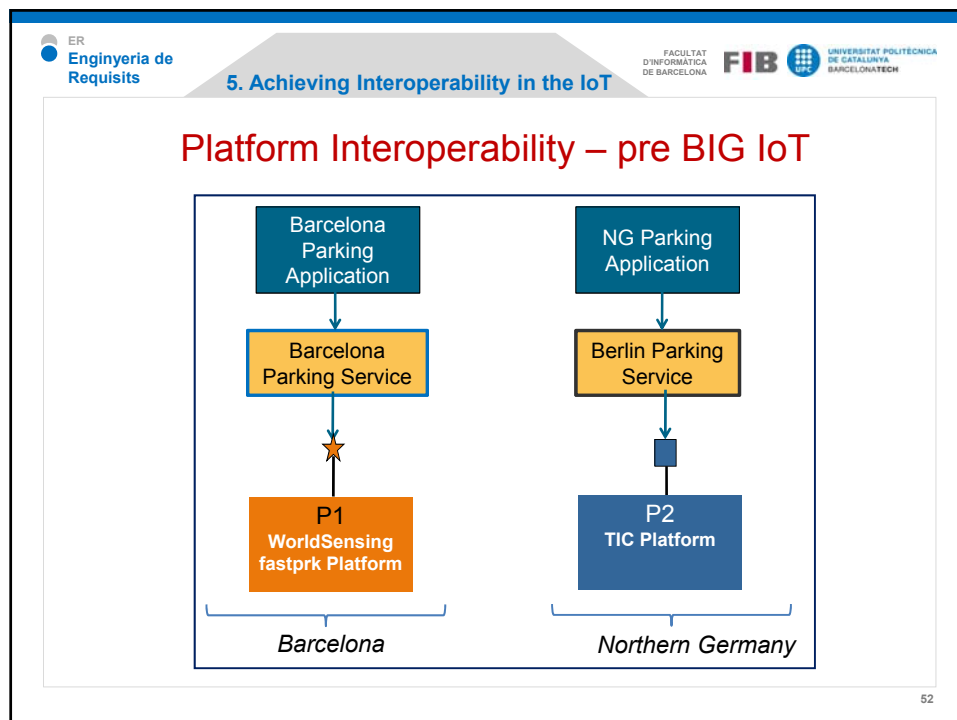
Integrity Constraints:

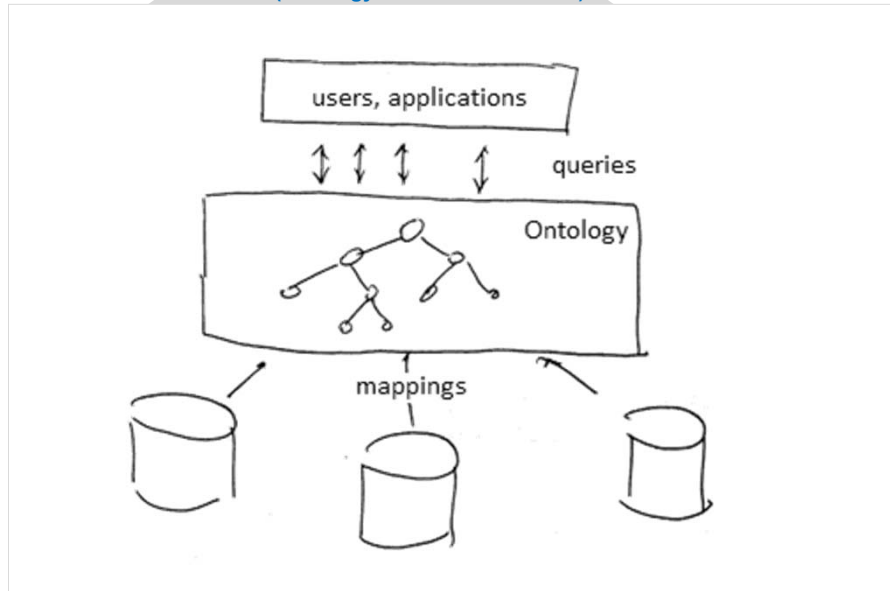
$$\text{lc1}(d,m1,m2) \leftarrow \text{managedBy}(d,m1) \wedge \text{managedBy}(d,m2) \wedge m1 \neq m2$$

$$\text{lc2}(e) \leftarrow \text{works}(e) \wedge \neg \text{labourAge}(e)$$

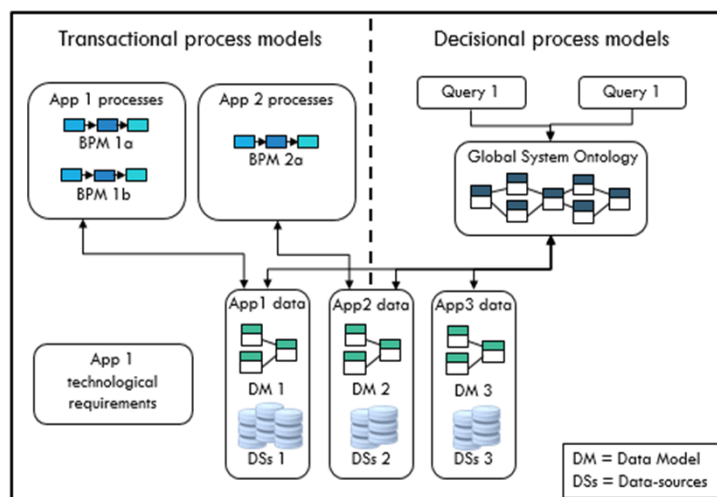
We can also define an ontology of a language!

51

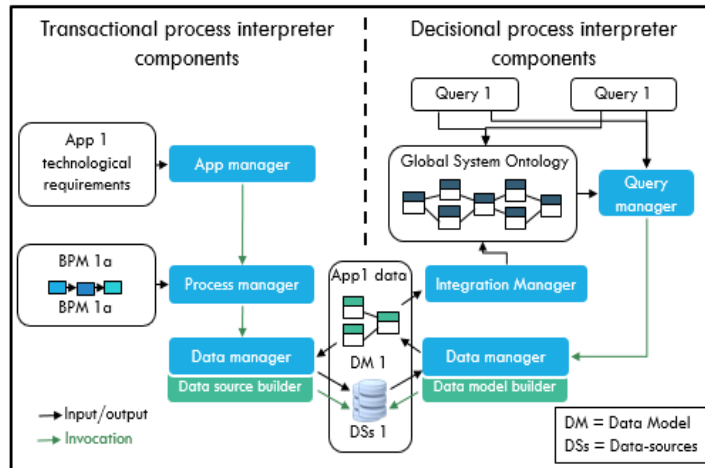




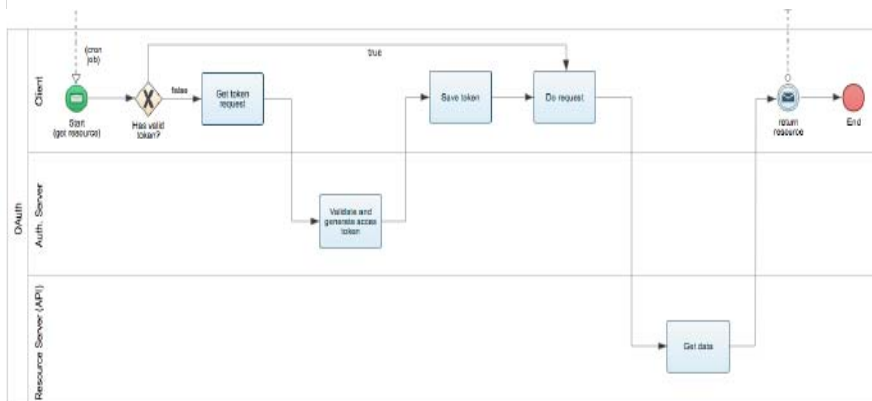
54



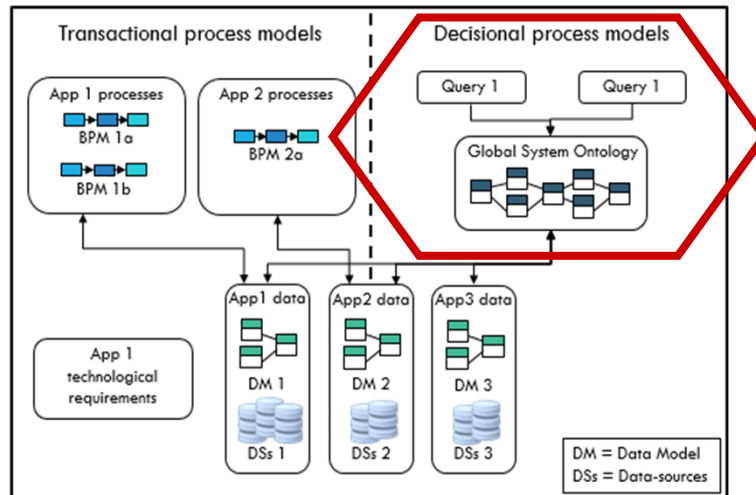
55



56



57



58

Thank you

59