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

Ontology

Introduction

Definition. (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)*"*
>
> Manera de comunicar-nos ("*Esperanto*")

Every field creates ontologies to limit complexity and organize information into data and knowledge.

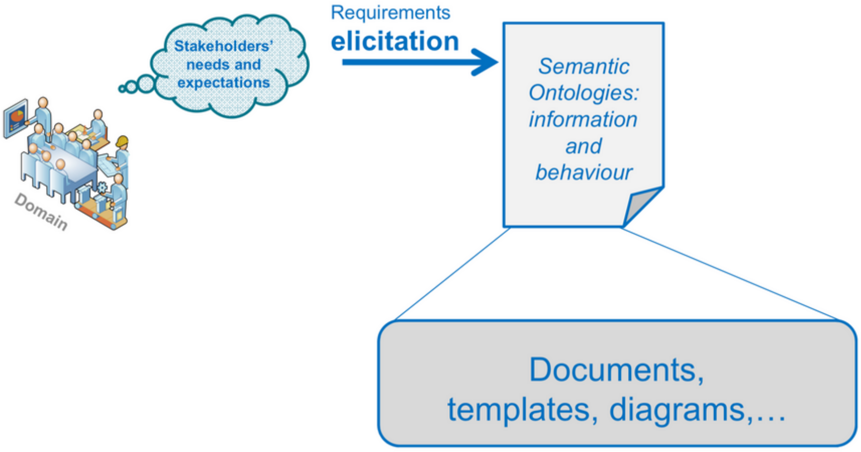
Having a common vocabulary **facilitates problem understanding and solution reuse**.

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.

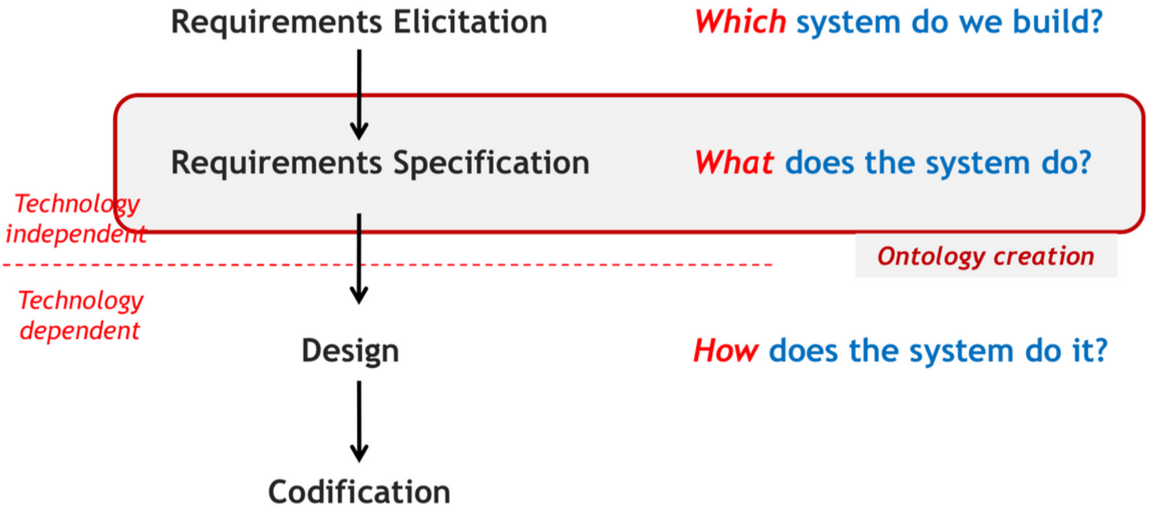
There are several languages for specifying **ontologies**:

- *Description Logics, OWL*
- *RDF, RDFS, Jason*
- *HL7*
- *Schema.org*
- *UML, OCL*

Semantic Model of the Domain


Stakeholders' needs and expectations
Domain
Requirements
elicitation
Semantic Ontologies: information and behaviour
Documents, templates, diagrams,…

Stages of Software Development



Requirements Elicitation

Which system do we build?

Requirements Specification

What does the system do?

Technology independent

Technology dependent

Ontology creation

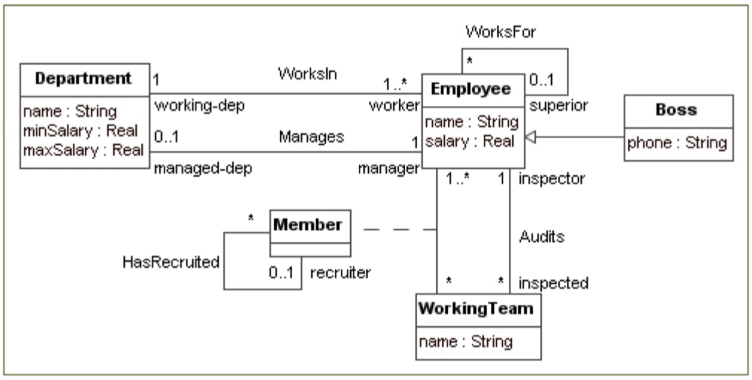
Design

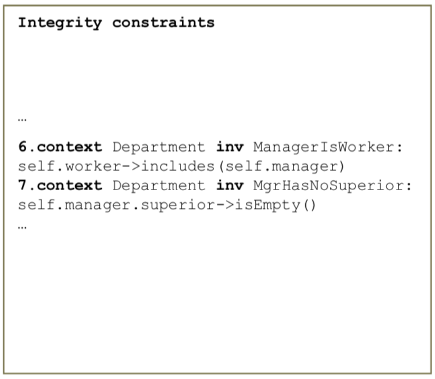
How does the system do it?

Codification

1. Generating Test Data

Give me a sample database where an Employee works for himself



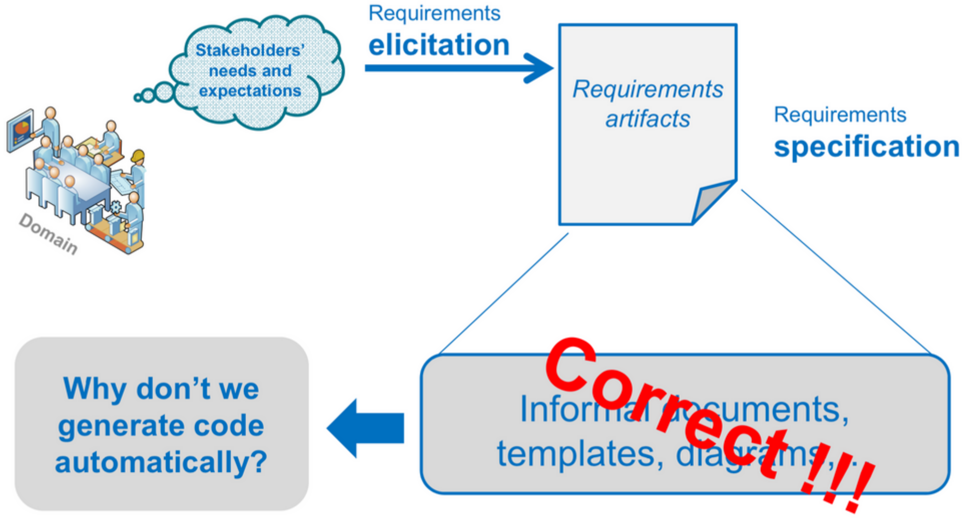


```
Integrity constraints

…

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)

2. Automatic Code Generation



Stakeholders' needs and expectations

Domain

Requirements elicitation

Requirements artifacts

Requirements specification

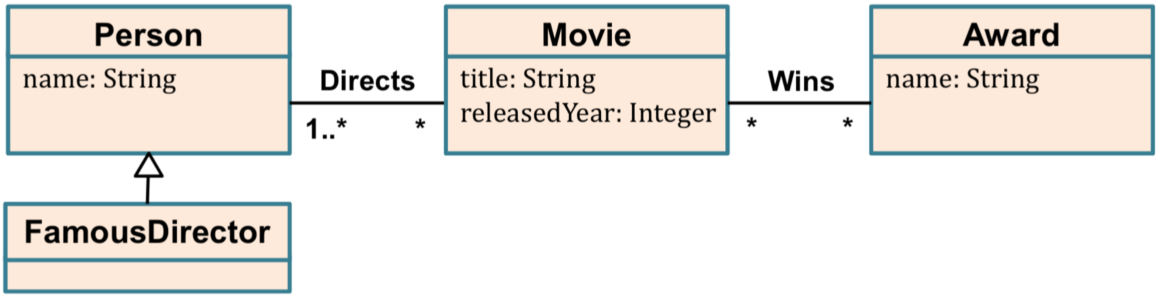
Informal documents, templates, diagrams, ...

Correct !!!

Why don't we generate code automatically?

TINTIN: Incremental Integrity Checking of SQL Assertions

Motivating Example.



Assume the constraint.

Any famous director has directed some award-winning movie

directed
won
Steven Spielberg
Jurassic Park (I)
Visual Effects

How can we check this constraint?

1. **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

2. **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!

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**

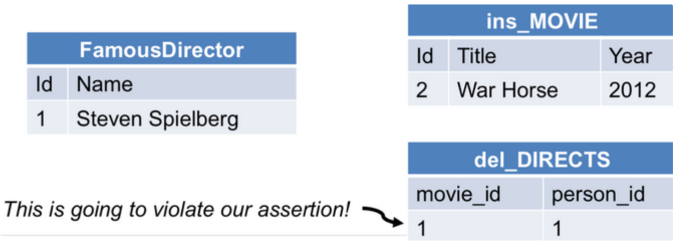
TINTIN steps

1. Connect TINTIN to your SQL Server DB

2. Write your assertions into TINTIN

3. Use your DB normally. Just recall to call `safeCommit()` at the end of your transactions

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



FamousDirector	
Id	Name
1	Steven Spielberg

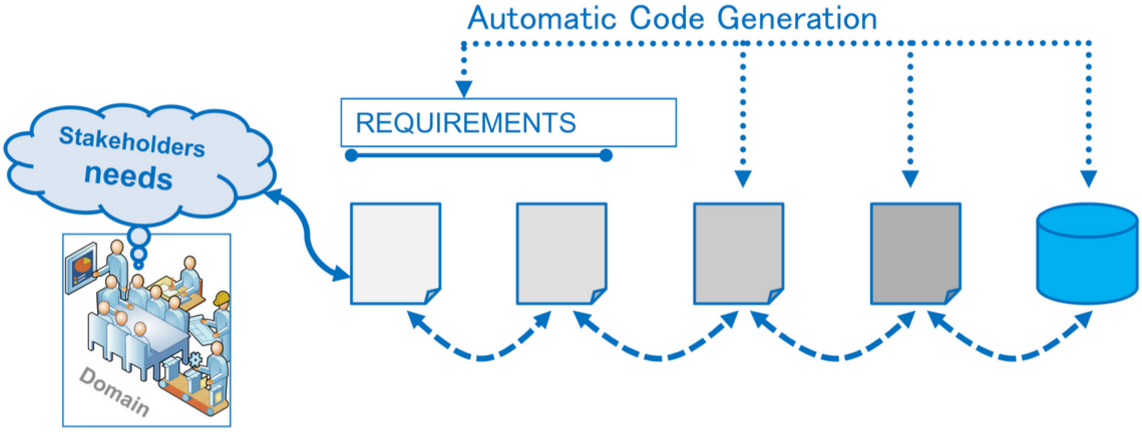
ins_MOVIE		
Id	Title	Year
2	War Horse	2012

del_DIRECTS	
movie_id	person_id
1	1

This is going to violate our assertion!

   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.

3. Model Driven Development



4. Metamodeling

We can also define an ontology of a language!

Deductive Rules:

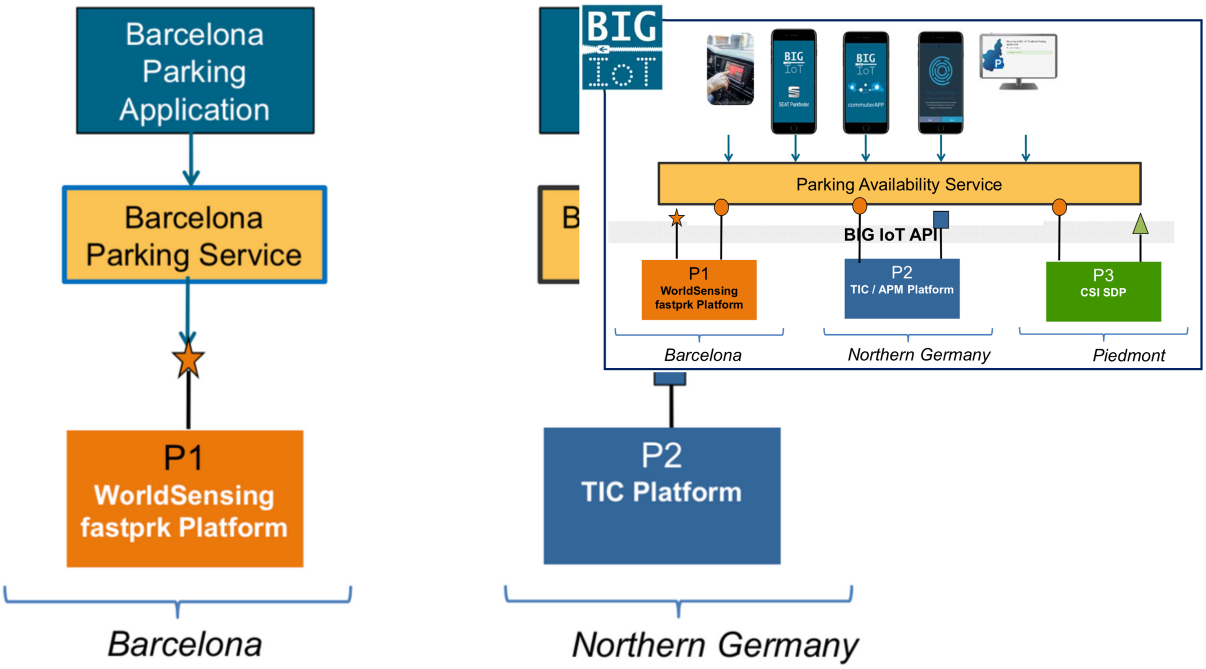
- edm(e,d,m) ← worksIn (e,d) ∧ managedBy (d,m)
- works(e) ← worksIn (e,d)
- unemployed(e) ← labourAge (e) ∧ ¬ works (e)

Integrity Constraints:

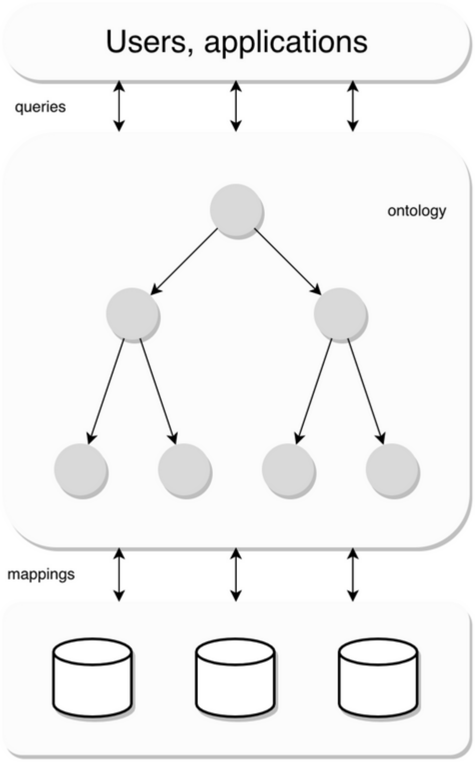
- Ic1(d,m1,m2) ← managedBy (d,m1) ∧ managedBy (d,m2) ∧ m1 ≠ m2
- Ic2(e) ← works (e) ∧ ¬ labourAge (e)

5. Achieving Interoperability in the IoT

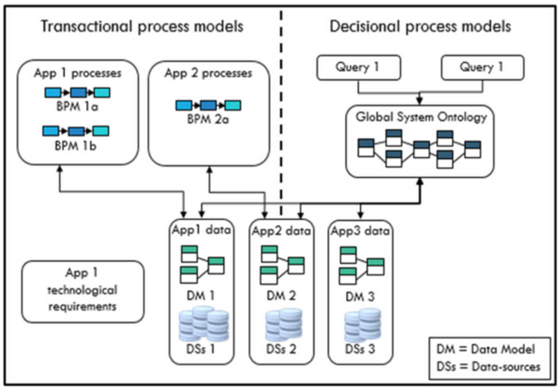
Platform Interoperability – pre BIG IoT

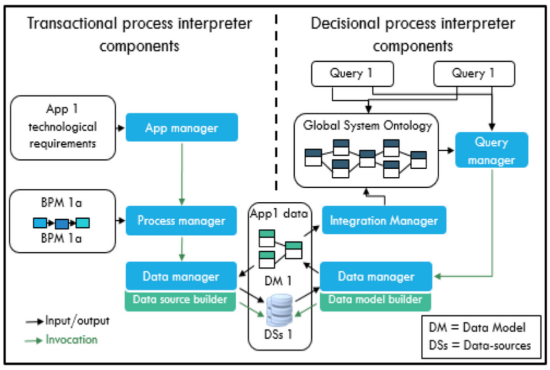

Barcelona Parking Application
Barcelona Parking Service
P1
WorldSensing fastprk Platform
Barcelona
B
BIG IoT
Parking Availability Service
BIG IoT API
P1
WorldSensing fastprk Platform
P2
TIC / APM Platform
P3
CSI SDP
Barcelona
Northern Germany
Piedmont
P2
TIC Platform
Northern Germany

6. OBDA (Ontology Based Data-Access)

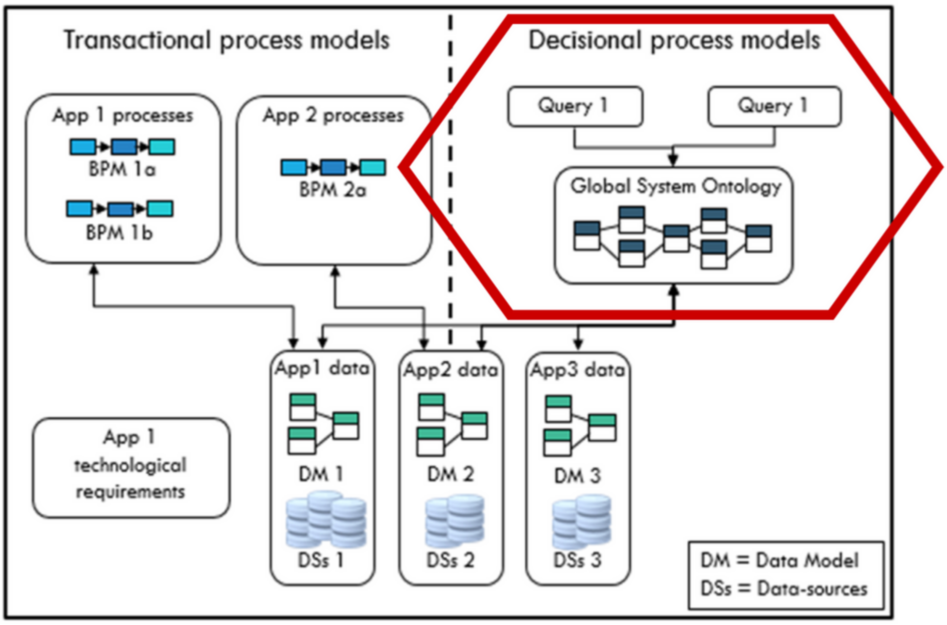


7. Automatic Software Execution (our vision)





8. Data Analytics


Transactional process models
Decisional process models
App 1 processes
BPM 1a
BPM 1b
App 2 processes
BPM 2a
Query 1
Query 1
Global System Ontology
App 1 technological requirements
App1 data
DM 1
DSs 1
App2 data
DM 2
DSs 2
App3 data
DM 3
DSs 3
DM = Data Model
DSs = Data-sources