Large-Scale and Multi-Structured Databases

Recap Software Engineering

Prof Pietro Ducange

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

UNIVERSITÀ DI PISA

CROSSLAB Innovation for industry 4.0

Objective of this Class

- Let the student to think and remember his/her skills software engineering
- We will recall some basics elements of UML.
- To use a practical example for introducing the basics of software engineering.
- To involve students in group practical activities.

From Requirements to the Application

Whenever we are asked to design and develop a specific software application, a set of steps must be performed:

1) ***Requirements elicitation*** from customer

2) ***Requirements definition***, both functional and non functional

3) ***Use case*** definition (better if with diagrams)

4) Identification of the main data structures (including the main procedures) and of the relations among them (***Analysis Workflow/Data Modelling***)

5) Refinement of 4) (***Project Workflow, including DB design***)

6) ***Implementation*** and ***test***

7) ***Deploy***

Agile Approach

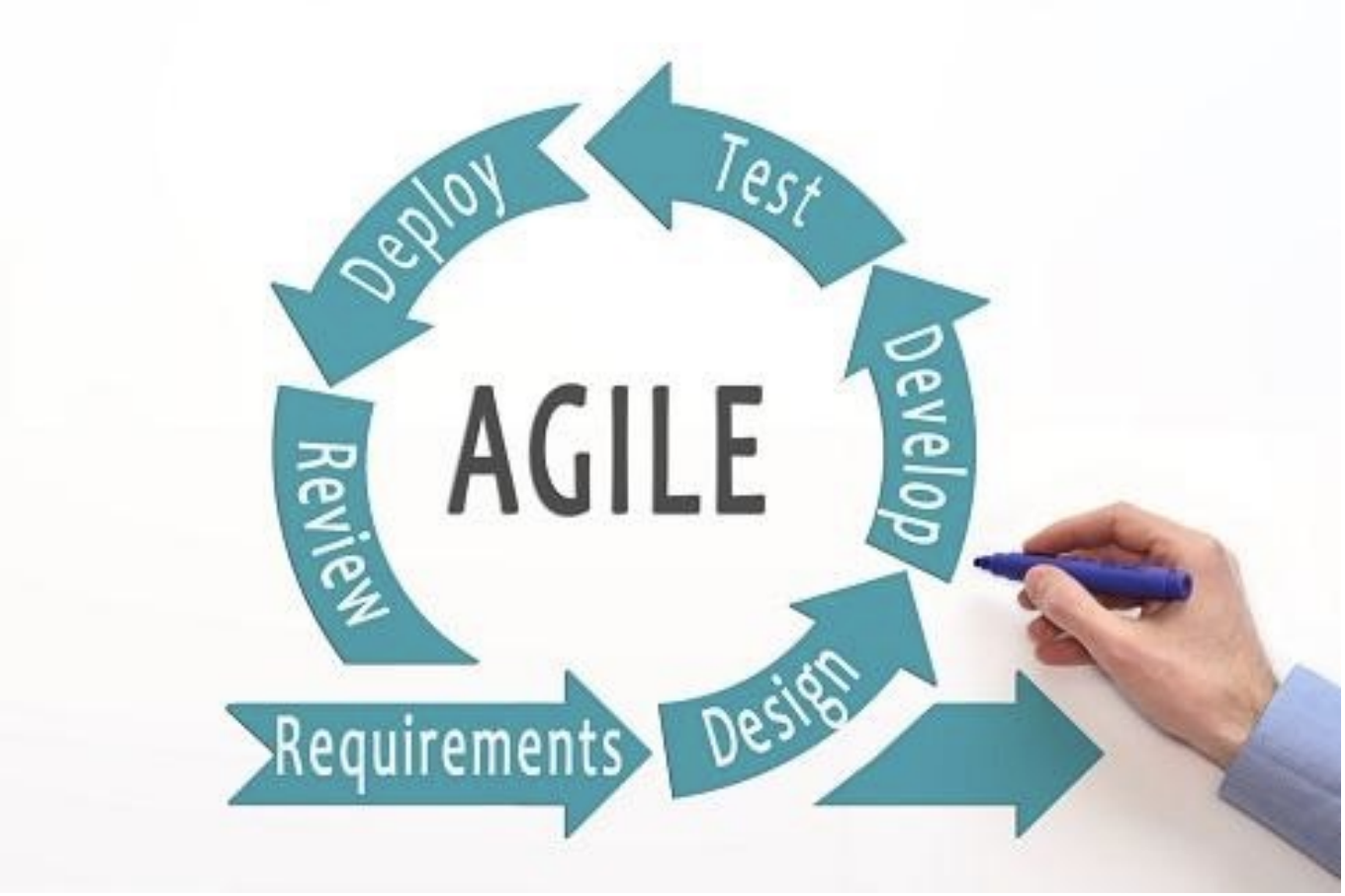


Image extracted from: "https://www.informationweek.com/youre-doing-agile-so-what/a/d-id/1329239

Role of the Architect/Engineer

To Identify the ***most suitable***:

1) Software/Hardware ***architecture***

2) Programming languages and development environments

3) Database management systems

Everything must be ***driven*** by:

1) Requirements

2) Common sense

3) Experience

Functional requirements

- Describe ***the main functionalities*** of the application
- Depend on the ***type of software***, ***expected users*** and the ***type of system*** where the software is used
- ***Functional*** requirements may ***be high-level statements*** of what the system should do.
- Functional requirements can define ***if/then behaviours.***

Examples of functional requirements

- The user shall be able to search either all of the initial set of databases or select a subset from it.
- The system shall provide appropriate viewers for the user to read documents in the document store.
- Every order shall be allocated a unique identifier which the user shall be able to copy to the account's permanent storage area.

Non-functional requirements

- Define system ***properties*** and ***constraints*** e.g. reliability, response time and storage requirements.

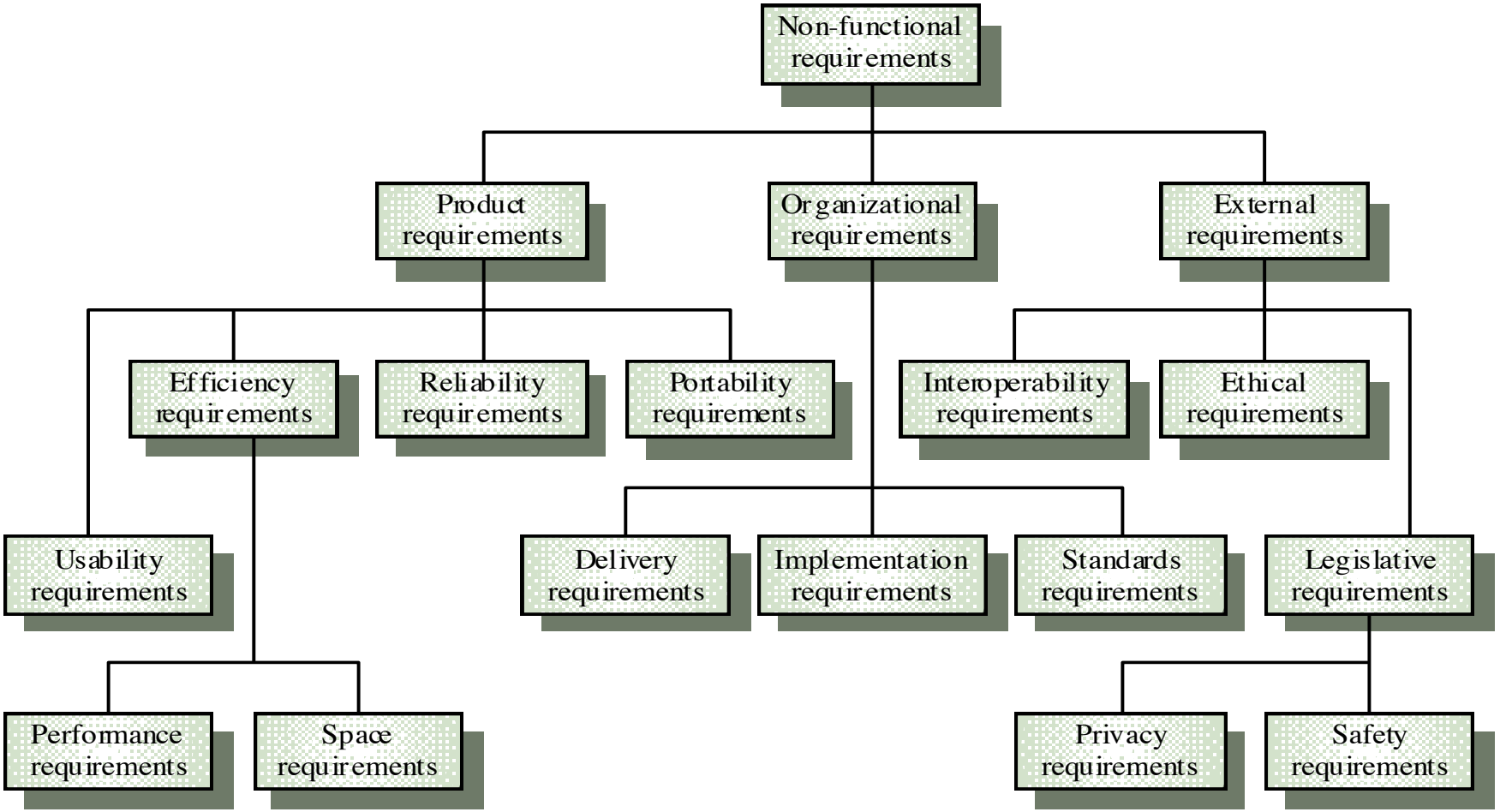
- They may regards also programming language or development method.

- Non-functional requirements may be ***more critical than*** functional requirements. If these are not met, the system is useless.

Non-functional classifications

- ***Product requirements***
  - Requirements which specify that the delivered product must behave in a particular way e.g. execution speed, reliability, etc.
- ***Organisational requirements***
  - Requirements which are a consequence of organisational policies and procedures e.g. process standards used, implementation requirements, etc.
- ***External requirements***
  - Requirements which arise from factors which are external to the system and its development process e.g. interoperability requirements, legislative requirements, etc.

Non-functional requirement types



Use Cases

They represent a ***formal way*** of describing how a system interacts with its environment.

A use case Illustrates the ***activities*** that are performed by the users of the system.

Use cases are ***sequence of actions*** a system performs that yields a ***valuable result*** for a particular actor.

A ***scenario-based*** technique in the UML.

Use Cases Diagrams

- **Use case diagrams** describe what a system does from the standpoint of an external observer. The emphasis is on *what* a system does rather than *how.*

- Use case diagrams are closely connected to scenarios.

- A **scenario** is an example of what happens when someone interacts with the system.

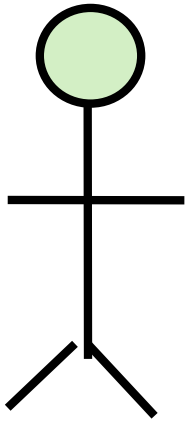
Use Case Analysis

- What is an ***Actor***?
  - A ***user*** or ***outside system*** that interacts with the system being designed in order to obtain some value from that interaction

- Use Cases describe the interaction between users of the system (the actor) and the system itself.

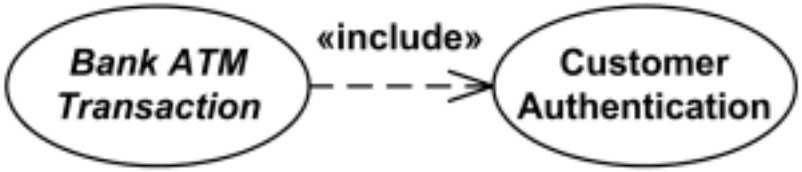
Use Cases Diagrams: Actors

- An actor can be a:
  - Human
  - Peripheral device (hardware)
  - External system or subsystem
  - Time or time-based event
- Represented by stick figure



Include and Extend

A ***base use case is dependent*** on the ***included*** use case(s); without it/them the base use case is incomplete as the included use case(s) represent sub-sequences of the interaction that must always happen .



The extending use case is dependent on the base use case; it literally extends the behavior described by the base use case. The base use case should be a fully functional use case in its own right without the extending use case's additional functionality.

Data Modelling (I)

- Data models provide a ***structured*** and ***formal*** description of the ***data*** and data ***relationships*** required for an information system.

- ***Entity Sets*** and ***Relationship Sets*** do not depend on the type of Data Base and DBMS.

- Data and data relationships will ***remain stable*** from the users' perspective throughout the development and expansion of information systems.

Data Modelling (II)

The overall Data Modelling process includes ***three stages***:

- Data ***analysis***
- Designing a conceptual data model (***entity-relationship model***)
- Converting the conceptual data model into an actual ***DB*** data organization ***model***

Data Modelling: a Simple Example (I)

1. Data analysis	Order no. 112 Date: 07/14/2015
Goal	For project monitoring purposes, employees, work, and project times should periodically be logged per department.

1. Employees report to departments, with each employee being assigned to exactly one department.
2. Each project is centrally assigned a unique project number.
3. Employees can work on multiple projects simultaneously; the respective percentages of their time are logged.
4. ...

Image extracted from: "Andreas Meier, Michael Kaufmann , SQL & NoSQL databases : models, languages, consistency options and architectures for big data management, 2019"

Data Modelling: a Simple Example (IIa)

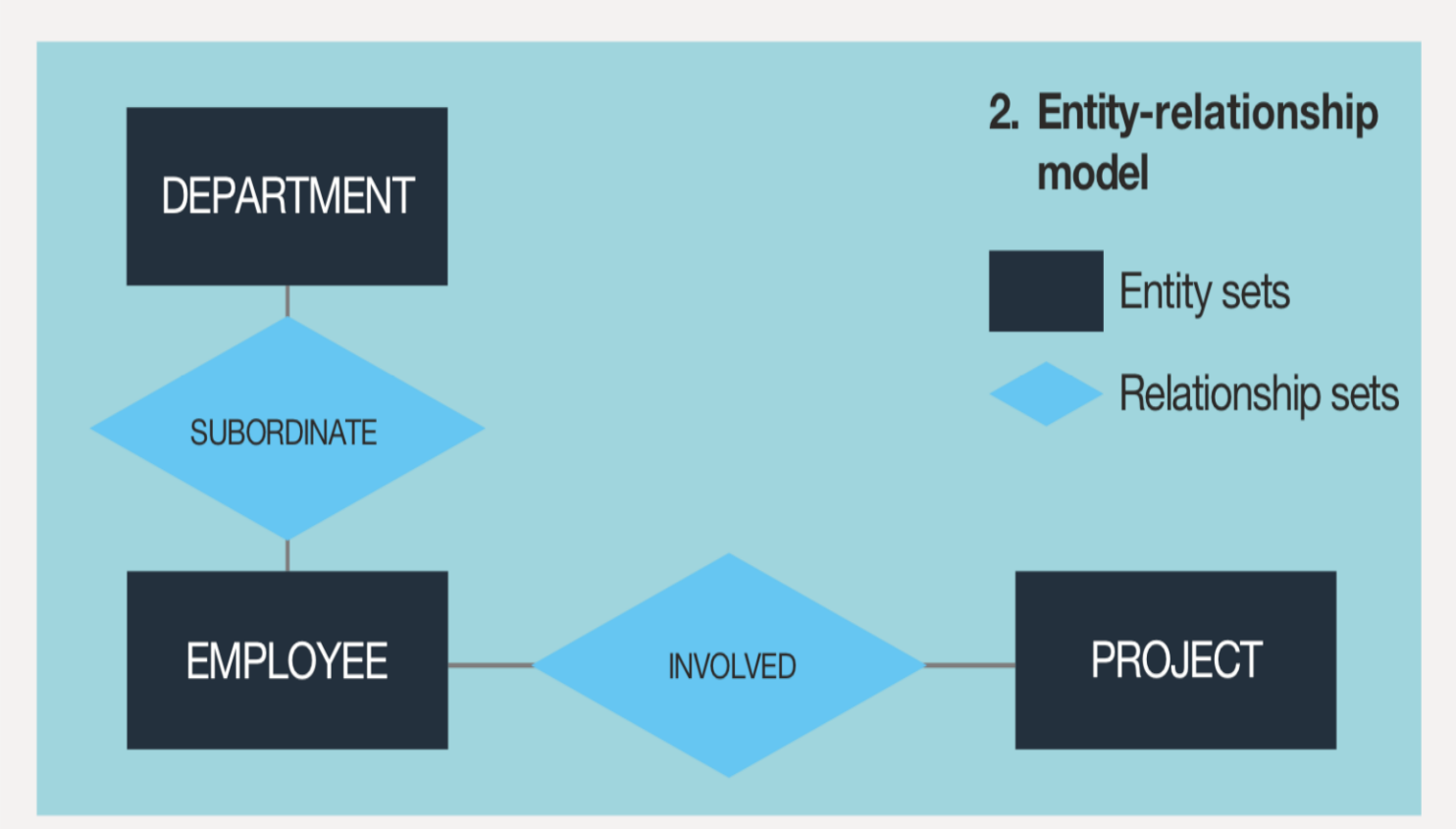
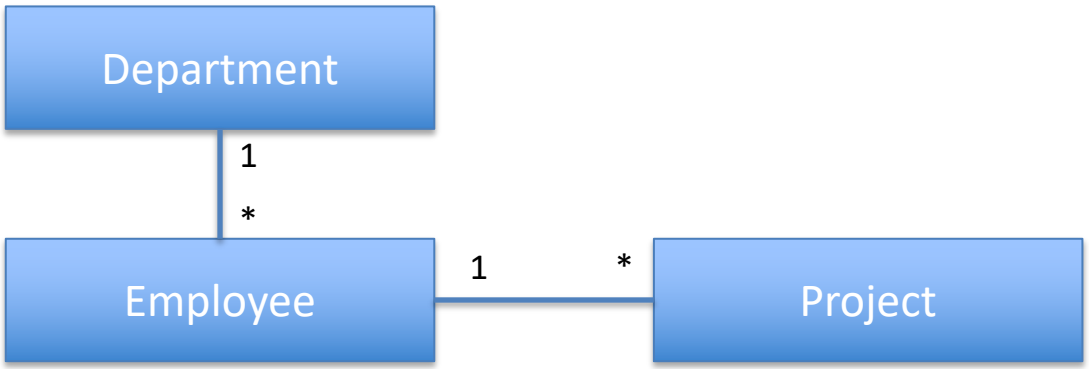


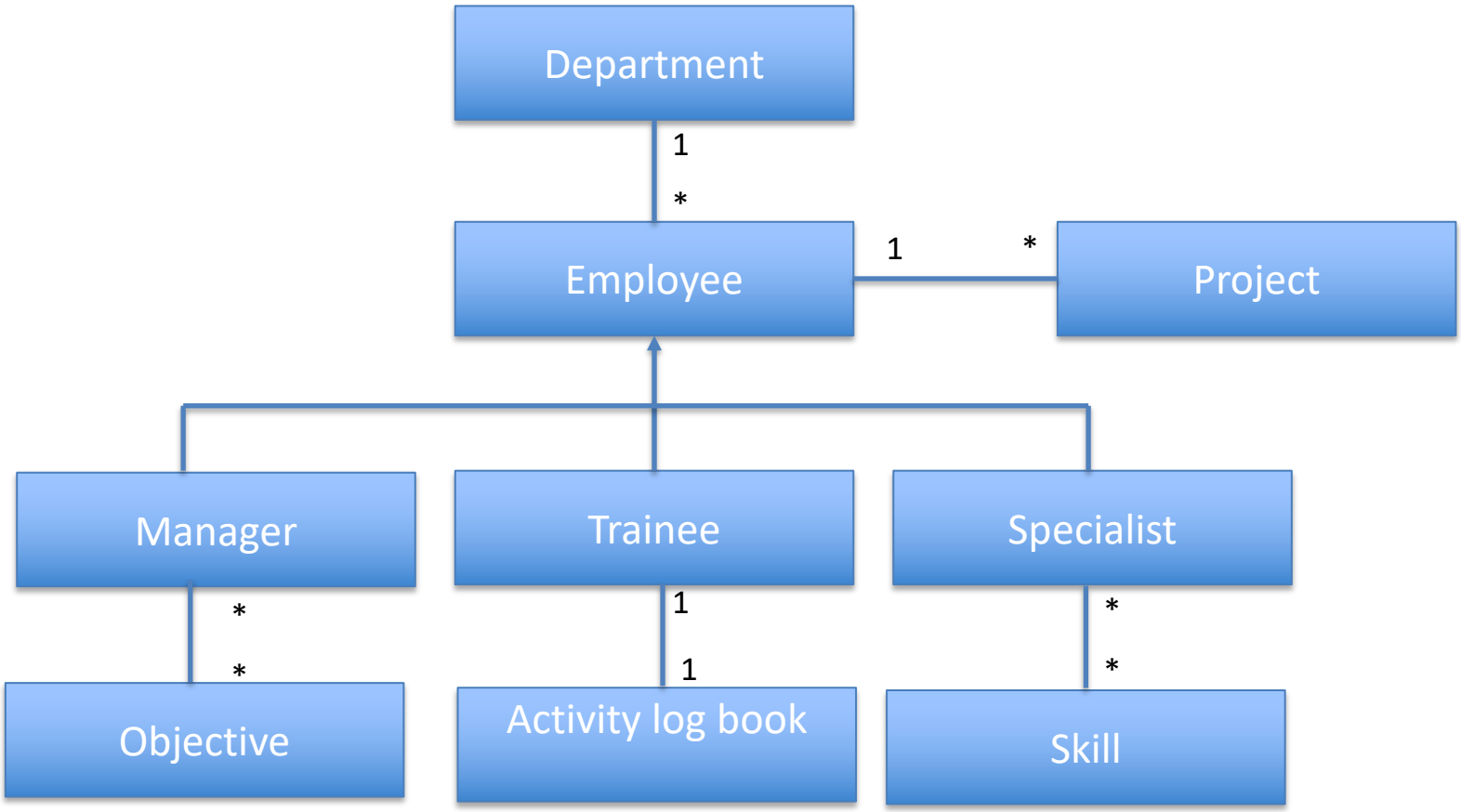
Image extracted from: "Andreas Meier, Michael Kaufmann , SQL & NoSQL databases : models, languages, consistency options and architectures for big data management, 2019"

Data Modelling: a Simple Example (IIb)



UML diagrams preferred for this course!

Data Modelling: a Simple Example (IIb)



UML diagrams preferred for this course!

Data Modelling: a Simple Example (III)

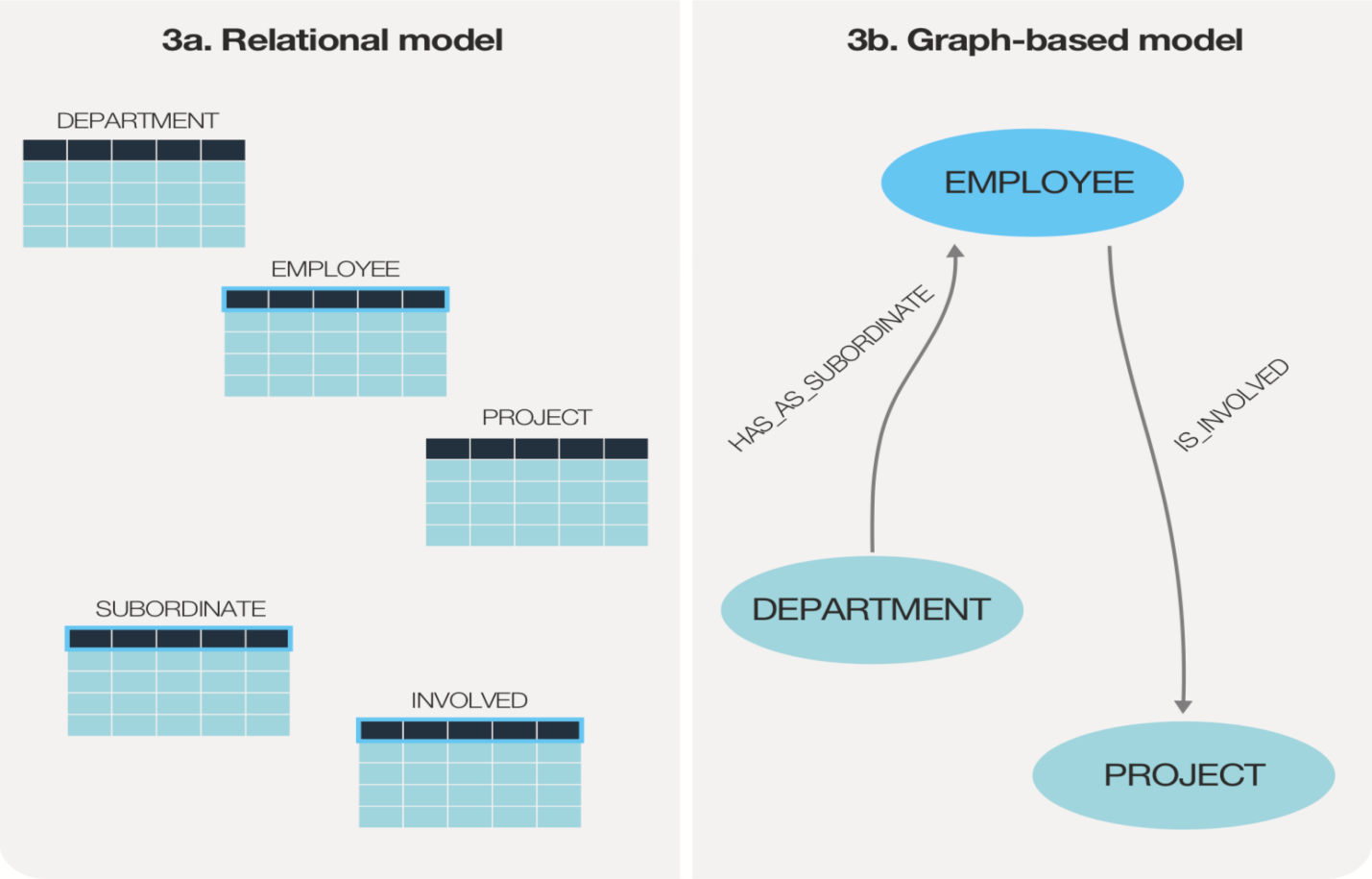


Image extracted from: "Andreas Meier, Michael Kaufmann , SQL & NoSQL databases : models, languages, consistency options and architectures for big data management, 2019"

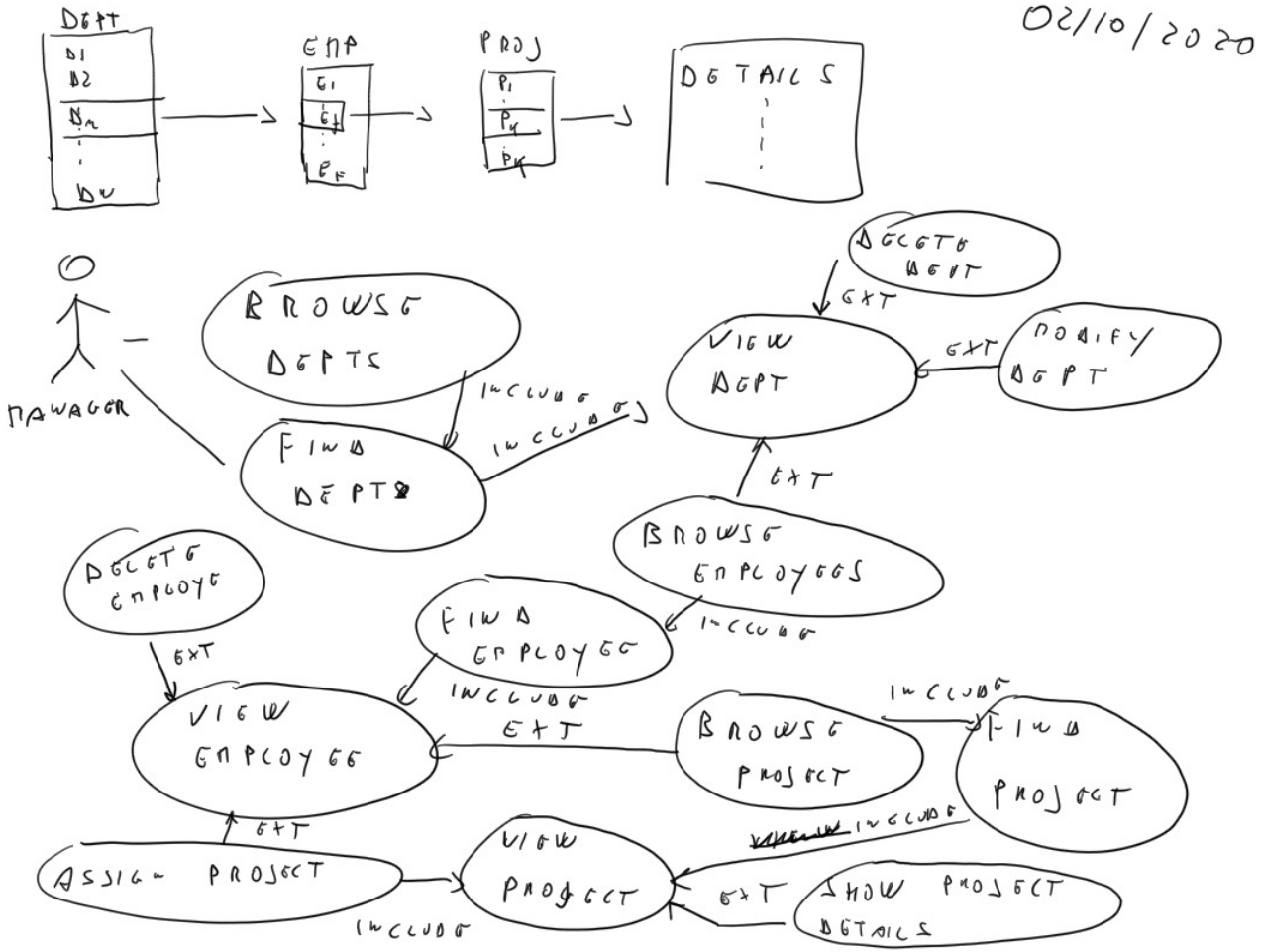
Use Case Diagrams: A simple Example

Consider the following requirements:

1. A Manager can browse the list of Departments
2. A Manager can find a Department (By Name)
3. A Manager can add a new Department
4. A Manager can select and delete a Department
5. A Manager can select and modify the description of a Departments
6. Once a Department has been selected, a Manager can browse the list of its Employees
7. Once an Employee has been selected, a Manager can browse the list of his/her Projects
8. Once an Employee has been selected, a Manager can assign a Project to him/her
9. Once a Project has been selected, a Manager can view the its details

Check Figure attached to this class material.

Use Case Diagram



Suggested Readings

Jim Arlow; Ila Neustadt , ***UML 2 And The Unified Process***: Practical Object-Oriented Analysis And Design, ISBN 10: 0321321278 ISBN 13: 9780321321275, Addison-Wesley Professional, 2005