# Object Oriented Software Engineering
## Modeling with UML

**Presented By:**

**T.A. Asmaa Hamad El-saied**

**E-mail: eng.asmaa134@gmail.com**

Lab 2

FCI, Minia University

# Agenda

1. What is Modeling?
2. What is UML?
3. UML and Risks
4. Why UML?

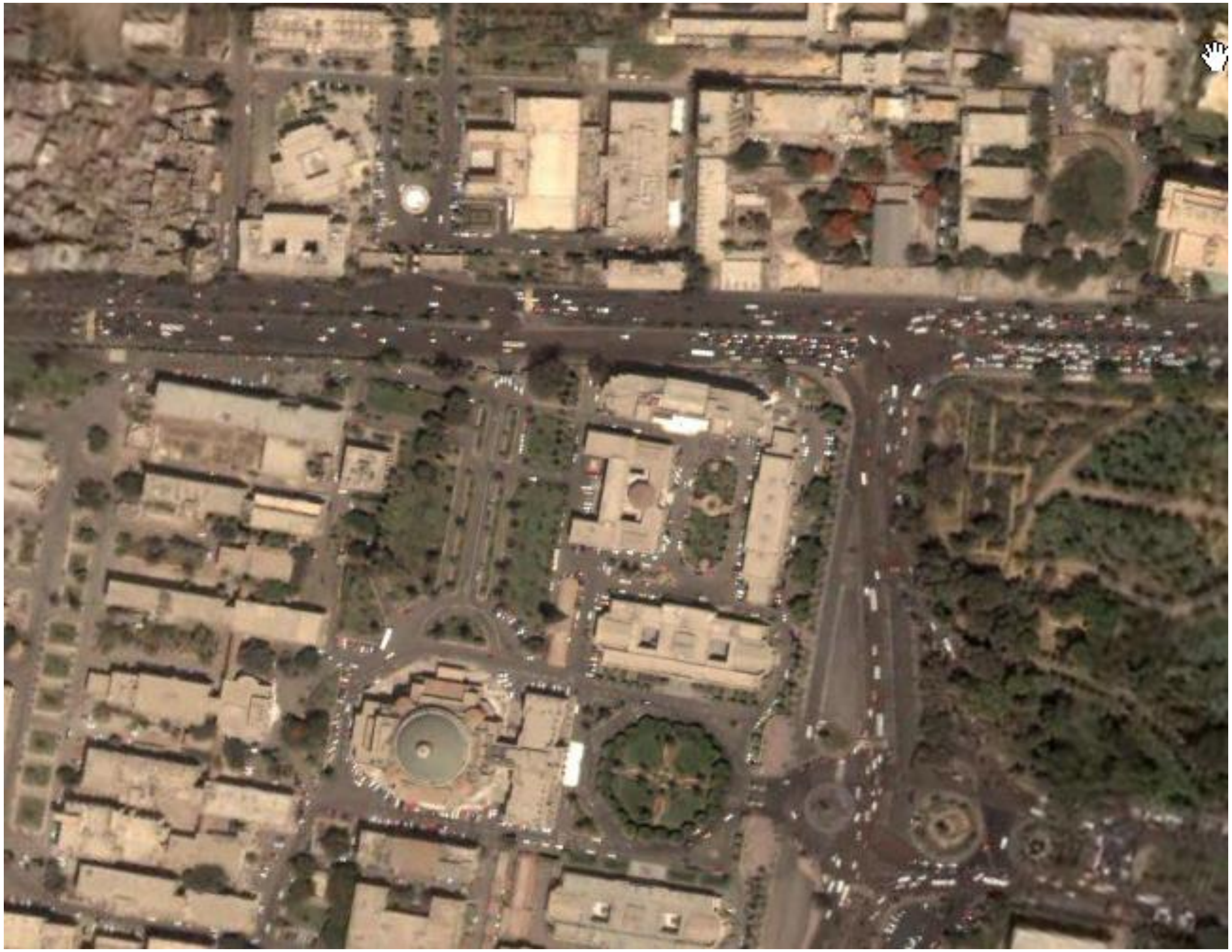
# What is Modeling?

- Modeling consists of building an abstraction of reality.

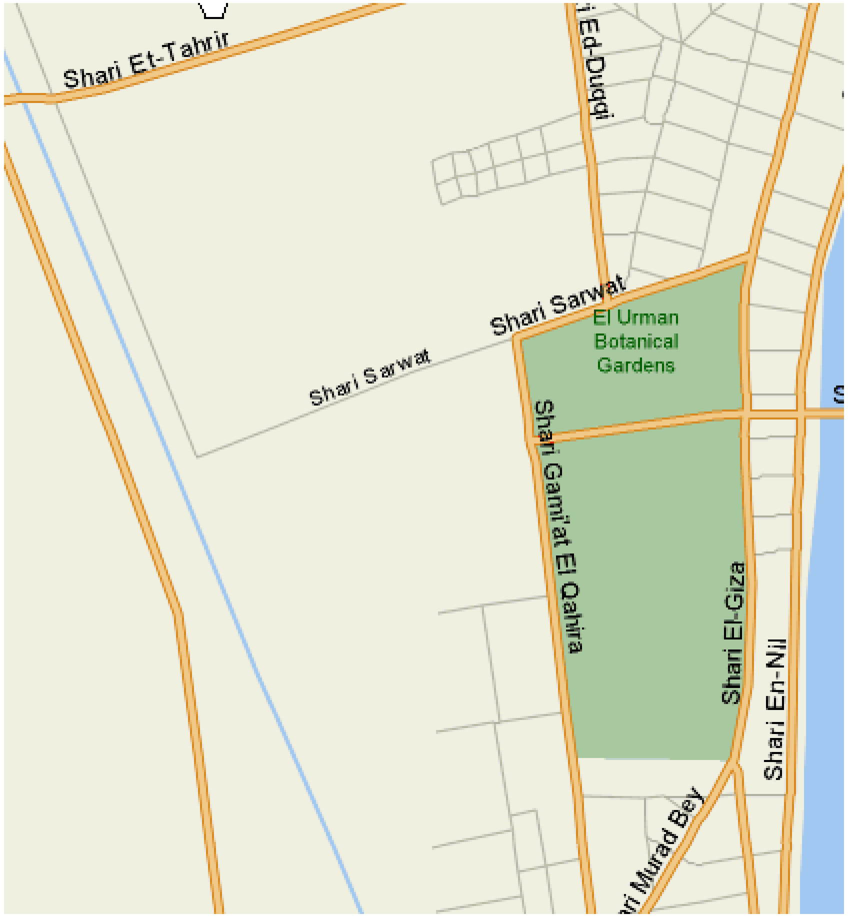
- Abstractions are simplifications because:
  - ✓ They ignore irrelevant details
  - ✓ They only represent the relevant details.

- What is *relevant* or *irrelevant* depends on the purpose of the model.
- Modeling is the designing (i.e., understanding) of software applications before coding.

# Example: street map



# Example: street map (2)



# Why model software?

- Software is getting increasingly more complex
  - ✓ Windows XP > 40 millions lines of code.
  - ✓ A single programmer cannot manage this amount of code in its entirety.

- Code is not easily understandable by developers who did not write it.

- We need simpler representations for complex systems
  - ✓ Modeling is a mean for dealing with complexity.

- Modeling is the only way to visualize the design and check it against requirements before one starts to code.

# Example

- **Q. If you want to make systems that deal with real world problems, how do you get your hands around real world complexities?**

- Ans. The key is to organize the design process in a way that clients, analysts, programmers and others involved in system development can understand and agree on. UML is a key in providing this organization.

# What is UML?

- UML (Unified Modeling Language)
  - ✓ Modeling(visual) language not a *method* for modeling and communicating about systems through the use of diagrams and supporting text.
  - ✓ An emerging standard for modeling ***object-oriented software.***
- Supported by several CASE tools
  - ✓ ArgoUML (Open Source)
  - ✓ Rational ROSE (IBM)
  - ✓ Edraw Max
  - ✓ Visio (Microsoft)

# Historical Background

- 1970 – Object-oriented modeling languages began to appear.

- 1996 – Release of UML 0.9 by by Grady Booch, Jim Rumbaugh of Rational Software Corporation, Ivar Jacobson of Objectory company.

- 1996 – Release of UML 1.0 by Digital Equipment, HP, ILogix,IntelliCorp, IBM, ICON, MCI, Microsoft, Oracle, Rational, TI and Unisys.

# Historical Background (Cont')

- 1997 – Release of UML 1.1 by IBM, ObjecTime, Platinum, Ptech, Taskon, Reich and Softeam

- 2001 – Work on UML 2.0 specifications.

# Benefits of UML

- Software systems are professionally designed and documented before they are coded so that all stakeholders know exactly what they are getting, in advance.

- Since system design comes first, UML enables re-usable code to be easily identified and coded with the highest efficiency, thus reducing software development costs.

# Benefits of UML (Cont')

- UML enables ease of maintenance by providing more effective visual representations of the system. Consequently, maintenance costs are reduced.

- UML diagrams assist in providing efficient training to new members of the development team member.

# UML and Risks

- Software Risks
  - **Requirements risks.** What are the requirements of the system? The big danger is that you will build the wrong system, one that does not do what the customer needs.
  - **Technological risks.** What are the technological risks you have to face? Are you selecting technology that will actually do the job for you? Will the various pieces fit together?

# UML and Risks (cont.)

- **Skills risks.** Can you get the staff and expertise you need?
- **Political risks.** Are there political forces that can get in the way and seriously affect your project?

# UML and Risks (cont.)

- **Dealing with Requirements Risks**
  - The starting point is use cases. Use cases drive the whole development process.
    - A use case is a typical interaction that a user has with the system in order to achieve a goal.
  - Another important task is to come up with the skeleton of a conceptual model of the domain.

# UML and Risks (cont.)

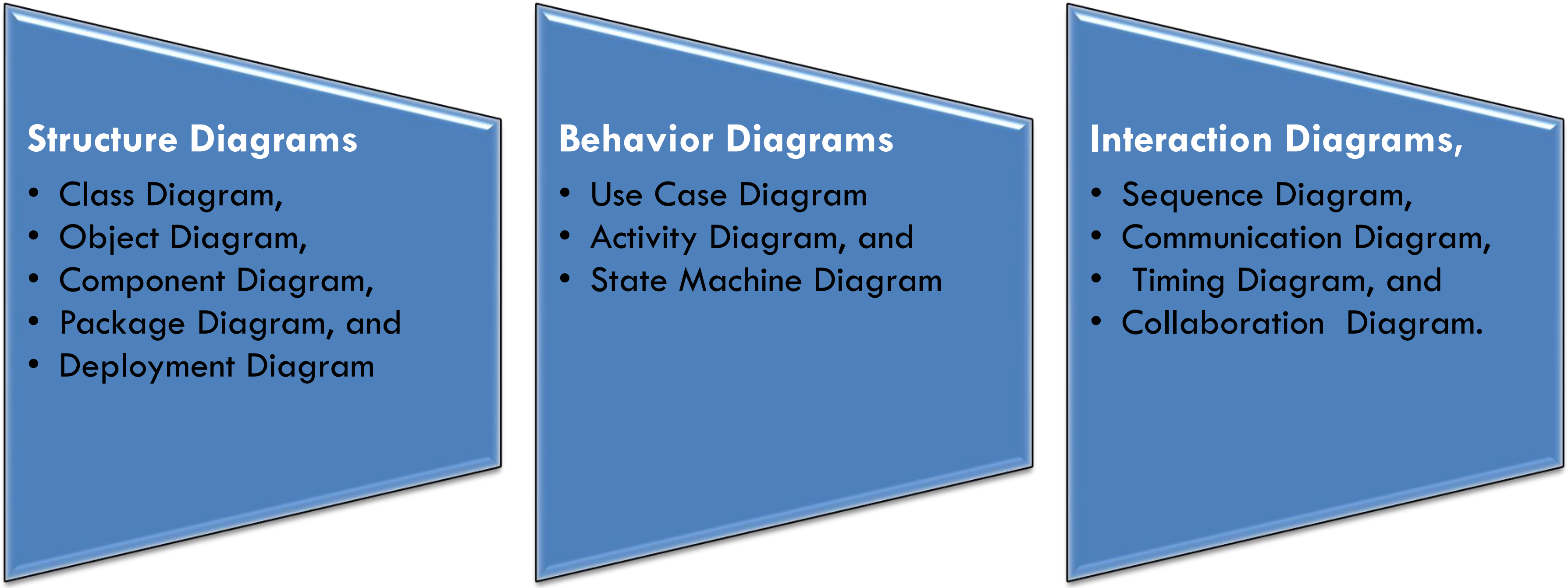
- **Dealing with Technological Risks**
  - Class diagrams and interaction diagrams are useful in showing how components communicate.
  - Package diagrams can show a high-level picture of the components at this stage.
  - Deployment diagrams can provide an overview of how pieces are distributed.

# UML and Risks (cont.)

- **Dealing with Political Risks**
  - You and your system.

# Why UML?

- UML 2.0 defines thirteen types of diagrams, divided into three categories:



**Structure Diagrams**

- Class Diagram,
- Object Diagram,
- Component Diagram,
- Package Diagram, and
- Deployment Diagram

**Behavior Diagrams**

- Use Case Diagram
- Activity Diagram, and
- State Machine Diagram

**Interaction Diagrams,**

- Sequence Diagram,
- Communication Diagram,
- Timing Diagram, and
- Collaboration Diagram.

# SRS Chapter One Clarify

- **Software Requirement Specification (SRS)** Document: to write software documentation and explanation
- **Chapter one provides an overview of the software**
  - Document purpose
  - Product scope
  - Definitions and abbreviations
  - References
  - Overview of the rest of SRS

# SRS Chapter Two Clarify

- **Chapter 2 include:**
  - Product Perspective
  - Product Functionality
  - Users Characteristics
  - Operating Environment
  - User Documentation
  - Assumptions and Dependencies

# Assignment #2

- Complete chapter 1 and 2 in your SRS.
- Send it

  eng.asmaa134@gmail.com

# Thanks