Software Engineering

Requirement Analysis

Requirements Analysis

Functional requirements:

define what a system is supposed to do

(functions of software)

Non-functional requirements:

define how a system is supposed to be

(properties or qualities or “constraints” on software)

Requirements Analysis

- **Standard approach:**
  - Use natural language specifications written with a Word Processor (e.g. MS Word)
- **Standard problems:**
  - Difficulties in understanding how the system works
  - Ambiguities/Incompleteness in the specifications
  - Requirements Management (impact of requirements, traceability, …)

Requirements Analysis

New approach:

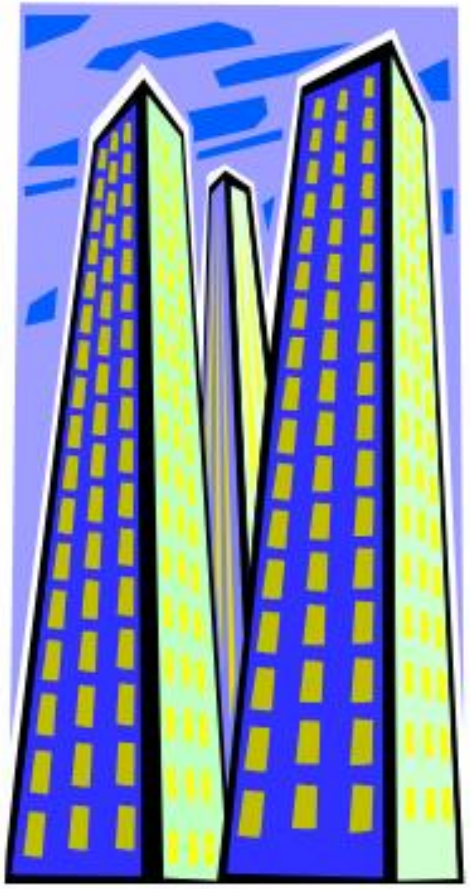
- Goal:
  - Define the requirements of the product
- Output
  - Requirements Document with UML diagrams and tables
- UML Diagrams for Functional requirements
  - Use Case Diagram and Activity Diagram
  - Sequence Diagram
- Tables for Non-functional requirements

Requirements Analysis: example

Well, maybe
you shouldn't'

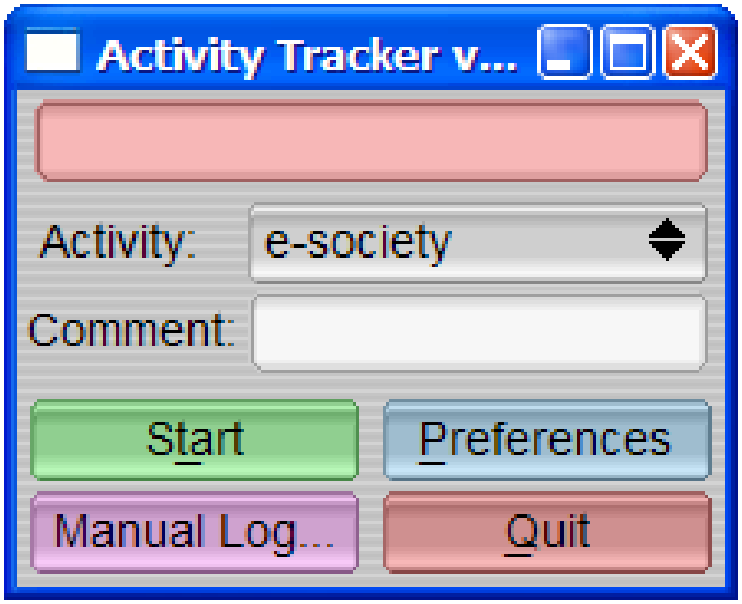
But then, maybe
you should

Maybe you
have to



Example: requirements Time Tracker

We have been contacted by a small firm. They want us to build a system for letting employees track how they spend their time when working on a computer. The idea is that of a stop-watch: the users of the system can start and stop counting the time spent on different activities; the system logs such activities and can be used to produce reports.



The system can also be integrated with a billing system. The billing system receives all the information about the time spent by programmers on the different projects and computes the cost of projects. This information is then used to charge clients.

Standard approach:

Functional requirement:

- users can start and stop counting the time spent on activities
- the system logs such activities
- logs can be used to produce reports
- …

Non-functional requirement:

- logs need to be updated in real time on a server (or not?)
- users can learn how to use this software in 30 minutes without reading documentation
- …

UML for functional requirements:

- Identify Actor:
  - Set of **roles** that users can play
- Identify Use Case:
  - Set of **scenarios** describing a coherent unit of functionality **as seen by a user of the system**.
- Make a UML - Use Case Diagram
  - A graph collecting the use cases, the actors taking place in the use cases, and the relationship among use cases and actors.
- Write Use Case Diagram by using https://draw.io/

Why UML

- Simple and intuitive
- Support whole cycle
- Widely used
- ISO standard

... and the rest of the story:

UML is not the only specification language

Examples: DFD, SDL, MSC, Statecharts, B, Z.

Why draw.io

- Good Support of UML
- Widely used
- Integrated with Google Drive and GitHub
- Free plan

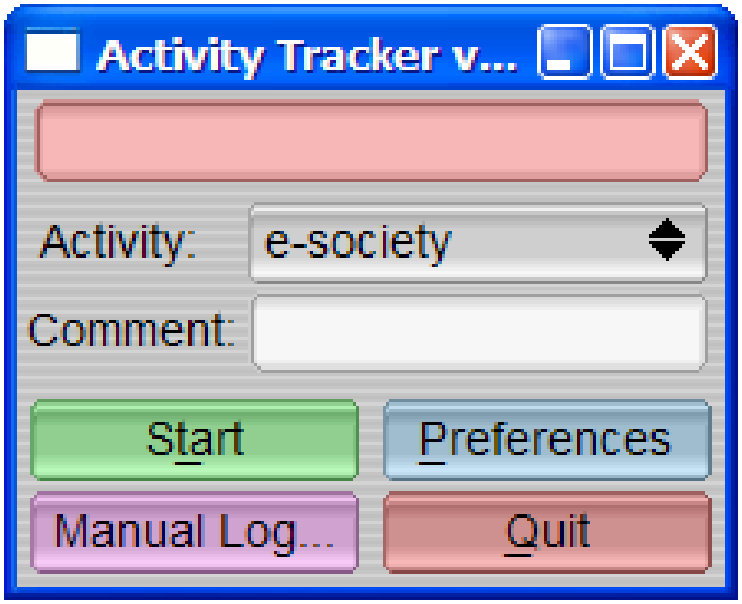
... and the rest of the story:

draw.io is not the only tool supporting UML

Examples: Lucidchart, yEd, Rational Software Architect, Enterprise Architect, Argo UML, Poseidon UML, Visio, …

Example: Time Tracker

We have been contacted by a small firm. They want us to build a system for letting employees track how they spend their time when working on a computer. The idea is that of a stop-watch: the users of the system can start and stop counting the time spent on different activities; the system logs such activities and can be used to produce reports.



The system can also be integrated with a billing system. The billing system receives all the information about the time spent by programmers on the different projects and computes the cost of projects. This information is then used to charge clients.

Step 1: Identify Actors

An actor is someone or something that must interact with the System Under Development

Step 1: Time Tracker

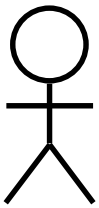
We have been contacted by a small software firm.

They want us to build a system for letting employees track how they spend their time when working on a computer. The idea is that of a stop-watch: the users of the system can start and stop counting the time spent on different activities; the system logs such activities and can be used to produce reports by an administrator.

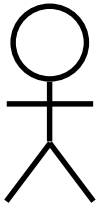
The system can also be integrated with a billing system. The billing system receives all the information about the time spent by programmers on the different projects and computes the cost of projects.

Step 1: Identify Actors

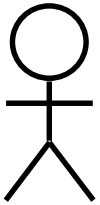
An actor is someone or something that must interact with the System Under Development



User of the system



Administrator



Billing System

Step 2: Identify Use Cases

A use case is a pattern of behavior the system exhibits

Each use case is a sequence of related transactions performed by an actor and the system in a dialogue

Strategies for identifying Use Cases

- Examine Actors to determine their needs
- Find verbs and actions in the description

Step 2: Time Tracker

We have been contacted by a small software firm.

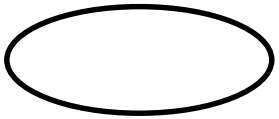
They want us to build a system for letting employees track how they spend their time when working on a computer. The idea is that of a stop-watch: the users of the system can start and stop counting the time spent on different activities; the system logs such activities and can be used to produce reports by an administrator.

The system can also be integrated with a billing system. The billing system receives all the information about the time spent by programmers on the different projects and computes the cost of projects.

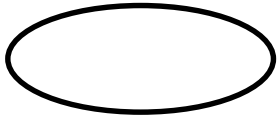
Step 2: Identify Use Cases

A use case is a pattern of behavior the system exhibits

Each use case is a sequence of related transactions performed by an actor and the system in a dialogue



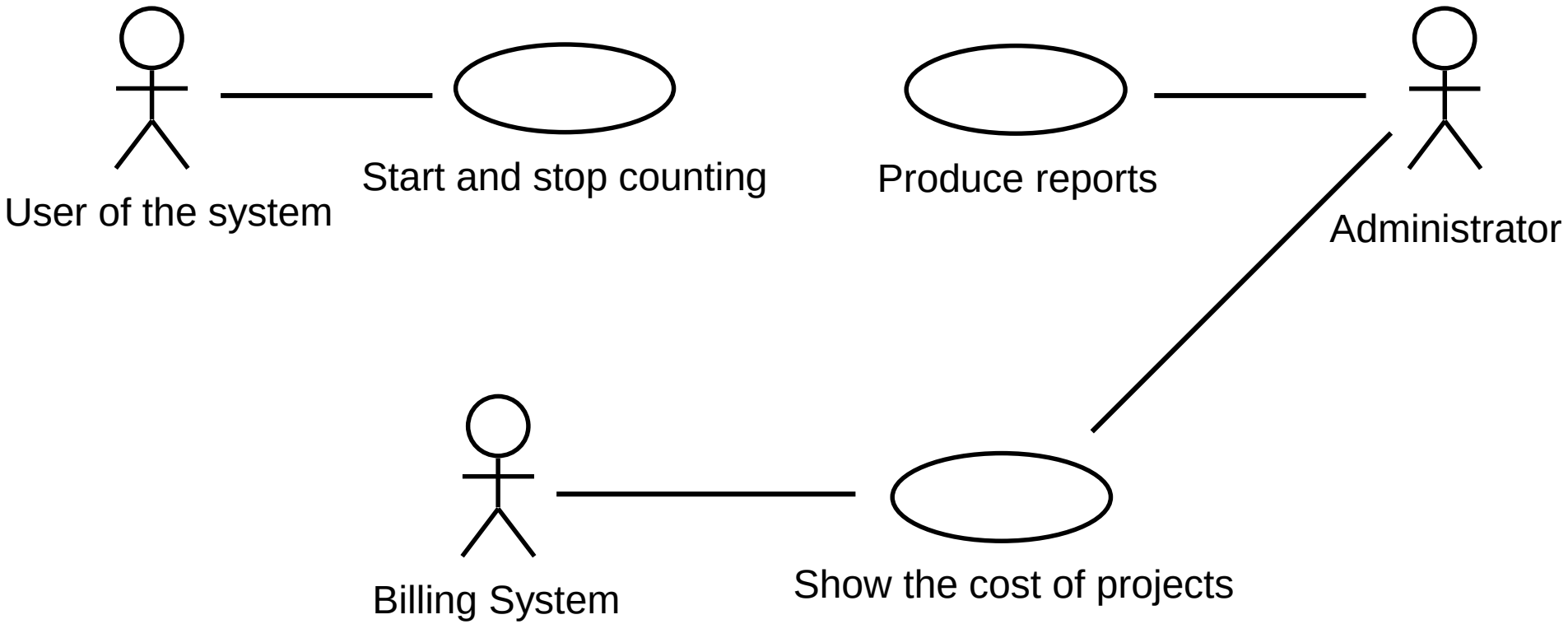
Start and stop counting



Produce reports

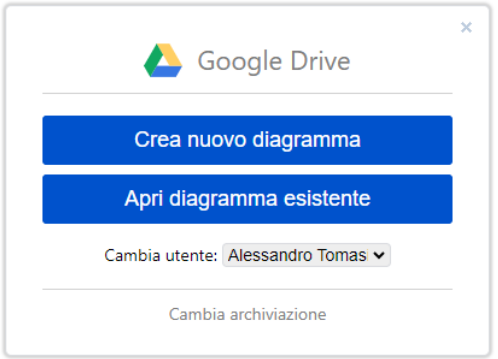
Step 3: Make Use Case Diagram

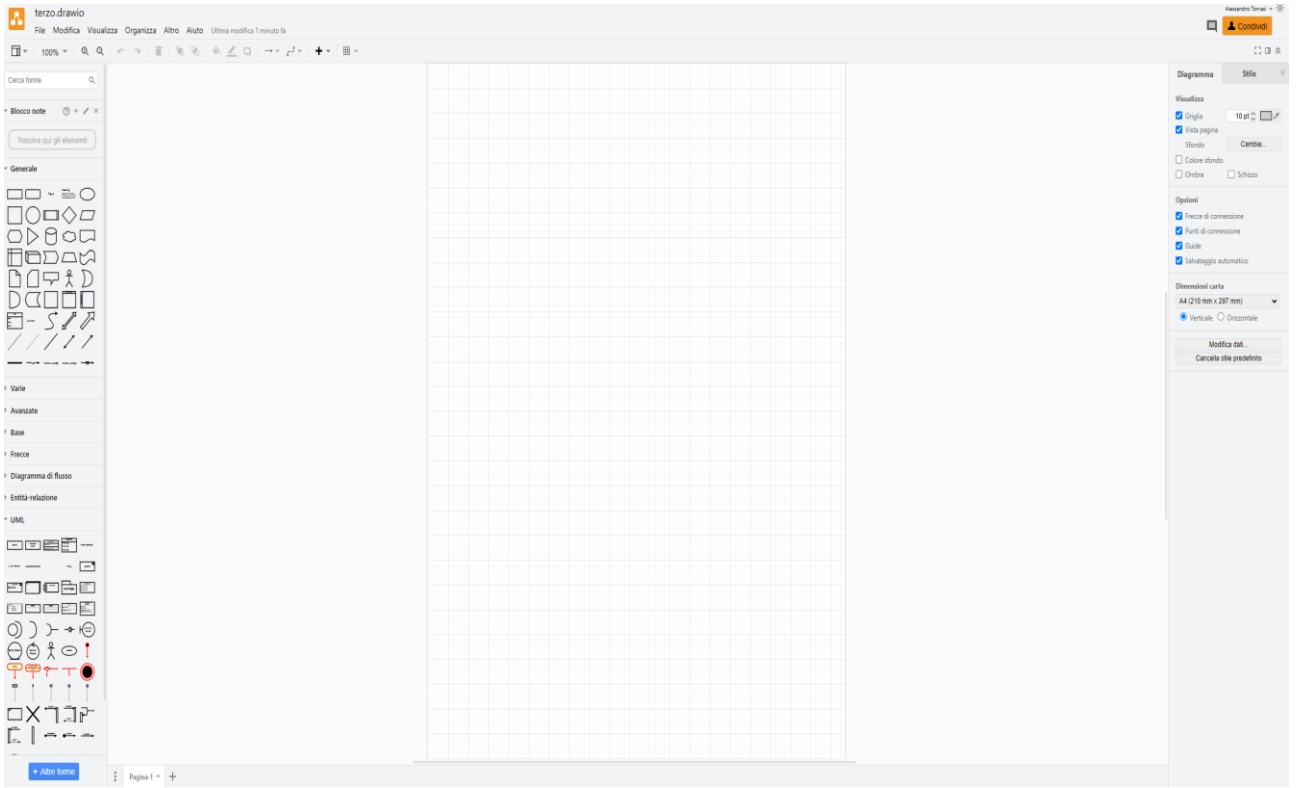
Use case diagrams are created to visualize the relationships between actors and use cases



Try to make it

- Go on: https://draw.io
- Connect to your Google Drive or your GitHub
- Click on «Crea nuovo diagramma»
- Use elements in UML





Use Case Diagram: Elements

- Actor is a coherent set of roles that users of an entity can play
  - Actors need not be human
  - Actors represent roles and not instances (e.g. a person can play different roles when using a system)
- Use Case represent a coherent unit of functionality of the system

Software Engineering

Restaurant Management System

Restaurant Management System

Restaurant Management System

We have been asked to build a system to automate the ordering and billing activities of a restaurant. The system is distributed: waiters and waitresses are provided with handheld devices to take orders. The handheld devices communicate orders to the kitchen and to the cashier. The handheld devices receive real-time information about availability of the different items in the menu. Once placed, orders can be changed by the customers, within a time frame from the order (5 minutes) or after the time-out, if the corresponding order has not yet been processed by the Cook. The system computes bills and is also used to manage reservations of tables. Reservations can either happen by phone or via the internet.