THE UNIVERSITY OF WESTERN AUSTRALIA

UML Use Case Diagrams and UML Use Cases

Software Requirements and Design

CITS4401

Week 3- Part 2

Outline

UML = the Unified Modelling Language

UML Use Case Diagram

UML Use Case

Conceptual Modelling

Conceptual Models

The development of **models of a real-world problem** is key to software requirements analysis.

Their purpose is to aid in understanding the situation in which the problem occurs, as well as depicting a solution.

Hence, conceptual models comprise models of entities from the problem domain, configured to reflect their real-world relationships and dependencies.

SWBOK c1 1 Section 4.2 Conceptual modelling

Architectural Design and Requirements Allocation

Architectural Design and Requirements Allocation

At some point, the solution architecture must be presented.

Architectural design is the point at which the requirements process overlaps with software or systems design and illustrates how impossible it is to cleanly decouple the two tasks …

because the process of analyzing and elaborating the requirements demands that the architecture/design components that will be responsible for satisfying the requirements be identified.

SWBoK Ch 1 Section 4.3

Use Case Diagrams

These describe functionality of a system from the users' point of view

They show functions that produce a visible result for an actor

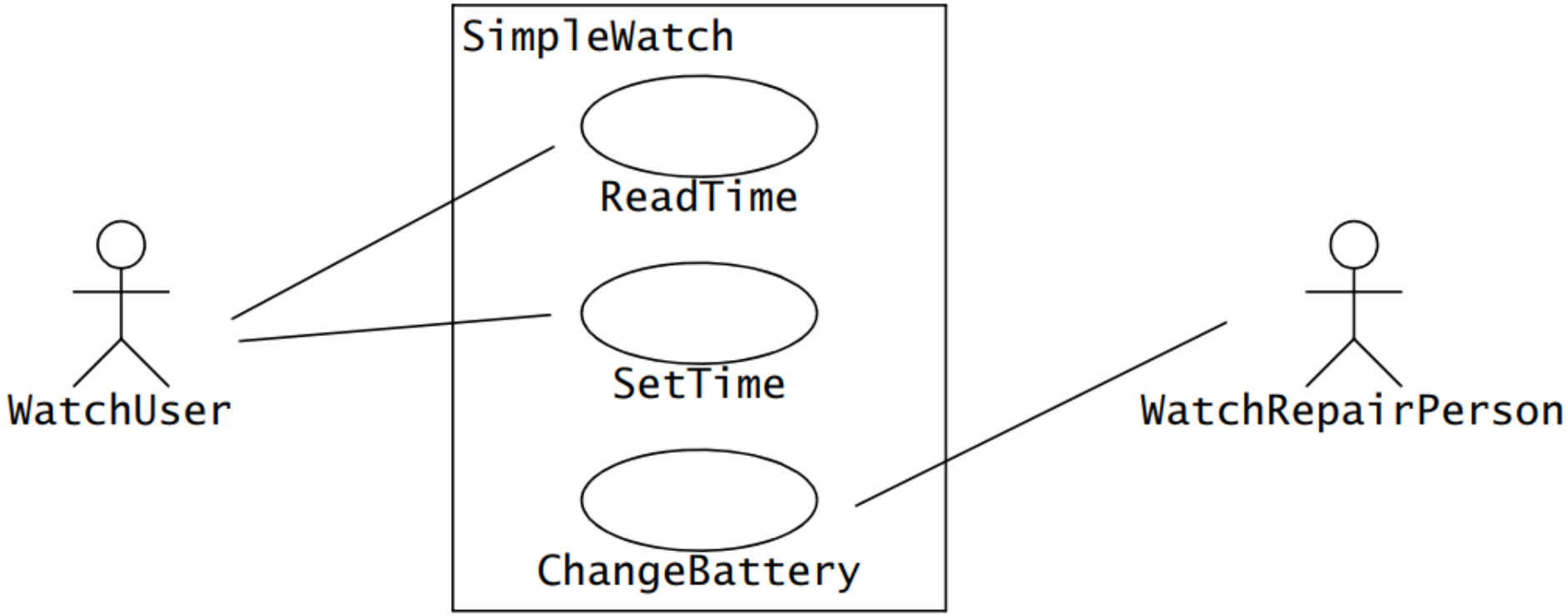
Use Case Diagrams

- UML notation to

  describe the **functionality** of a system as seen by the users in terms of

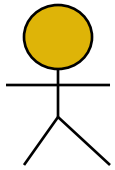
  - actors and their goals
  - the system boundary: what is in scope and out of the scope of the system?
  - the names of use cases for the system

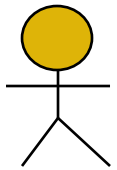
Use case diagram for a simple watch



Actors in UML

- external entities that interact with the system
- an actor can be a **user role**
  - e.g. …
- an actor can be **another system**
  - e.g. …
- actors have unique names and descriptions
- actors have **goals** that must be satisfied by the system





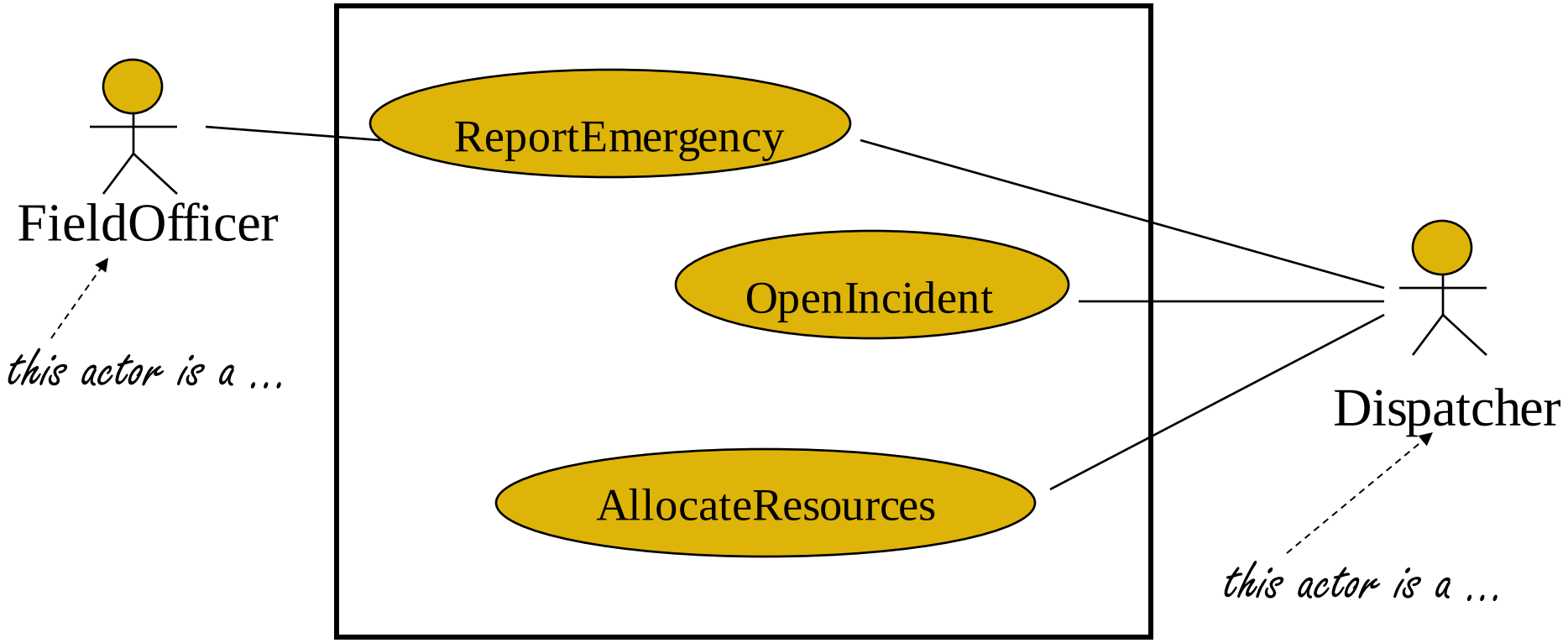
Use Case Diagrams

- UML notation to

  describe the **functionality** of a system as seen by the users in terms of

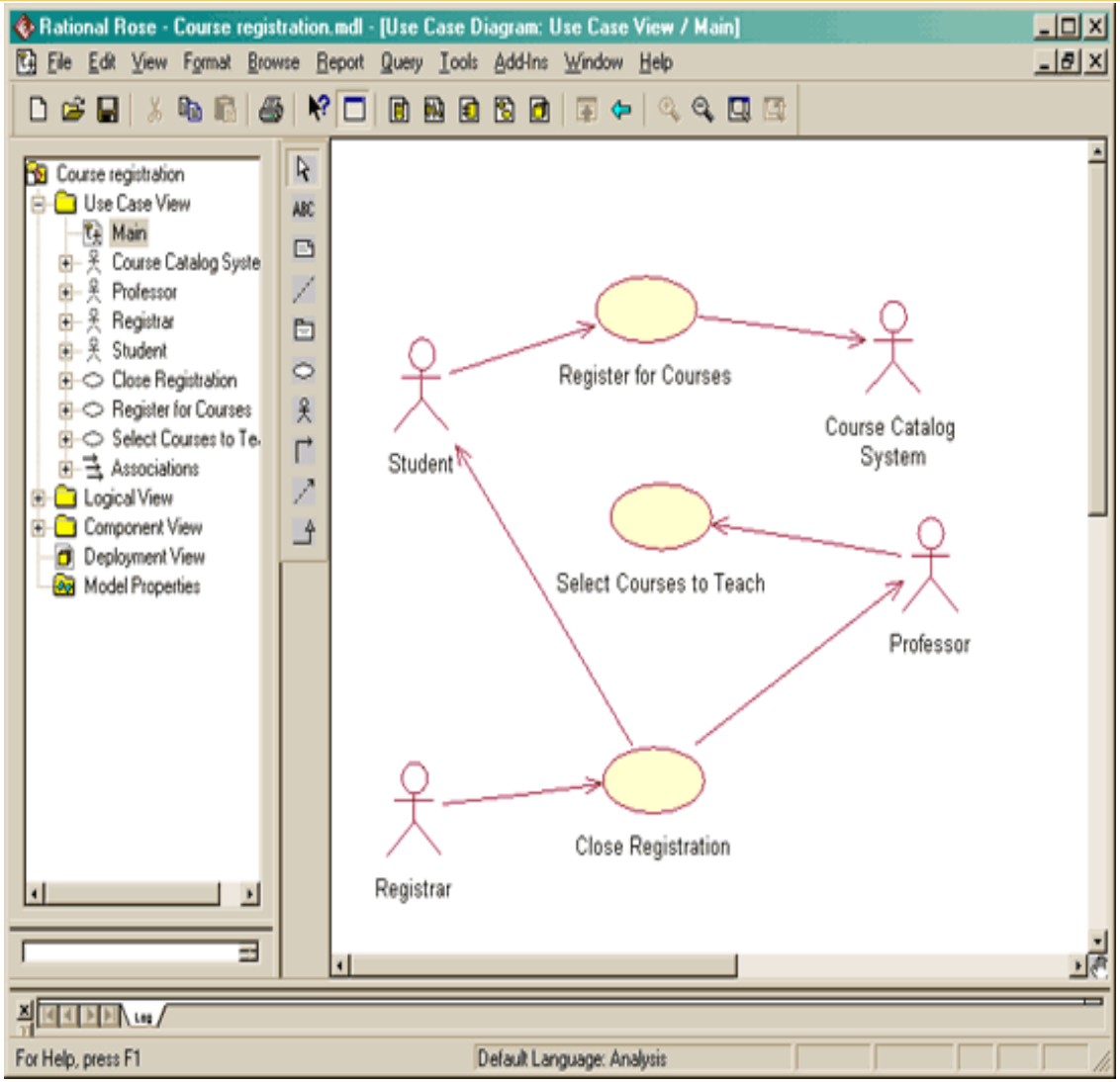
  - actors and their goals
  - the system boundary: what is in scope and out of the scope of the system?
  - the names of use cases for the system

FRIEND Use Case Diagram



FIRST: An accident management system and stands for
First **R**esponder **I**nteractive **E**mergency **N**avigational **D**atabase

Use Case Diagrams



The ovals represent use cases, and the stick figures represent actors, either humans or other systems

The lines represent communication between an actor and a use case.

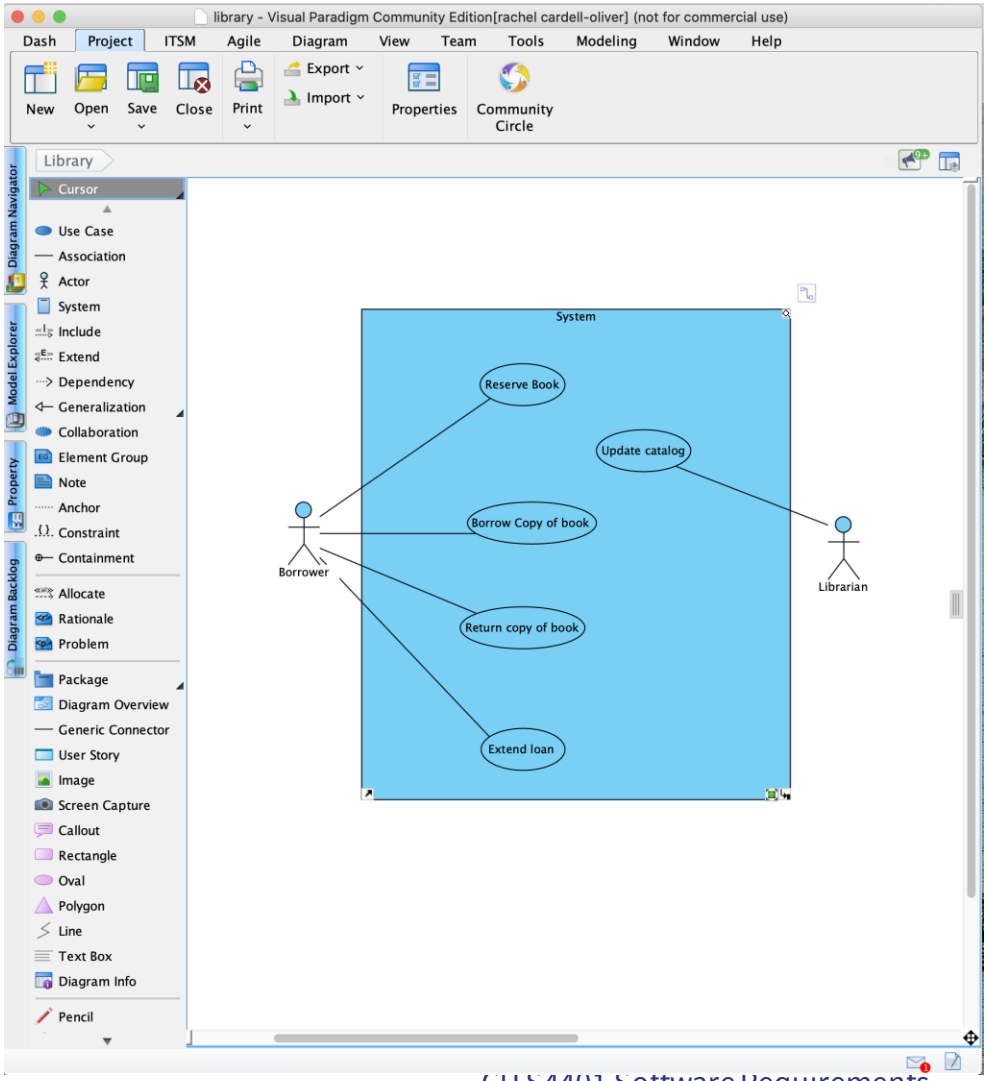
A use-case diagram provides the big picture:

Each use case represents a big chunk of functionality that will be implemented, and

each actor represents someone or something outside our system that interacts with it.

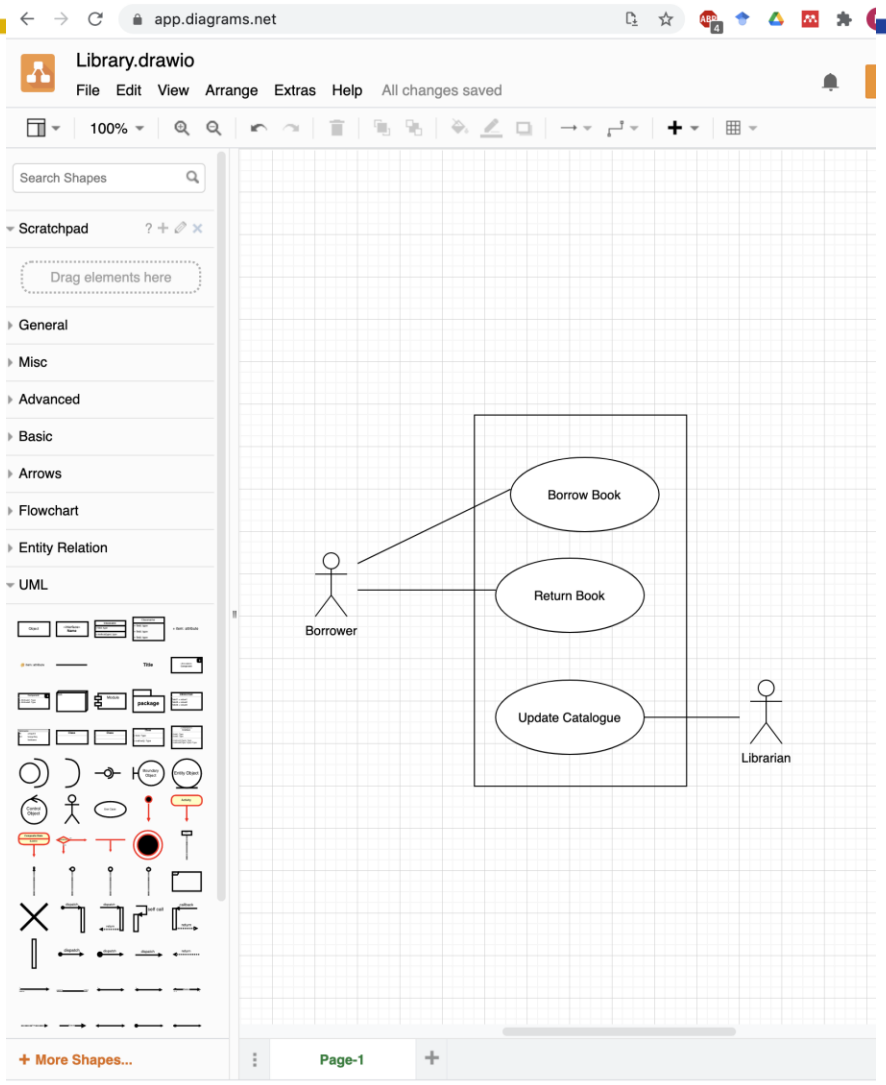
UML drawing tools

- https://www.visual-paradigm.com/download/community.jsp
- Make sure you download the FREE community edition (free for non-commercial use)



UML drawing tools (cont)

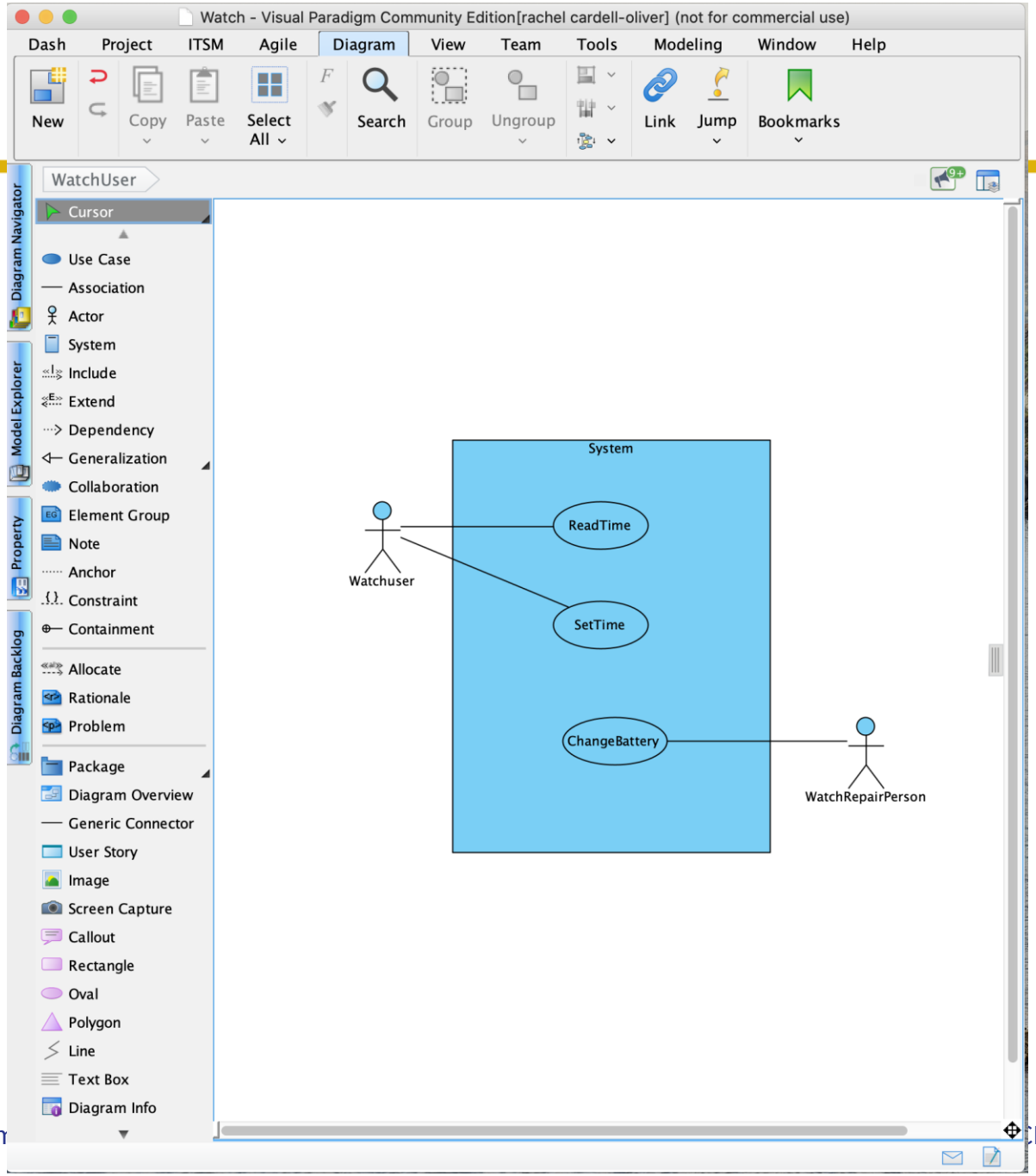
- https://app.diagrams.net/
- draw.io
- Online portal



Example

Draw a use case diagram for a simple watch based on the following problem statement:

The `WatchUser` actor may either consult the time on their watch (with the `ReadTime` use case) or set the time (with the `SetTime` use case). However, only the `WatchRepairPerson` actor can change the battery of the watch (with the `ChangeBattery` use case).


Watch - Visual Paradigm Community Edition[rachel cardell-oliver] (not for commercial use)
Dash
Project
ITSM
Agile
Diagram
View
Team
Tools
Modeling
Window
Help
New
Copy
Paste
Select All
Search
Group
Ungroup
Link
Jump
Bookmarks
WatchUser
Cursor
Use Case
Association
Actor
System
Include
Extend
Dependency
Generalization
Collaboration
Element Group
Note
Anchor
Constraint
Containment
Allocate
Rationale
Problem
Package
Diagram Overview
Generic Connector
User Story
Image
Screen Capture
Callout
Rectangle
Oval
Polygon
Line
Text Box
Diagram Info
Diagram Navigator
Model Explorer
Property
Diagram Backlog
System
ReadTime
SetTime
ChangeBattery
Watchuser
WatchRepairPerson

Terminology - don't get confused

A use case diagram is not the same as a UML use case

Use cases (planned requirements methodologies)

A use case is similar to a user story, because it also describes one specific interaction between the user and the software. Use cases are about the behaviour you'll build into the software to meet those needs. Use cases describe a complete interaction between the software and users (and possibly other systems).

A **use case diagram** is a high level summary of the actors and named used cases of a system

Sometimes the term **use case** is used to describe a specific situation in which a product or service could potentially be used.

Use cases – some history

- Projects have struggled for years to conceive the best approach to elicit, document, and trace functional requirements.
- Approaches range from mini-specifications -- a text narration of the requirements in paragraph form -- to diagrams that show each requirement's flow of control.
- *Ivar Jacobson* pioneered the notion of use cases while working on complex telecommunications projects at Ericsson

UML Use Cases

- A powerful technique for identifying requirements.
- One of the foundation techniques used in Object-oriented analysis methodologies such as UML (Unified Modeling language), Objectory and Booch OOA/OOD.
- A user scenario is a sequence of events that a user performs in order to perform some function within the system.
- A table and text-based template is used to describe each use case

Use Case example 1

Use case name: PlanTrip

Flow of events:

1. The Driver activates her computer and logs into the trip-planning Web service
2. The Driver enters constraints for a trip as a sequence of destinations
3. Based on a database of maps, the planning service computes the shortest way of visiting the destinations in the order specified. The result is a sequence of segments being a series of crossings and a list of directions.
4. The driver can revise the trip by adding or removing destinations
5. The Driver saves the planned trip by name in the planning service database for later retrieval

Use case example 2

Use case name: ExecuteTrip

Flow of events:

1. The driver starts her car and logs into the onboard route assistant
2. Upon successful login, the Driver specifies the planning service and the name of the trip to be executed
3. The onboard route assistant obtains the list of destinations, directions, segments and crossings from the planning service
4. Given the current position, the route assistant provides the driver with the next set of directions
5. The Driver arrives at the destination and shuts down the route assistant

Scenarios and Use Cases. Why?

- Comprehensible by all system stakeholders
  - Describe functional requirements from the users' point of view
- Great tools to manage a project. They can form basis for whole development process
  - User manual
  - System design and object design
  - Implementation
  - Test specification
  - Client acceptance test
- An excellent basis for incremental & iterative development

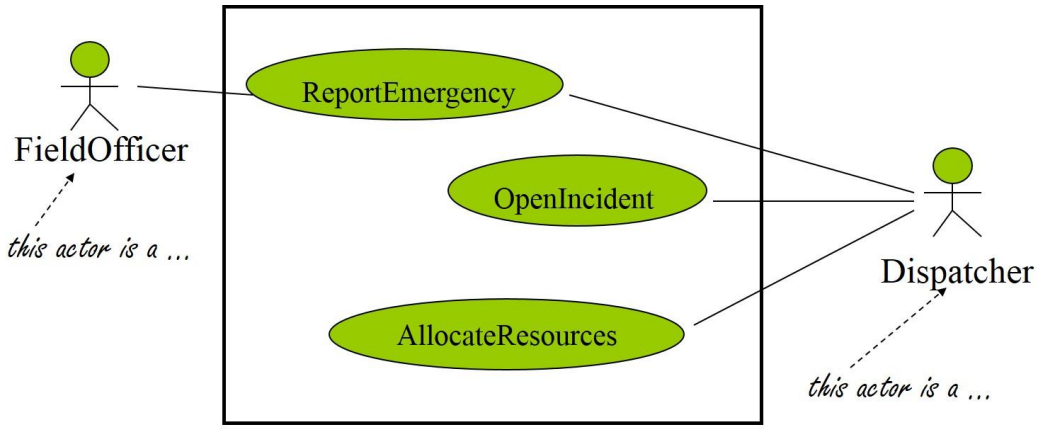
Scenarios vs. use cases

- Use case
  - a sequence of steps describing an interaction between a user and a system
  - used to gather stories and build requirements
  - focus is on understandability
- Scenario
  - an abstraction that describes all possible scenarios involving the described functionality
  - focus is on completeness

Scenario example (from FRIEND system)

- Scenario name: `warehouseOnFire`

- Participating actors:
  - Bob – `FieldOfficer`
  - James – `Dispatcher`

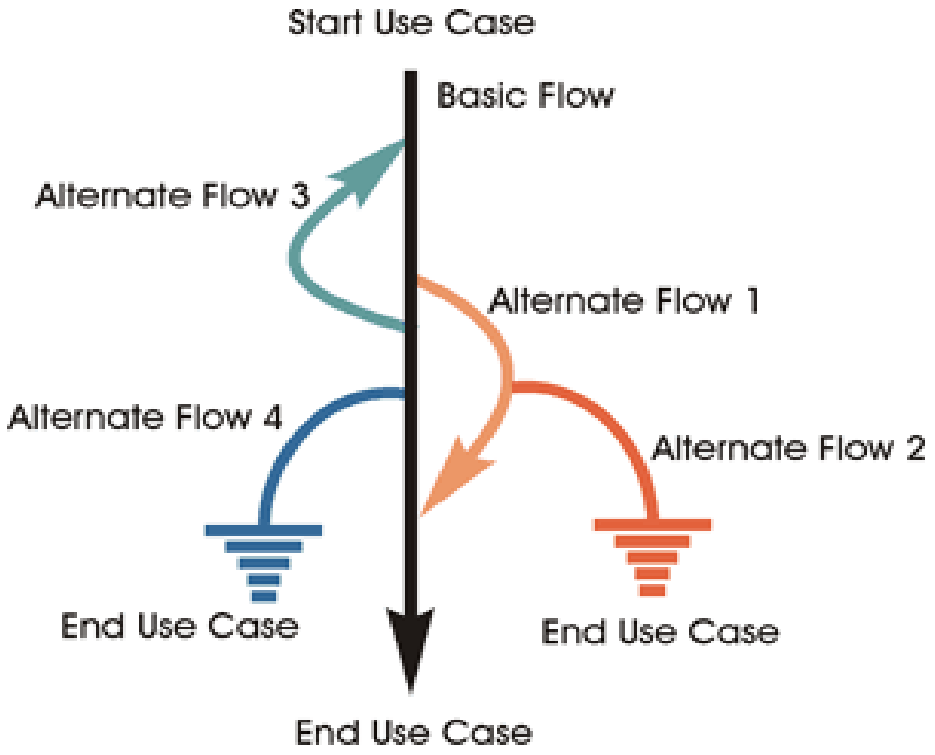


- Flow of events:
  - Bob is driving down main street in a patrol car and notices smoke coming out of a warehouse. He activates the `ReportEmergency` function of FRIEND
  - Bob enters the details into the system
  - James is alerted by a several beeps. He reviews the information and records the incident using the `OpenIncident` function
  - James allocates a fire unit and two paramedic units to the scene with the `AllocateResources` function

Use Case Text Description (detailed)

Use Case Section	Description
Name	An appropriate name for the use case – a short active verb phrase e.g. RegisterForCourses
Goal	A brief description of the use case's role and purpose, that is its goal
Flow of Events	A textual description (understandable to the customer) of what the system does with regard to the use case *(not* how specific problems are solved by the system).
Special Requirem-ents	Collects all requirements on the use case, e.g. non-functional reqs, that are not considered in the use-case model, but that need to be taken care of during design or implementation.
Pre-conditions	A textual description that defines any constraints on the system at the time the use case may start.
Post conditions	A textual description that defines any constraints on the system at the time the use case will terminate.

Basic and Alternate Flow of Events



Use cases describe possible flows of events.

basic flow of events describes what "normally" happens when the use case is performed.

alternate flows of events covers behavior of an optional or exceptional character relative to normal behavior, and also variations of the normal behavior.

Think of the alternate flows of events as "detours" from the basic flow of events.

RegisterForCourses Basic Flow of Events

1.Logon This use case starts when a Student accesses the University Web site. The system asks for, and the Student enters, the student ID and password.

2.Select 'Create a Schedule' The system displays the functions available to the student. The student selects "Create a Schedule."

3.Obtain Course Information The system retrieves a list of available course offerings from the Course Catalog System and displays the list to the Student.

4.Select Courses The Student selects four primary course offerings and two alternate course offerings from the list of available course offerings.

5.Submit Schedule The student indicates that the schedule is complete. For each selected course offering on the schedule, the system verifies that the Student has the necessary prerequisites.

6.Display Completed Schedule The system displays the schedule containing the selected course offerings for the Student and the confirmation number for the schedule.

Some Alternate Flows of Events (1)

THE UNIVERSITY OF WESTERN AUSTRALIA

1. Unidentified Student

In Step 1 of the Basic Flow, Logon, if the system determines that the student ID and/or password is not valid, an error message is displayed

2. Quit

The Course Registration System allows the student to quit at any time during the use case. The Student may choose to save a partial schedule before quitting. All courses that are not marked as "enrolled in" are marked as "selected" in the schedule. The schedule is saved in the system. The use case ends.

Alternate Flow of Events (2)

3. Unfulfilled Prerequisites, Course Full, or Schedule Conflicts

In Step 5 of the Basic Flow, Submit Schedule, if the system determines that prerequisites for a selected course are not satisfied, that the course is full, or that there are schedule conflicts, the system will not enroll the student in the course. A message is displayed that the student can select a different course. The use case continues at Step 4, Select Courses, in the basic flow

Alternate Flow of Events (3)

4. Course Catalog System Unavailable

In Step 3 of the Basic Flow, Obtain Course Information, if the system is down, a message is displayed and the use case ends

5. Course Registration Closed

If, when the use case starts, it is determined that registration has been closed, a message is displayed, and the use case ends.

Writing Use Cases - Summary

- Writing a use case involves a lot of work
- Ideally, the flows should be written as "dialogs" between the system and the actors
- Each step should explain what the *actor does* and what the *system does in response*
- It should also be **numbered** and have a **title**
- *Alternate flows* always specify where they start in the basic flow and where they go when they end

Recommended reading

- Martin Fowler, UML Distilled
  - Introduction
- Bruegge and Dutoit
  - Section 2.2.1, 2.3.5, 2.4.1, 4.4
- Pressman
  - Understanding Requirements > Developing Use Cases
  - Requirements Modelling > Scenario-Based Modelling
  - Appendix 1: An Introduction to UML