COMP354

Software Engineering

Lecture 3 and 4

Requirements and Analysis

Requirements and Analysis

Analysis is the process of understanding a problem and breaking it down into its constituent parts.

For software systems, the “parts” are the *requirements*, both *functional* requirements concerned with the behavior of the system in terms of responding to user inputs, and
all other constraints on the system, the so-called *non-functional* requirements.

Analysis concentrates on **what** is required

Communicating with customers and users is aided by use cases, usage scenarios, and diagrams

Main product is the *Software Requirements Document* (SRD)

but sometimes a *user manual* is used to capture requirements

and *test cases* and *prototypes* of user interface
are also produced during analysis

In OO methodologies the document produced may be an *Analysis Model*, or *Analysis Document* with several models.

Object-Oriented Analysis Document

This document describes the system from the users perspective

Records WHAT, not HOW.

Problem Statement summarises the purpose of the system

Context Diagram shows the external actors that interact with the system. Places the system in context.

Use Cases capture what services the system provides to users in a user-friendly text narrative, and organizes the use cases by identifying common services

Structural Models record the *entities* and their *relationships*. The entities are organised into *classes*.

Behavioral Models record when, and in what order, *events* occur as the user interacts with the system.

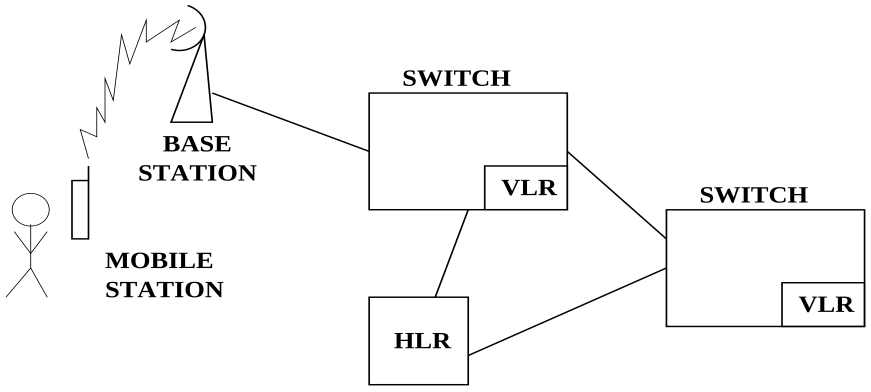
Global collaboration diagram shows overview of message-passing between structural elements.

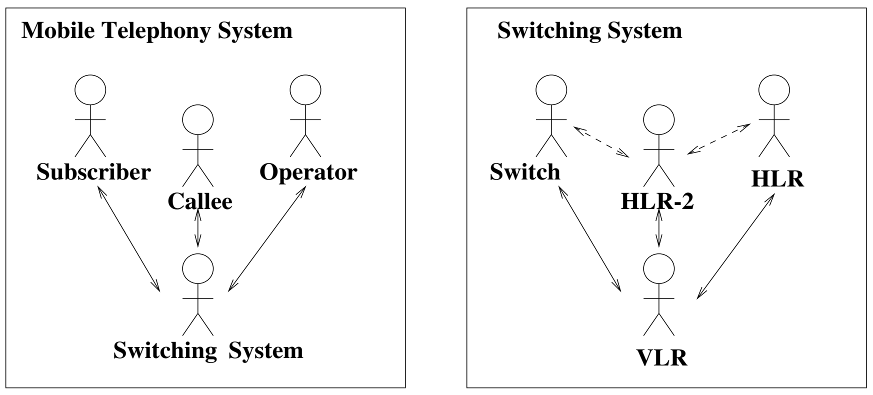
Data Dictionary records definitions of terms used in the above models

Context

The *context* captures where the software system is to fit in the broader scheme of things, such as the operations of the company.

Context diagram is often a use case diagram; often just free-form picture





Home Location Register(HLR) is a database of subscriber information

Visitor Location Register(VLR) is the local database of information on subscribers roaming in the service area of the Switch

Use Cases

Use cases capture the functionality of a system, called the *target system*, as it is meant to behave in a given environment called the *host system*.

A use case describes how a group of external entities, called *actors*, make use of the target system.

The use they make is modeled by the passing of signals or information between the actors and the system.

We will call these *messages*.

stimuli are messages from actors to the system

responses are messages from the system to actors

Several actors may participate in a use case.

the *primary actor* is the actor who initiates the use case

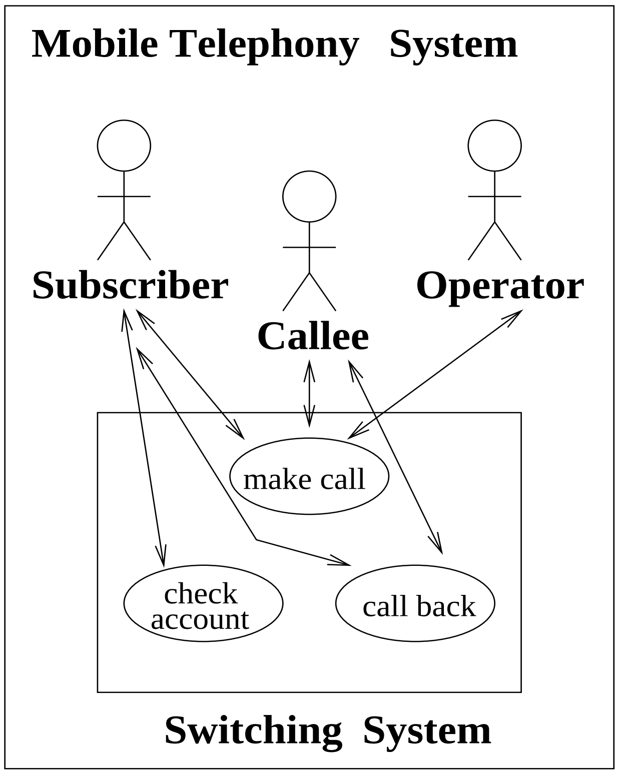
the other actors are called *secondary*

Use Cases

A *use case* is a description of a cohesive set of dialogues that the primary actor initiates with a system.

The dialogues are *cohesive* in the sense that they are related to the same task, or form part of the same transaction.

Cohesiveness is often determined by having a *goal* in common for the tasks, or by having a common *responsibility* that must be fulfilled.



Describing a Use Case

Textual Synopsis of a Use Case

Use Case # 1: Make a Call
Primary Actor: Subscriber
Secondary Actor: Callee, Operator
Goal: To connect to Callee and talk.

As a Collection of Scenarios

a *scenario* is

"a description of one or more end-to-end transactions involving the required system and its environment"

- main scenario
- variant scenario
- exceptional scenario
- recovery scenario
- failure scenario
- alternative scenario

Kinds of Scenarios

main scenario describes the usual way in which the task is successfully performed

Typically, in the main scenario, the simplest sequence of interactions is described, and it is assumed that all steps execute successfully.

One high-level way to describe a use case is to include the main scenario with the textual synopsis.

Use Case # 1: Make a Call
Primary Actor: Subscriber
Secondary Actor: Callee, Operator
Goal: To connect to Callee and talk.
Main Scenario: The Subscriber dials the Callee's number, is connected, and speaks with the Callee. The Subscriber hangs up.

variant scenario describes another way of using the system where it is assumed that all steps execute successfully

exceptional scenario describes a scenario where exceptional or error conditions may arise

recovery scenario is an exceptional scenario where it is possible to recov er and to complete the task

failure scenario is an exceptional scenario where it is not possible to re cover

alternative scenario is either variant or exceptional

Describing Scenarios

Narrative

The Subscriber dials the Callee's number, is connected, and speaks with the Callee. The Subscriber hangs up.

Example of Variant Scenario as Narrative

The Subscriber calls the Operator and requests a person-to-person call. The Operator connects the Subscriber with the Callee. They speak. The Subscriber hangs up.

Example of Recovery Scenario as Narrative

The Subscriber dials the Callee's number. It is an old number and the Operator tells the Subscriber the new number. The Subscriber dials the Callee's new number, is connected, and speaks with the Callee. The Subscriber hangs up.

Example of Failure Scenario as Narrative

The Subscriber calls the Operator and requests a person-to-person call. The Callee is not answering.

Describing Scenarios

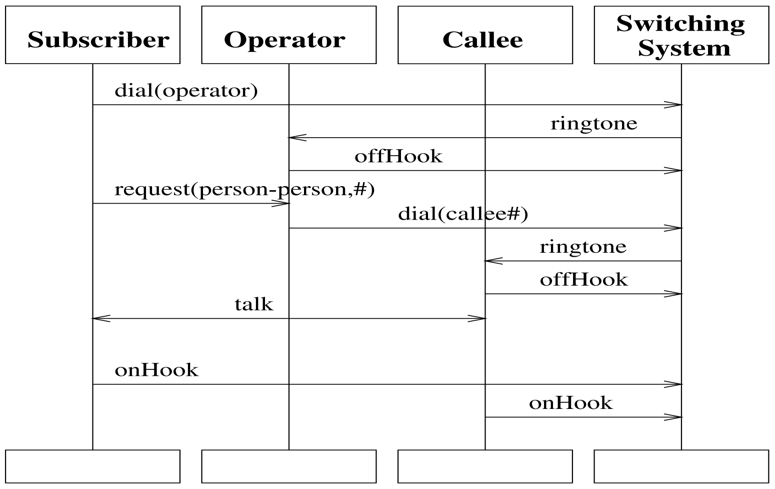
Script — Example of Variant Scenario

1. Subscriber dials Operator.
2. Subscriber requests person-to-person.
3. Operator connects Subscriber to Callee.
4. Subscriber talks to Callee.
5. Subscriber disconnects.

Agent-Action Table — Example of Recovery Scenario

No.	Agent	Action
1	Subscriber	Dials Callee (old number)
2	Operator	Informs Subscriber of new number
3	Subscriber	Dials Callee (new number)
4	System	Connects Subscriber and Callee
5	Subscriber	Talks to Callee
6	Subscriber	Disconnects

Interaction Diagram — Example of Variant Scenario



Episodes

Scenarios represent end-to-end executions of the system (the dialogues) grouped together according to some criteria. This is a horizontal gathering of dialogues.

It is also possible to divide a scenario vertically into a sequence of *episodes*.

Each episode represents a subtask, or the parts of the dialogues in the scenario that perform the subtask.

Episode 1: Failure to Connect

1.1 Subscriber dials Callee (old number)

Episode 2: Operator Intervention

2.1 Operator informs Subscriber of new number

Episode 3: Successful Call

3.1 Subscriber dials Callee (new number)
3.2 System connects Subscriber and Callee
3.3 Subscriber talks to Callee
3.4 Subscriber disconnects

The Analysis Process and Use Cases

A *requirements use case* is a use case that describes a complete user task or activity.

For a requirements use case, the system performs a meaningful unit of work of value to the primary actor.

An *analysis use case* describes a service, feature, or function that is offered by the system and is initiated by an actor.

An actor may perform a requirements use case by initiating a sequence of analysis use cases.

A requirements use case is a special kind of analysis use case where the service being described is a complete user task rather than simply a subtask.

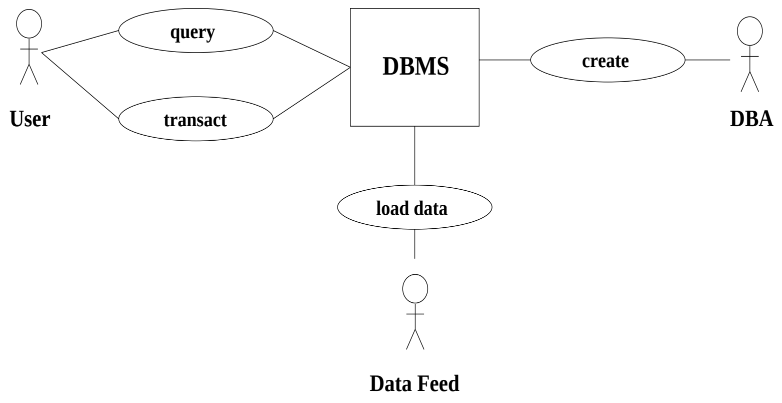
An episode can give rise to an analysis use case.

These can be further refined until one is at the level of *atomic use cases*, which represents a primitive unit of interaction with the system.

i.e. an operation of the implemented system

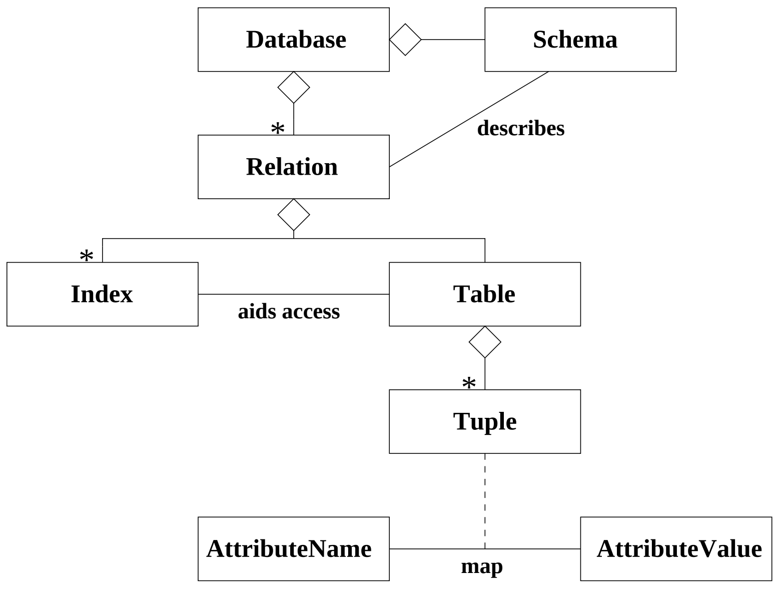
Use Case Diagram

Summarizes all the use cases and their relationships.



Class Diagram

Captures the entities and their relationships.



Usually very little information on attributes or operations.

Behaviour

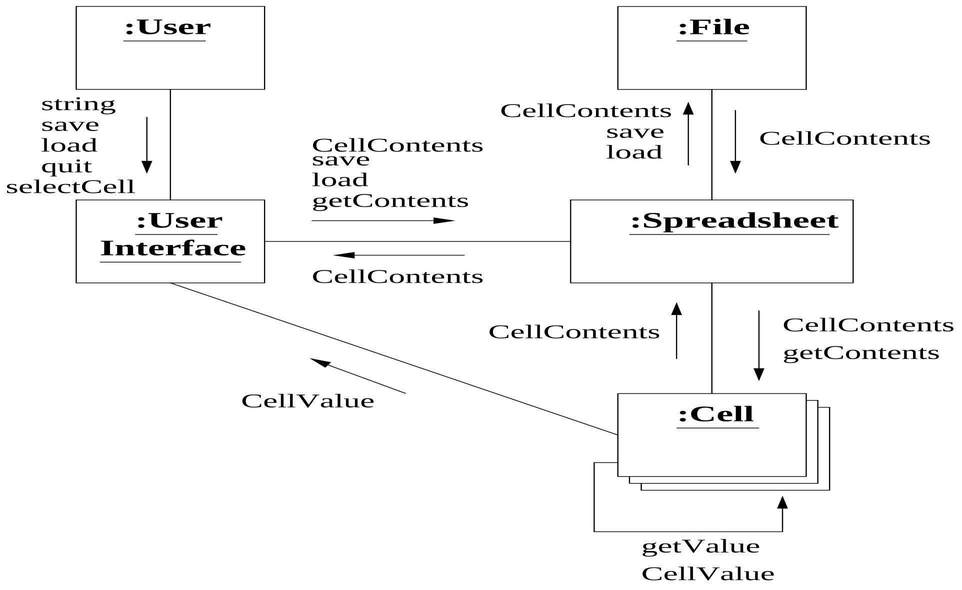
Behavioral Models record when, and in what order, *events* occur

- interaction diagram for each scenario
- statechart diagram for each class (with significant states)
- global collaboration diagram

Global Collaboration Diagram

The diagram summarises all the *events* that can flow between all *objects*.

An overlay of a collaboration diagram for each scenario but omitting sequence numbers, arguments, etc



NB events are both "data" and "control"

Software Requirements Document (SRD)

What does it contain?

a complete description of WHAT the software product will do (**without** describing HOW it will do it)

What is the purpose of the SRD?

agreement between the customer and the developer

clearly identifies what is needed and
what is to be developed

guidance for system testing and maintenance

Who are the users or beneficiaries of SRD?

- customers/clients/users/end-users
- supplier/developer/analyst
- person who verifies the software
- person who maintains the software

Why requirements should be precisely stated beforehand?

The later in the development lifecycle that an error is detected the more expensive it is to correct

Many errors remain concealed until well after the stage in which they are made.

(This is experienced in real life)

The requirements analysis can identify errors such as

- incorrectness
- inconsistency
- ambiguities

which if allowed would cause drastic effects later.

Industry Experience

After delivery, it is

100 times more expensive

to fix a

problem with the *requirements*

than to fix a

problem with the *implementation*

SRD production process

analysis to understand, organize, and document the problem

- learning the jargon of the problem domain
- learning about the problem to be solved
- identifying the real users of the product
- understanding the needs of the real users
- understanding the constraints on the solution, such as existing resources and skill levels
- paraphrase requirements from domain/user jargon to a mutually understandable notation

validate your analysis with the customer/user

write the SRD: a complete description of the external behaviour of the product

- functional requirements : a description of how various inputs map into outputs
  (also called behavioral or operational requirements)
- non-functional constraints: attributes of the system such as
  - mandated hardware and software
  - performance requirements
  - reliability, portability, security, etc
- priorities
- anticipated or likely changes

also review your writing and its consistency with the analysis

Properties of a good SRD

correct: every requirement stated in the SRD represents something required of the system

complete: everything that the software is supposed to do is included in the SRD
no sections of the SRD contain the phrase
"To be determined" (TBD)

consistent: no conflicts between requirements

precise, unambiguous: every requirement stated in the SRD has only one interpretation

verifiable: every requirement is testable by means of some finite process which is cost effective

understandable: every requirement is easy to understand by all who use the SRD

modifiable: it is easy to make changes to the requirements and to check that such requirements do not lead to conflicts

annotated: annotations state the relative importance or priority of features, especially the necessity or optionality of functional features and the priority of nonfunctional requirements

General Outline of an SRD

Introduction Purpose and context of system

The System Model Overview of the system components and their relationship
(*declarative* point of view, from users perspective)

Functional Requirements Description of each functionality in terms of required and optional inputs, conditions on inputs, and effects on changing the state of the system and producing outputs

Non-functional Requirements Description of constraints on the system that are not functional, eg
hardware and software platform: SUN, Unix, X
compatibility with external standards: GIF, ODE
performance constraints: speed, memory and disk usage, availability/reliability

Fundamental Assumptions Aspects of the context that are deemed unlikely to change

Anticipated Changes Aspects of the context that are likely to change, and in what respect they will change

Glossary Definition of terminology and acronyms
(for nontechnical users)

Bibliographic References to standards documents, company policy documents, user/customer reference documents

Index Guide to location of definition and uses of terms

Information must be precise and easily found

- unique label/number per requirement
- unique name for each concept and entity
- table of contents
- one or more indexes
- glossary of terms
- cross-references between requirements