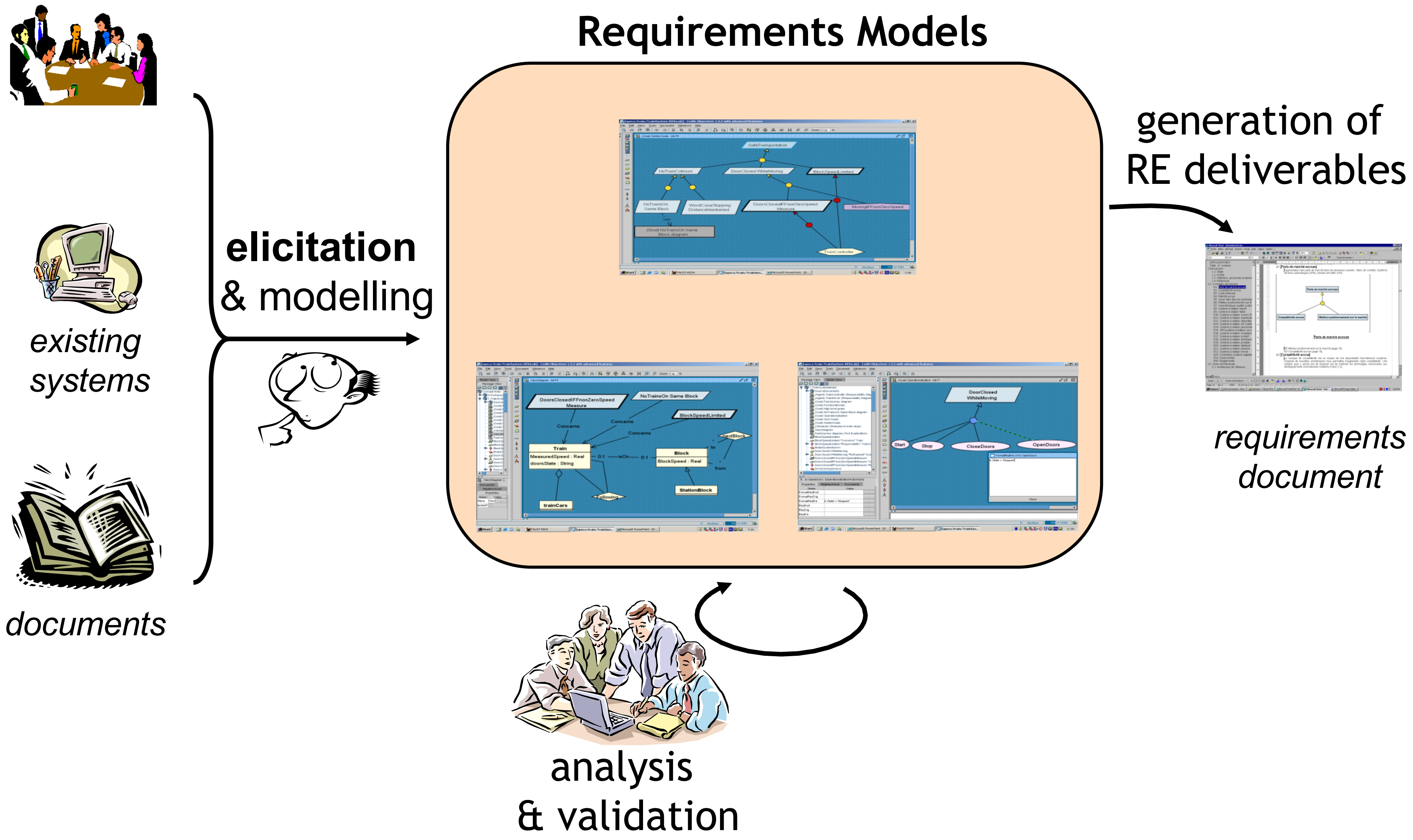
Politecnico
di Milano

Elicitation of requirements

Role of requirements models



Complexity example: rules and procedures for loan approval

- Implicit knowledge:
  - *There is no document in which the rules for approving loans are written down or the document are ambiguous, incomplete, outdated*
- Conflicting information:
  - *Different members of the department have different ideas about what the rules are*
- Bias:
  - *The loan approval officers fear that your job is to computerize their jobs out of existence, so they are deliberately emphasizing the need for case-by-case discretion (to convince you it has to be done by a human!)*
- Tacit knowledge:
  - *The loan approval process described to you by the loan approval officers is quite different from your observations of what they actually do*
- Probe effect:
  - *The loan approval process used by the officers while you are observing is different from the one they normally use*

Scenarios

- “A narrative description of what people do and experience as they try to make use of computer systems and applications” [M. Carrol, Scenario-based Design, Wiley, 1995]

- A concrete, focused, informal description of a single feature of the system

Scenario example: warehouse on fire

- *Bob, driving down main street in his patrol car notices smoke coming out of a warehouse. His partner, Alice, reports the emergency from her car.*

- *Alice enters the address of the building, a brief description of its location (i.e., north west corner), and an emergency level. In addition to a fire unit, she requests several paramedic units on the scene given that area appears to be relatively busy. She confirms her input and waits for an acknowledgment.*

- *John, the Dispatcher, is alerted to the emergency by a beep of his workstation. He reviews the information submitted by Alice and acknowledges the report. He allocates a fire unit and two paramedic units to the Incident site and sends their estimated arrival time (ETA) to Alice.*

- *Alice received the acknowledgment and the ETA.*

Observations

- Concrete scenario
  - Describes a single instance of reporting a fire incident
  - Does not describe all possible situations in which a fire can be reported

- Participating actors
  - Bob, Alice, and John

Questions for identifying actors

- Which user groups are supported by the system to perform their work?
- Which user groups execute the system’s main functions?
- Which user groups perform secondary functions such as maintenance and administration?
- With what external hardware or software system will the system-to-be interact?

Heuristics for finding scenarios

- Ask yourself or the client the following questions:
  - What are the primary tasks that the system needs to perform?
  - What data will the actor create, store, change, remove or add in the system?
  - What external changes does the system need to know about?
  - What changes or events will the actor of the system need to be informed about?
- However, don’t rely on questionnaires alone
- Insist on **task observation** if the system already exists (interface engineering or reengineering)
  - Ask to speak to the end user, not just to the software contractor
  - Expect resistance and try to overcome it

So far…

- Scenarios provide a nice summary of what the requirement analysis team can derive from
  - Observation
  - Interviews
  - Analysis of documentation
- … they can be very specific
- How to abstract from details and specificities?
  - … Use cases!

After the scenarios are formulated…

- Generalize scenarios
  - Example: from “Warehouse on fire” we can generalize a “Report Emergency” use case
- Describe each of these use cases in more detail
  - Participating actors
  - Describe the Entry Condition
  - Describe the Flow of Events
  - Describe the Exit Condition
  - Describe Exceptions
  - Describe Special Requirements (Constraints, Nonfunctional Requirements)

Use cases

- A use case is a flow of events in the system, including interaction with actors
- It is initiated by an actor
- Each use case has a name
- Each use case has a termination condition

Use Case Model: The set of all use cases specifying the complete functionality of the system

Use case example: ReportEmergency

- Use case name: ReportEmergency
- Participating actors:
  - Field Officer (Bob and Alice in the Scenario)
  - Dispatcher (John in the Scenario)
- Exceptions:
  - The FieldOfficer is notified immediately if the connection between her terminal and the central is lost
  - The Dispatcher is notified immediately if the connection between any logged in FieldOfficer and the central is lost
- Flow of Events: on next slide…
- Special Requirements:
  - The FieldOfficer’s report is acknowledged within 30 seconds; the selected response arrives no later than 30 seconds after it is sent by the Dispatcher

Use case example: flow of events

- The **FieldOfficer** activates the “Report Emergency” function of her terminal. **FRIEND** (the system to be developed) responds by presenting a form to the officer.
- The FieldOfficer fills the form, by selecting the emergency level, type, location, and brief description of the situation. The FieldOfficer also describes possible responses to the emergency situation. Once the form is completed, the FieldOfficer submits the form, at which point, the **Dispatcher** is notified.
- The Dispatcher reviews the submitted information and creates an Incident in the database by invoking the OpenIncident use case. The Dispatcher selects a response and acknowledges the emergency report.
- **Termination condition:** The FieldOfficer receives the acknowledgment and the selected response.

Example of a poor use case

- Use case name: Accident
- Participating Actors:
  - Field Officer
- Flow of Events:
  - 1. The field officer report the accident
  - 2. An ambulance is dispatched
  - 3. The Dispatcher is notified when the ambulance arrives on site

(not an action, i.e., a verb)

(Dispatcher actor is missing here but mentioned in the next section)

(by who?)

(what does the field officer do after dispatching?)

Tips to define proper use cases

- Use cases named with verbs that indicate what the user is trying to accomplish
- Actors named with nouns
- The boundary of the system should be clear, the steps accomplished by actors and those accomplished by the system should be clearly distinguished
- Use cases steps in active voice
- The causal relationship between steps should be clear
- A use case per user transaction
- Separate description of exceptions
- Keep use cases small (no more than two/three pages)

Use case example: allocate a resource

- Actors:
  - ***Resource Allocator:*** The Resource Allocator is responsible for allocating resources in case they are scarce
  - ***Dispatcher:*** A Dispatcher updates, and removes Emergency Incidents, Actions, and Requests in the system. The Dispatcher also allocates a resource to an Emergency if the resource is available
  - ***Resources:*** The Resources that are allocated to the Emergency

Allocate a resource (1)

- Use case name: AllocateResources
- Participating Actors:
  - Dispatcher (John in the Scenario)
  - Resource Allocator
  - Resources
- Entry Condition
  - An Emergency Incident has been opened
- Flow of Events
  - The Dispatcher selects the types and number of Resources that are needed for the incident
  - FRIEND replies with a list of Resources that fulfill the Dipatcher’s request
  - The Dispatcher selects the Resources from the list and allocates them for the incident
  - FRIEND automatically notifies the Resources
  - The Resources send a confirmation

Allocate a resource (2)

- Exit Condition
  - The use case terminates when the resource is committed.
  - The selected Resource is now unavailable to any other Emergency Incidents or Resource Requests
- Exceptions
  - If the list of Resources provided by FRIEND is insufficient to fulfull the needs of the emergency, the Dispatcher informs the Resource Allocator
  - The Resource Allocator analyzes the situation and selects new Resources by decommitting them from their previous work
  - The Resources confirm the new allocation

Steps when formulating use cases

- First step: name the use case
  - Use case name: ReportEmergency
- Second step: Find the actors
  - Generalize the concrete names (“Bob”) to participating actors (“Field officer”)
  - Participating Actors:
    - Field Officer (Bob and Alice in the Scenario)
    - Dispatcher (John in the Scenario)
- Third step: Then concentrate on the flow of events
  - Use informal natural language

How to specify a use case (summary)

- Name of Use Case
- Actors
  - Description of Actors involved in use case)
- Entry condition
  - “This use case starts when…”
- Flow of Events
  - Free form, informal natural language
- Exit condition
  - “This use cases terminates when…”
- Exceptions
  - Describe what happens if things go wrong
- Special Requirements
  - Nonfunctional Requirements, Constraints)

From use cases to functional requirements

- Each use case may lead to a requirement
- Examples
  - FRIEND shall support Field Officers in reporting an emergence
  - FRIEND shall support Dispatchers in allocating the resources to the incident

- The corresponding use cases will then describe in detail how the requirements are fulfilled