Introduction to Software Engineering Planning – Project and Product

CMPS115 – Spring 2016
Richard Jullig

Literature

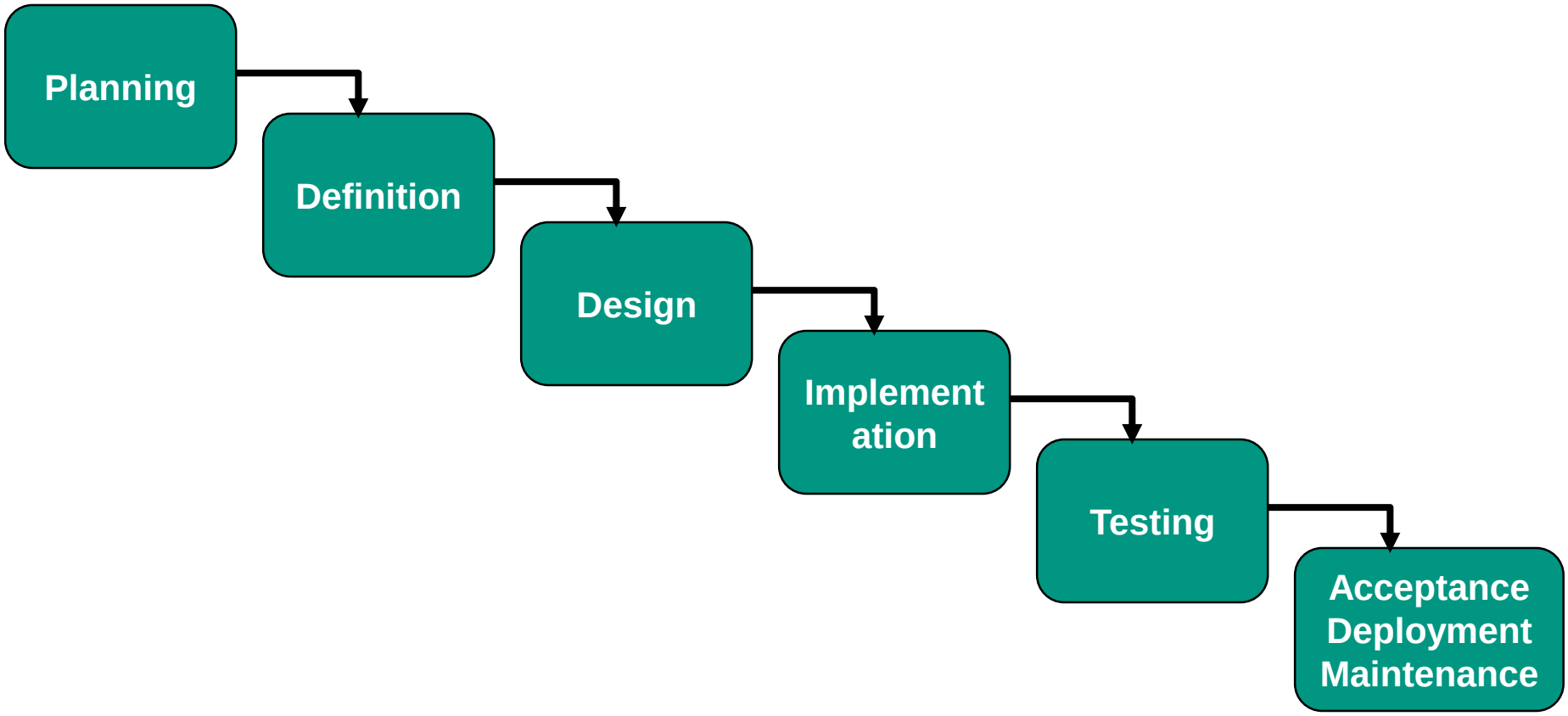
- Read
  - Section 2.4.1 Use Case diagrams
  - Chapter 4 Requirements Elicitation

in

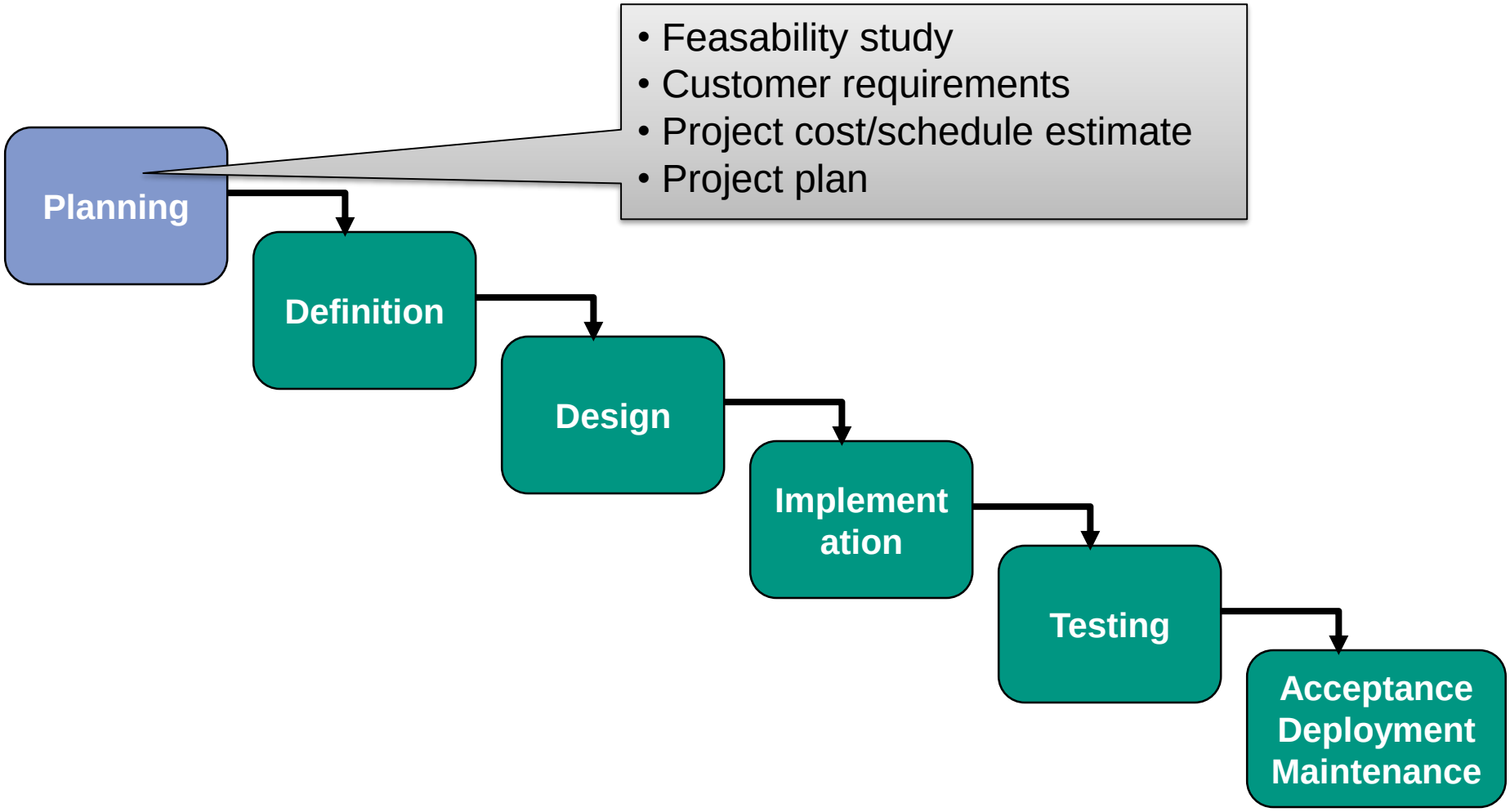
- B. Bruegge, A.H. Dutoit, **Object-Oriented Software Engineering: Using UML, Patterns and Java**, Pearson Prentice Hall.

- Available on Piazza > Resources > Reading

Waterfall of Activities



Waterfall of Activities - Overview



Planning phase

- Goal of planning phase
  - Describe target system in customer terms (language)
  - Check feasibility of project
    - Technical, organizational, schedule, cost
- Important part of planning phase
  - Customer/user requirements elicitation
- “Phase”
  - May be not be contiguous
  - May happen in several stages
  - Incremental requirements elicitation

What is a Requirement?

- IEEE Standard 610.12-1990

  Requirement:
  "A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document.
  The set of all requirements forms the basis for subsequent development of the system or system component".

Planning Phase – First Steps

- Selection of product/system to be developed based on
  - Customer order
  - Research results
  - Prior (earlier) developments
  - Trend studies
  - Market analyses

Product Requirements

- After product selection, determine key requirements
- Techniques for requirement elicitation
  - Questionnaires
  - Interviews
  - Task analysis
    - (what do people do now?)
  - Scenarios
    - Specific but typical situations/processes/events
  - Use cases

Note:

Use case is an alternative to User Stories for requirements capture.

Further discussion later

Scenario (1)

- Scenario
  - Description of event or sequence of actions and events
  - Description of target system use from user perspective
  - May contain
    - text, pictures, videos, diagrams
    - Details about workplace, social setting
    - Resource restrictions

Example scenario:

- While Bob is driving down Main street with his police car he notices smoke coming from a warehouse. His partner, Alice, reports the emergency from the police vehicle to the dispatcher.
- Alice enters the warehouse address in her mobile computer together with a short description of the location (north-west corner of intersection) and a priority.
- She confirms her input and awaits confirmation.
- John, the dispatcher in the central office, is alerted by a beep from his computer. He analyses the information from Alice and confirms her message. He alerts the firefighters and communicates the ETA (estimated time of arrival) to Alice.
- Alice receives the confirmation and the ETA.

Remarks to Example Scenario

- The scenario is specific
  - It describes a single instance of reporting a fire.
  - It does not describe all possible ways or situations of reporting a fire.
- Participating actors:
  - Police, dispatcher

Scenarios (2)

- Scenarios also used in
  - (Acceptance) Testing and deployment
  - Design
    - Scenario-based design

Scenarios (3)

- Scenario-based requirements elicitation is iterative
  - Each scenario a unit of work (“work package”)
- Each work package is (iteratively) extended or revised when requirements change
- Scenario-based requirements elicitation is based on concrete descriptions, not abstract ideas
- Scenario-based requirements elicitation is informal without defined stopping point.

How to find scenarios?

- Don’t expect specific instructions from your customer/user as long as the system doesn’t exist.
  - Customer understands problem domain, not the solution domain.
- Communicate with the customer
  - Help customer formulate requirements
  - Have customer help you understand requirements
  - Requirements will change as the scenario develops.

Tips for finding scenarios

- Ask the customer the following questions:
  - What are the main purposes/functions of the system?
  - What data will the users generate, store, modify, delete, or enter into the system?
  - What changes outside the system needs the system to know about?
  - What changes or events needs the system user to know about?
- But: Never rely solely (exclusively) on questions.
- Use observations if a system already exists

UML Use Case diagrams

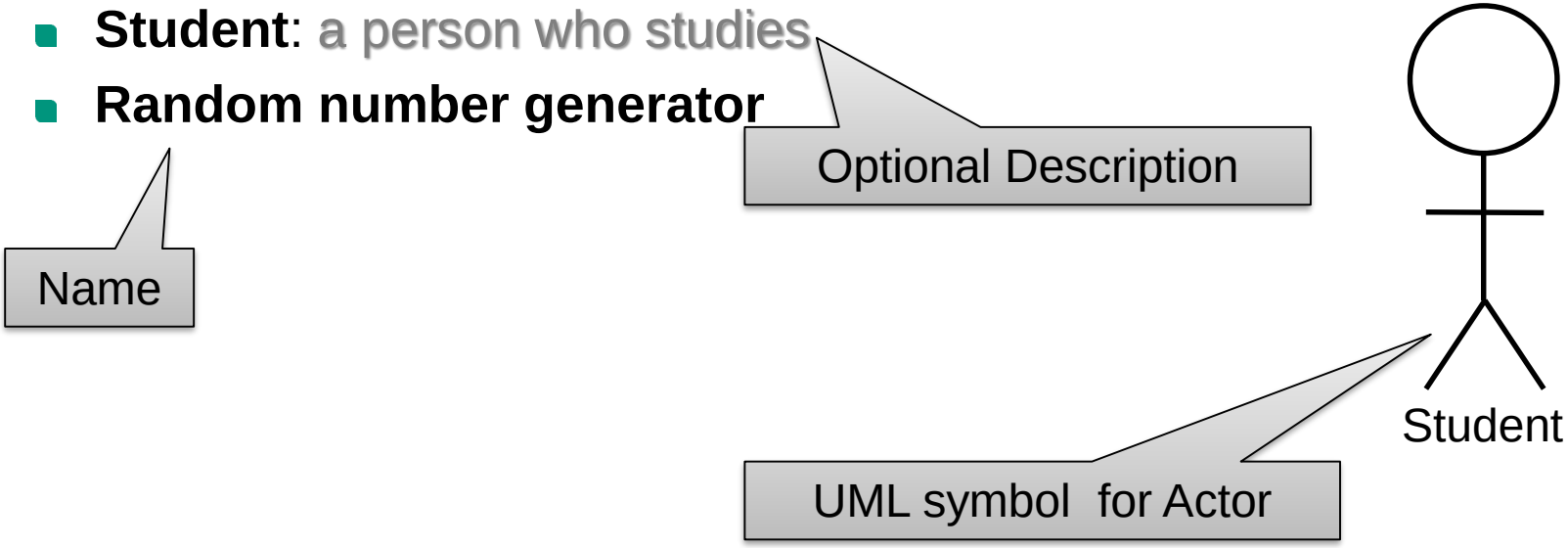
- Next time
- See literature on Piazza:
  - Ambler: Elements of UML 2.0 style
  - Bruegge 2.4.1 and chapter 4

First excursion into UML: Use case diagrams

- Use case diagrams
  - used during planning phase
  - capture requirements elicited from users
  - describe external system behavior during interactions with actors
- Actor
  - role filled by human user or other software system
- Use case
  - describes a class of functions/services offered by the system
  - that are meaningful to the actors involved
- Use case diagram (Use case model)
  - a collection of use cases
    - describing all or part of system functionality

Use cases

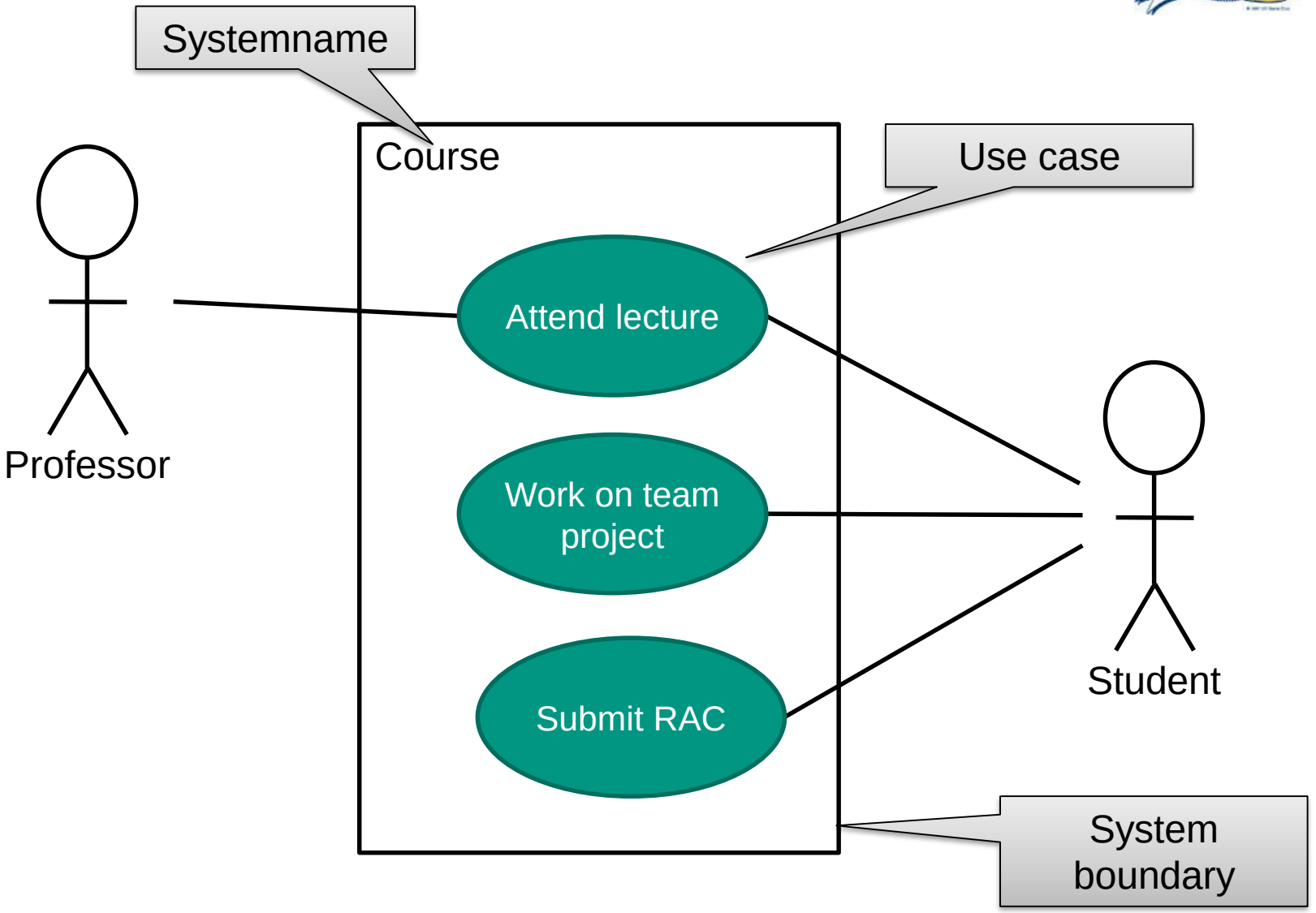
- Actor
  - models a unit (person, machine) that interacts with system
    - administrators, end users, environment, external systems
  - identified by unique name
  - optional description
- Example:
  - **Student**: a person who studies
  - **Random number generator**



UML Use case

- Use cases
  - focus on Actor-System interaction
  - described in natural language (English, Korean, German,…)
- UML specifies six parts for Use Case description
  - Unique name
    - recommended: verb-noun phrase
    - e.g. withdraw money (from ATM)
  - Participating Actors
  - Flow of events/interactions
  - Entry conditions
    - conditions that must hold when use case starts
  - Exit conditions
    - conditions that must hold when use ends
  - Quality requirements

UML Use Case Diagram

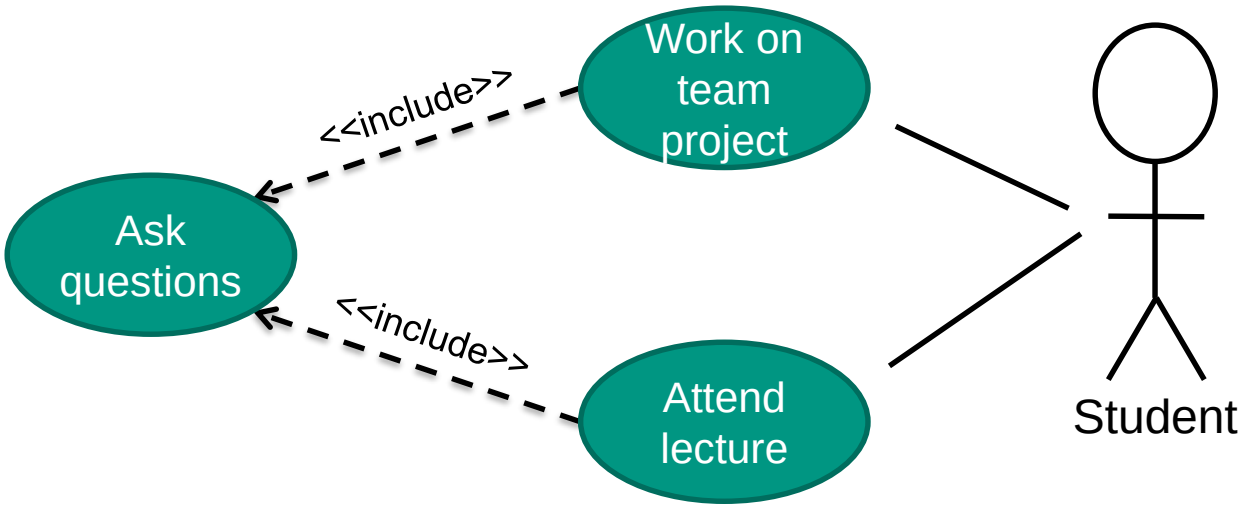


Relationships between Use Cases

- Use cases can be related to each other
- include relationship
  - used to factor out common functionality
    - include roughly like subroutine call
- extend relationship
  - used to enhance a common use case with additional functionality
  - extend roughly like inheritance/specialization

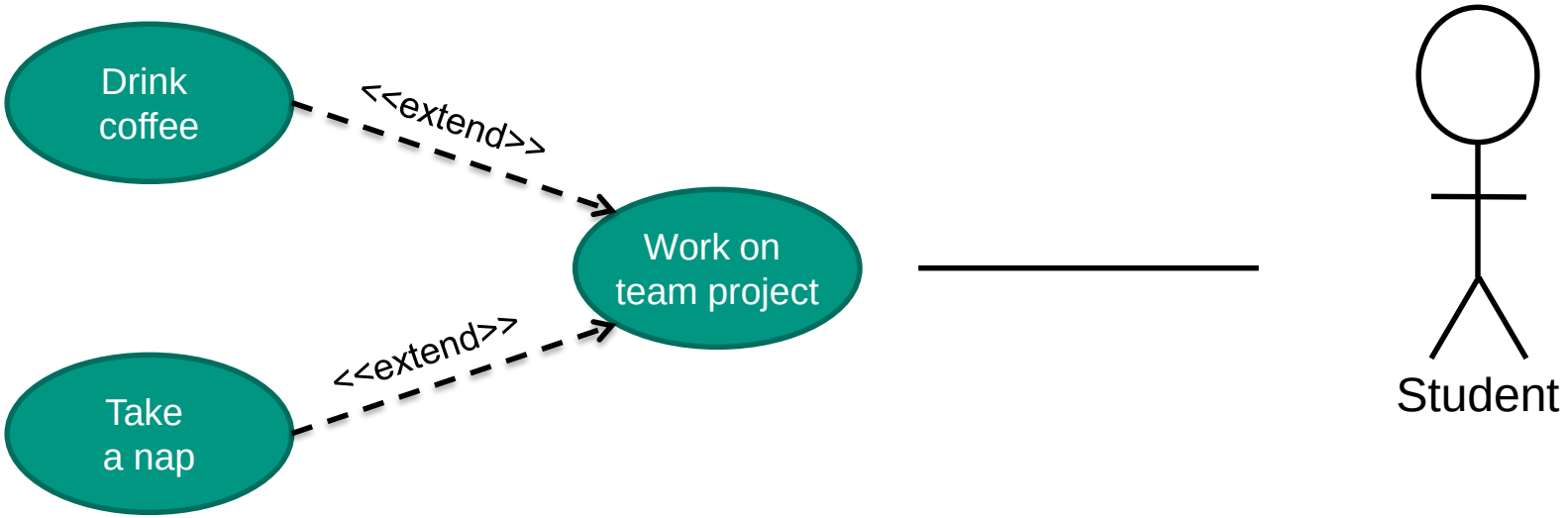
<<include>> Relationship

- <<include>>
  - allows use cases to refer to shared subfunctionality
  - allows factoring of shared functionality
  - <<include>> arrow goes from “using” use case to shared used case



<<extend>> relationship

- Exceptional or special functionality that needs to be added to common (“normal”) use cases
- <<extend>> arrow goes from extending use case to extended use case
- Exceptional/special use cases can extend more than one “normal” use case



How to find use cases?

- Study scenarios
- Once scenario may contain multiple instances of use cases
- Several scenarios may be instances of the same use case
- Example:
  Use case: Report emergency
  - Scenarios:
    - Warehouse on fire
    - Car accident
    - Tree fallen on powerline

Formulate use cases from scenarios

- Abstract
  - scenario name to use case name
    - Report warehouse fire → Report emergency
  - individual actors to roles
    - Bob, Alice → police officer
    - John → dispatcher
  - specific interactions between individuals to interactions between roles
    - describe interactions in natural language

Problems during Requirements Elicitation

- Shallow domain knowledge
  - distributed over many sources
    - rarely explicit
  - different sources may contradict each other
    - stakeholders have different goals/interests
    - stakeholders have different perspectives
- Tacit knowledge
  - difficult to express what is obvious/self-understood
- Limited observational awareness
  - observer may change behavior of observed
- Bias/distortion
  - stakeholders may not dare to say what is needed
    - politics, power relationships
  - stakeholders don't want to say what is needed
    - concern about own job
    - hidden agenda

Problems

- Underspecified requirement specification
  - Laser beam supposed to hit reflector on 10,023 high mountain
    - feet, meters, miles?
    - feet were intended, miles were computed

Problems

- Unintended “features”
  - London subway train leaves without conductor
- What happened?
  - conductor steps out to check on passenger door that didn’t close, leaving driver door open
  - when passenger closed, train started moving
  - condition “train should not move with door open” did not include driver door

Types of Requirements

- Functional requirements
  - describes interactions between actors and system
  - independent of implementation
  - “A police officer must be able to report an emergency.”
- Non-functional requirements
  - Quality aspects
  - “The response time to an emergency report must be less than 3 seconds.”
- Constraints
  - can concern any aspect of the system
  - “The implementation language shall be Java.”

Non-functional Requirements – Software Qualities

Qualities

- Usability
- Reliability
  - robustness
  - security/safety
- Performance
  - response time
  - scalability
  - throughput
  - availability
- Maintainability
  - adaptability
  - extensibility

Constraints on

- implementation
- interfaces
- operating environment
- deployment
- Legal
  - licenses
  - certificates
  - data privacy
  - …
- …