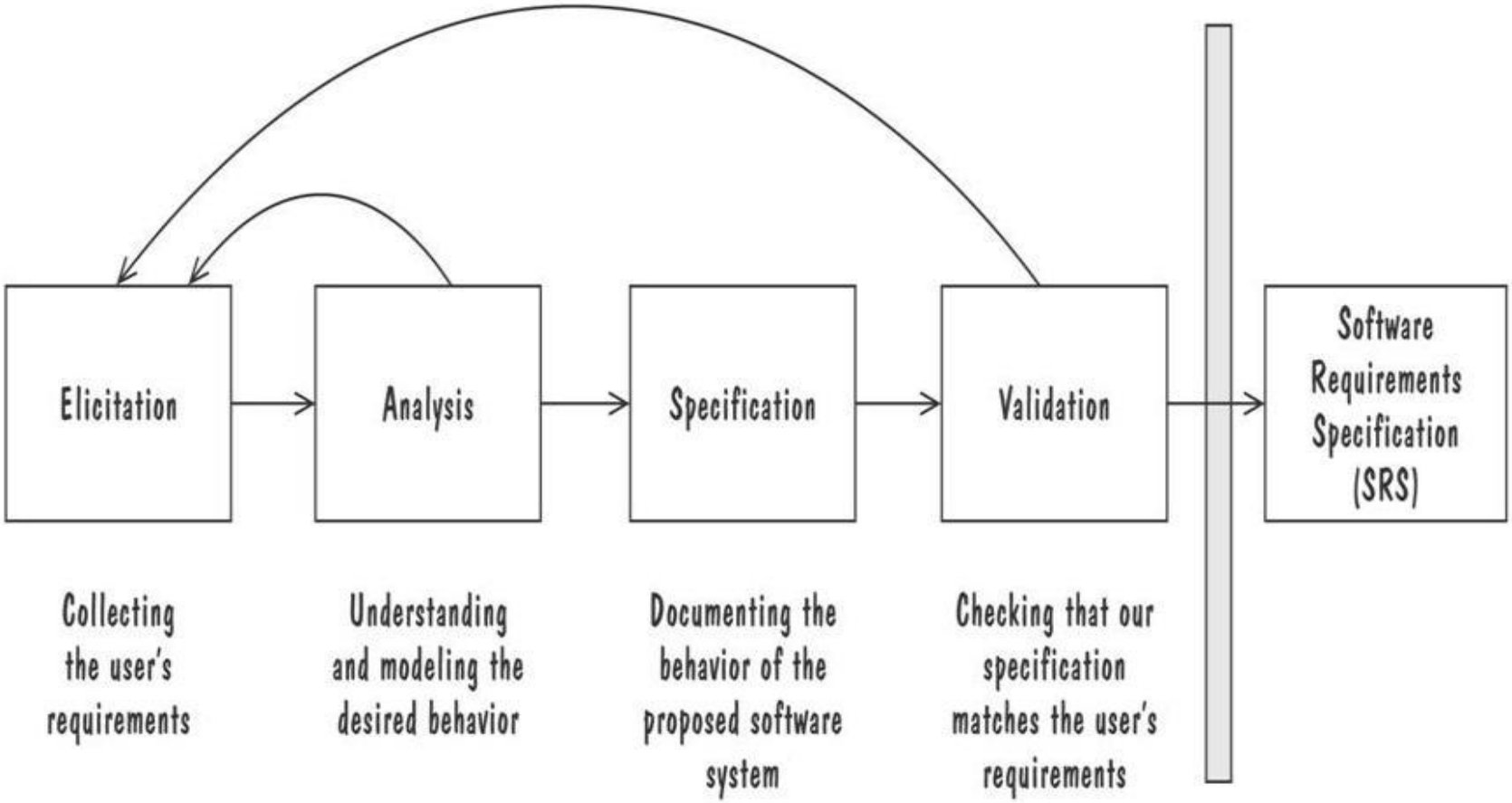
Use Cases (Overview)

Davide Falessi

Overview

- Use cases
  - Why use cases
  - Actors
  - Documenting use cases
  - Use case to requirements

Requirement Engineering Process Steps



[Source: Pfleeger & Atlee]

UML

- **Unified Modeling Language™ (UML®)** is a visual language for specifying, constructing, and documenting the artifacts of systems.
- **Complex** software designs are difficult for you to describe textually can readily be conveyed through diagrams using UML.
- Modeling provides three key benefits:
  - Visualization
  - Complexity management
  - Clear communication

UML

- You can use UML with all processes throughout the development lifecycle and across different implementation technologies.
- UML was approved by the Object Management Group™ (OMG™) a standard in 1997.
- Over the past few years, there have been minor modifications made to the language. UML 2 was the first major revision to the language.

Use Case

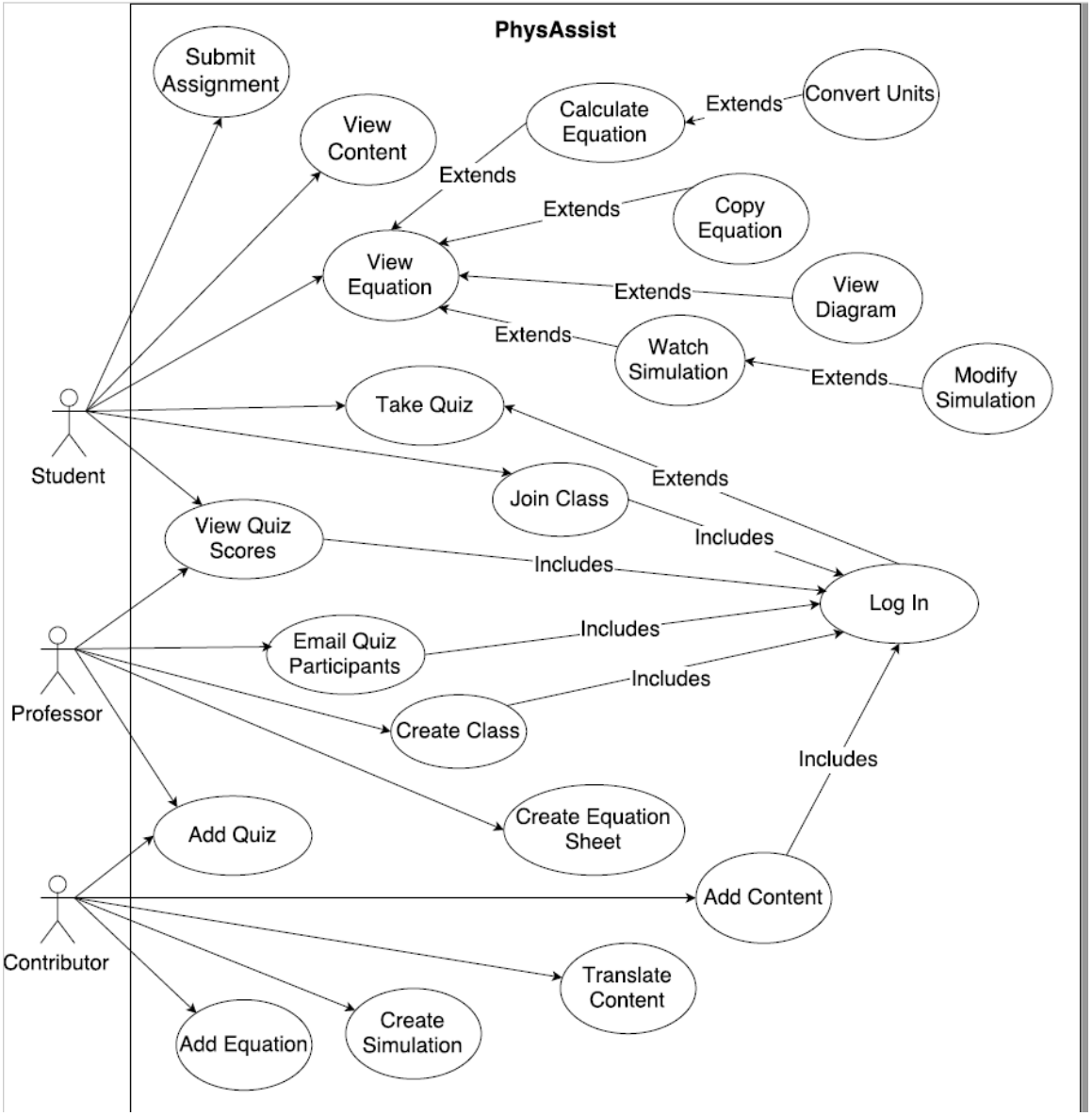
- (Product) use case: A description of a set of logically possible interactions between **an actor and a system** that results in **an outcome** that provides value to the actor. (Wieger, "Software Requirement")
- Use cases shift the perspective of requirements development to discussing what users of the system (actors) need to accomplish.
  - Contrast to asking users/customers what they want the system to do.
  - It is a way to break down the complexity of large system
- The goal of use case is to describe tasks that users will need to perform with the system.

Use Case Diagram

- **A high-level visual presentation of the user requirements**.

- Actors are modeled using “stick figures”.
- A use case in UML is drawn as an oval with a name that describes the interaction that it represents.
- A communication line connects an actor and use case to show actor participation in the use case.
- System boundary/scope is shown by drawing a box around the use cases, but keep the actors outside the box.

Use Case Diagram - Example



Use Case Actors

- A person, another software system, or a hardware device that interacts with the system to achieve a useful goal (Cockburn, 2001)
  - Can refer to user role:
    - Example: customer, buyer, requester.
    - A user can also assume multiple roles.

- Defines the system boundary – actor is not part of the system.

- Types of actor:
  - Primary: initiator of the interactions. E.g., User
  - Secondary: actors that the system needs assistance from. E.g., Legacy DB.

Finding Actors

- Identify:
  - User groups that require help from system to perform their tasks.
  - User groups that are needed to execute system's main function?
  - User groups that are required to perform secondary functions (e.g. system maintenance and administration)?
  - Any external hardware or software system that interacts with the system.
- For each actor, think about:
  - What or who the actor represents?
  - Why the actor is needed?
  - What interest the actor has in the system?

Source: Rational Software Corporation., "Guidelines: Actor"

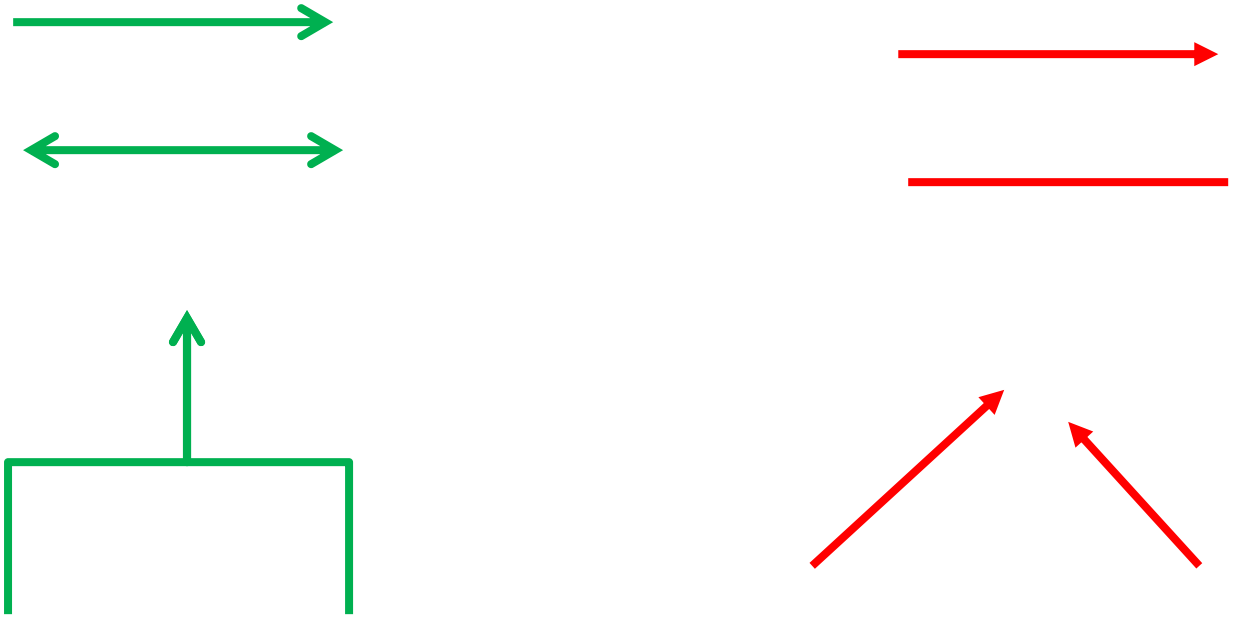
Identifying Use Cases

- Different approaches to identify use cases (Wiegers):
  - Identify the actors; then identify business processes in which each participates (i.e., focus on their goals).
  - Identify external events to which the system must respond, and then relate these events to participating actors and specific use cases.
  - Express business procedures in terms of specific scenarios, generalize scenarios into use cases, and identify actors involved in each use case.
  - Derive likely use cases from existing functional requirements. If some requirements don't trace to any use case, consider whether you really need them.

Re-using Use Case

- Whenever there's duplicate behavior between more than one use cases – there's a potential for re-use.
- Some use-case relationships:
  - "include": a generalization relationship by including the behavior described by another use case.
    - The "included" use case is usually shared by multiple use cases.
    - The result of the "included" use case is important to the base use case.
  - "extend": an extending use case continues the behavior of a base use case by inserting additional steps into the base use-case sequence. Common usages:
    - Part of a use case that is optional to the system behavior (it is executed only under certain conditions).

Layout guidelines



Common Mistakes

- Not using simple present
- Inconsistent arrows
- Actors as only humans
- Includes as extends
- Arrow direction of includes as extends