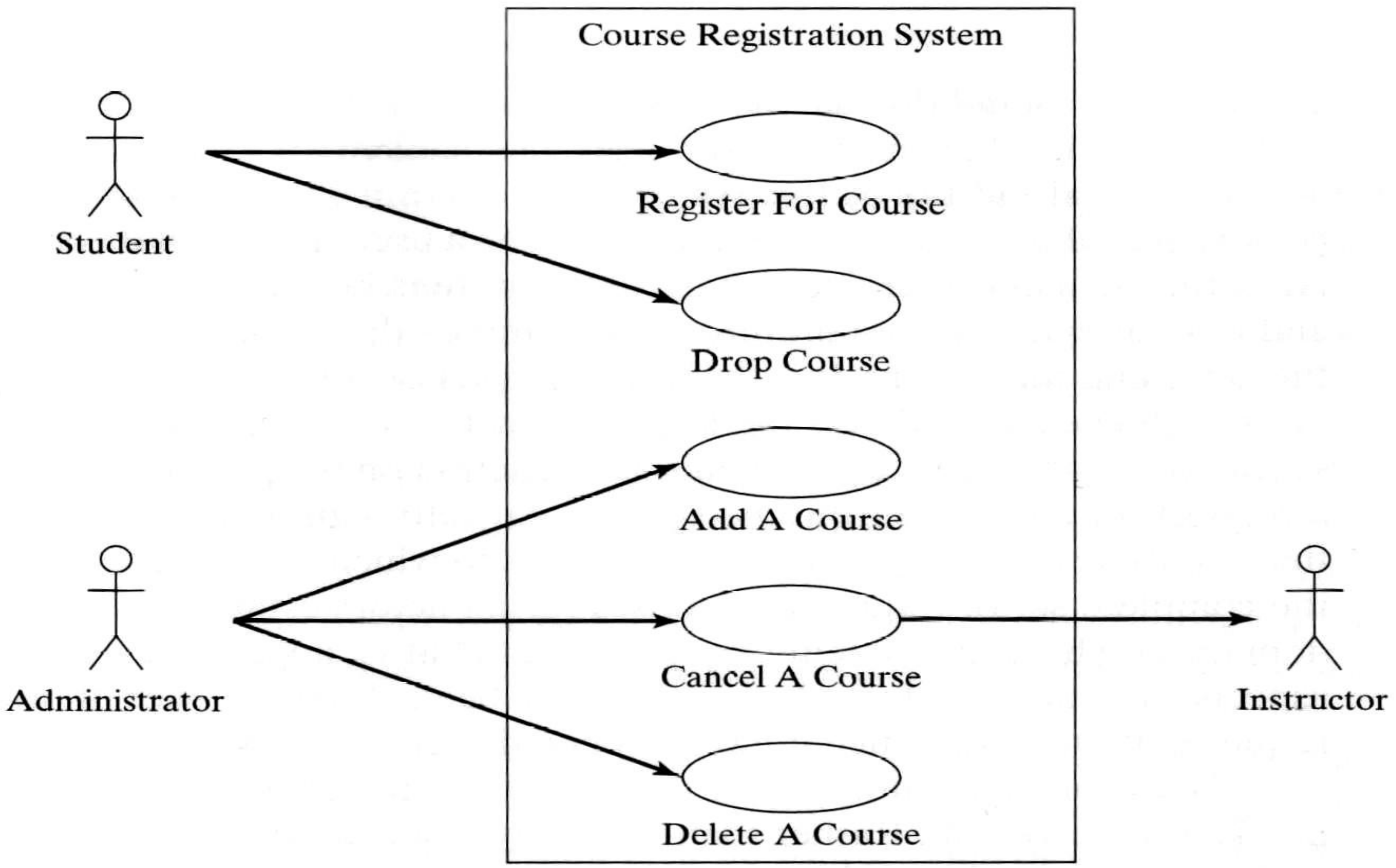
The Use-Case Diagram



Use-Case Diagrams (UCDs) (1/2)

- A use-case is…
  - a simplification of (a part of) a business process model
  - a ***set of activities*** within a system
  - presented from the point of view of the associated ***actors*** (i.e. those actors interacting with the system)
  - leading to an externally ***visible result***
- What is the model supposed to do?
  - offer a simplified and limited notation
  - allow other diagrams used to support (realise) it
  - combinatorial with prototypes as complementary information
  - not intended to model functional decomposition

Use-Case Diagrams (UCDs) (2/2)

Components: use-cases and actors

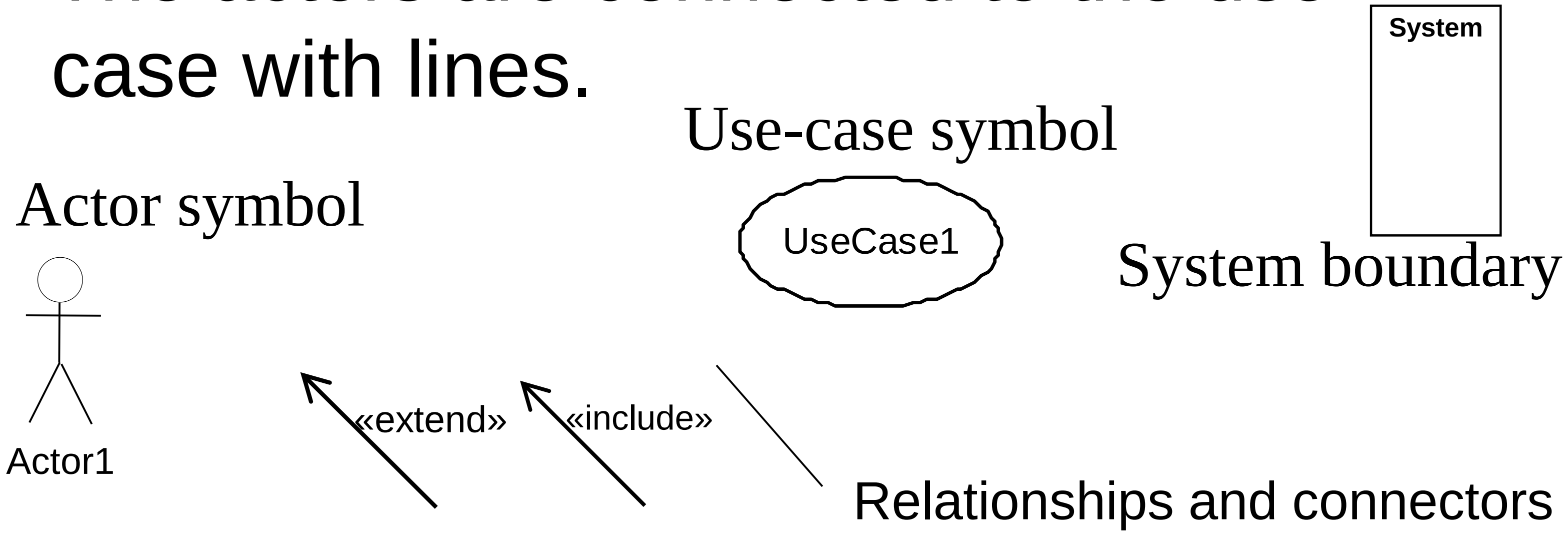
- a use-case <u>must always</u> deliver a value to an actor
- the aggregate of all use-cases is the system's complete functionality

Goals:

- consolidate system functional requirements
- provide a development synchronisation point
- provide a basis for system testing
- provide a basic function-class/operation map

UCD Components

- The use case itself is drawn as an oval.
- The actors are drawn as little stick figures.
- The actors are connected to the use case with lines.



UML Actors

- An actor
  - Is a class that forms a ***system boundary***
  - participates in a use-case
  - is not within our responsibility as systems analyst/s and/or designer/s
- Examples are
  - end-users (roles)
  - external systems (co-operations)
  - time related events (events)
  - external, passive objects (entities)

UML Actor Classification

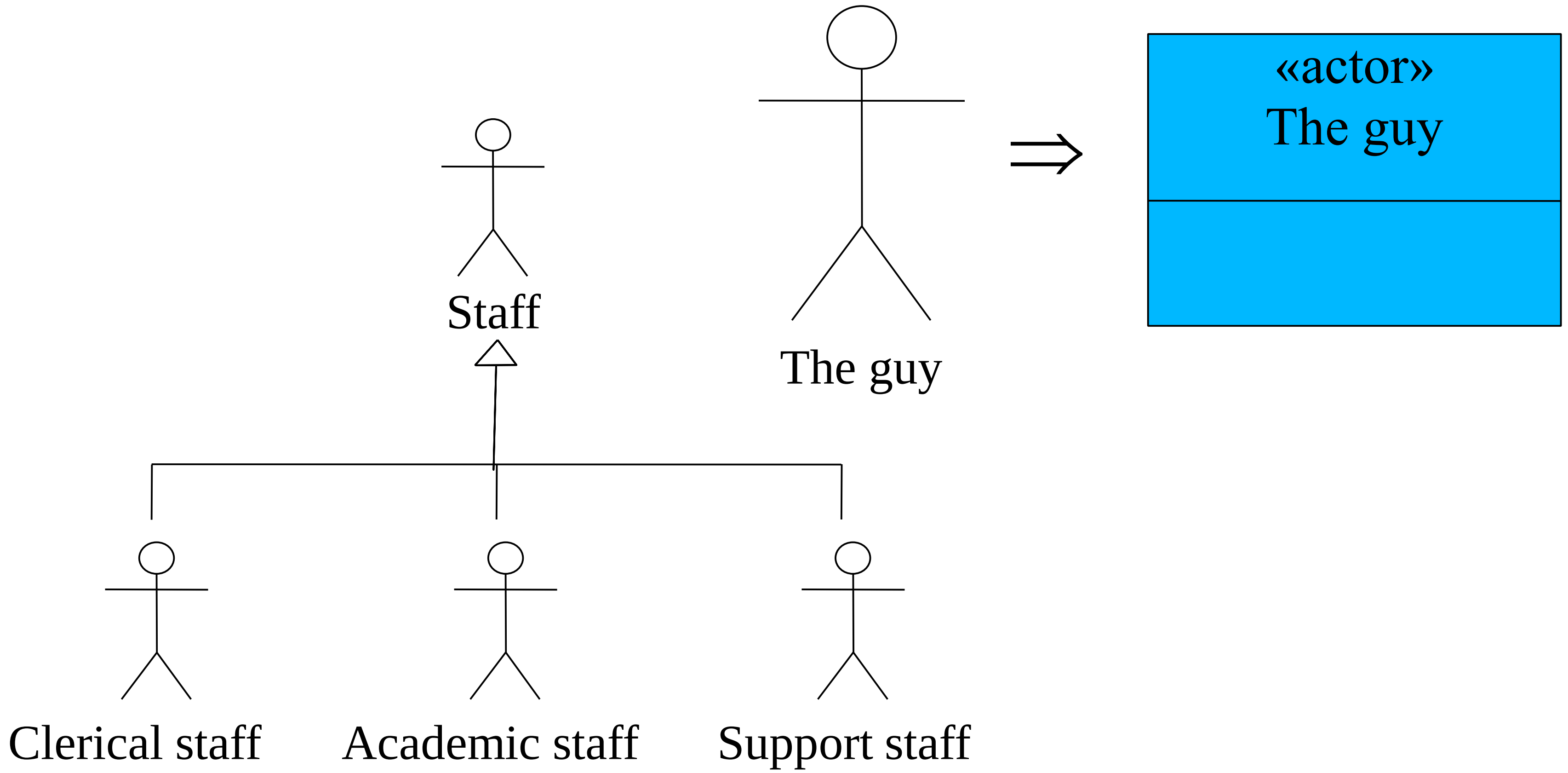
- A primary actor uses the system's primary functions (e.g. a bank cashier);
- A secondary actor uses the system's secondary functions (e.g. a bank manager, system administrator);
- An active actor initiates a use-case;
- A passive actor only participates in one or more use-cases.

Identifying UML Actors

Ask yourself the following questions:

- Who are the system’s primary users?
- Who requires system support for daily tasks?
- Who are the system’s secondary users?
- What hardware does the system handle?
- Which other (if any) systems interact with the system in question?
- Do any entities interacting with the system perform multiple roles as actors?
- Which other entities (human or otherwise) might have an interest in the system's output?

UML Actor Notation and Generalisation Examples



UML Use-Cases (UCs not UC Diagrams UCDs)

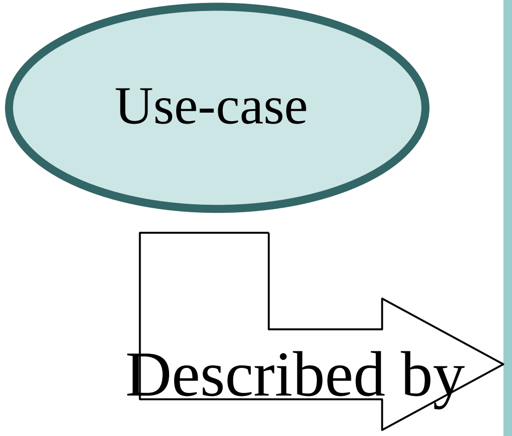
Definition: *"A set of sequences of actions a system performs that yield an observable result of value to a particular actor.“*

Use-case characteristics:

- Always initiated by an actor (voluntarily or
- involuntarily);
- Must provide discernible value to an actor;
- Must form a complete conceptual function.

(conceptual completion is when the end observable value is produced)

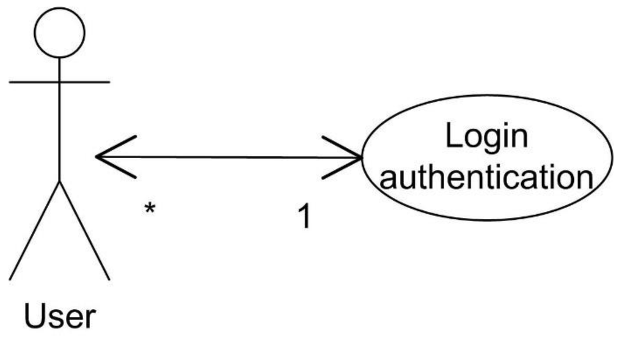
UC Description Criteria



Use-Case Number (ID) and Name

- actors
- pre- and post-conditions
- invariants
- non-functional requirements
- Behaviour modelled as:
  - activity diagram/s
  - decomposition in smaller UC diagrams
- error-handling and exceptions
- Rules modelled as:
  - activity diagram/s
- services
- examples, prototypes, etc.
- open questions and contacts
- other diagrams

UC Description Example



Example on the next slide

Example on the slide after the next

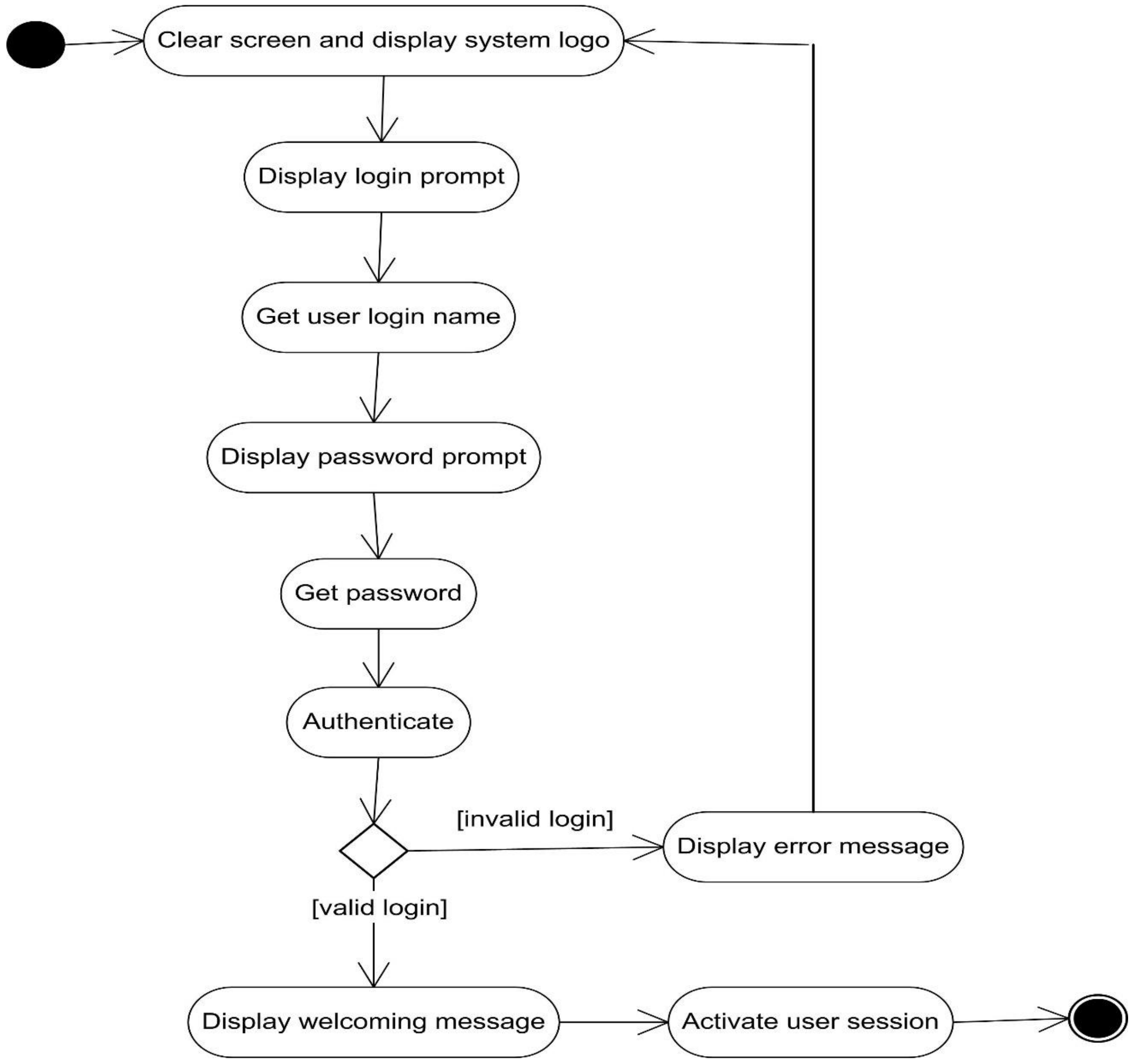
Example two slides further on

E.g. Collaboration diagram (tackled later on)

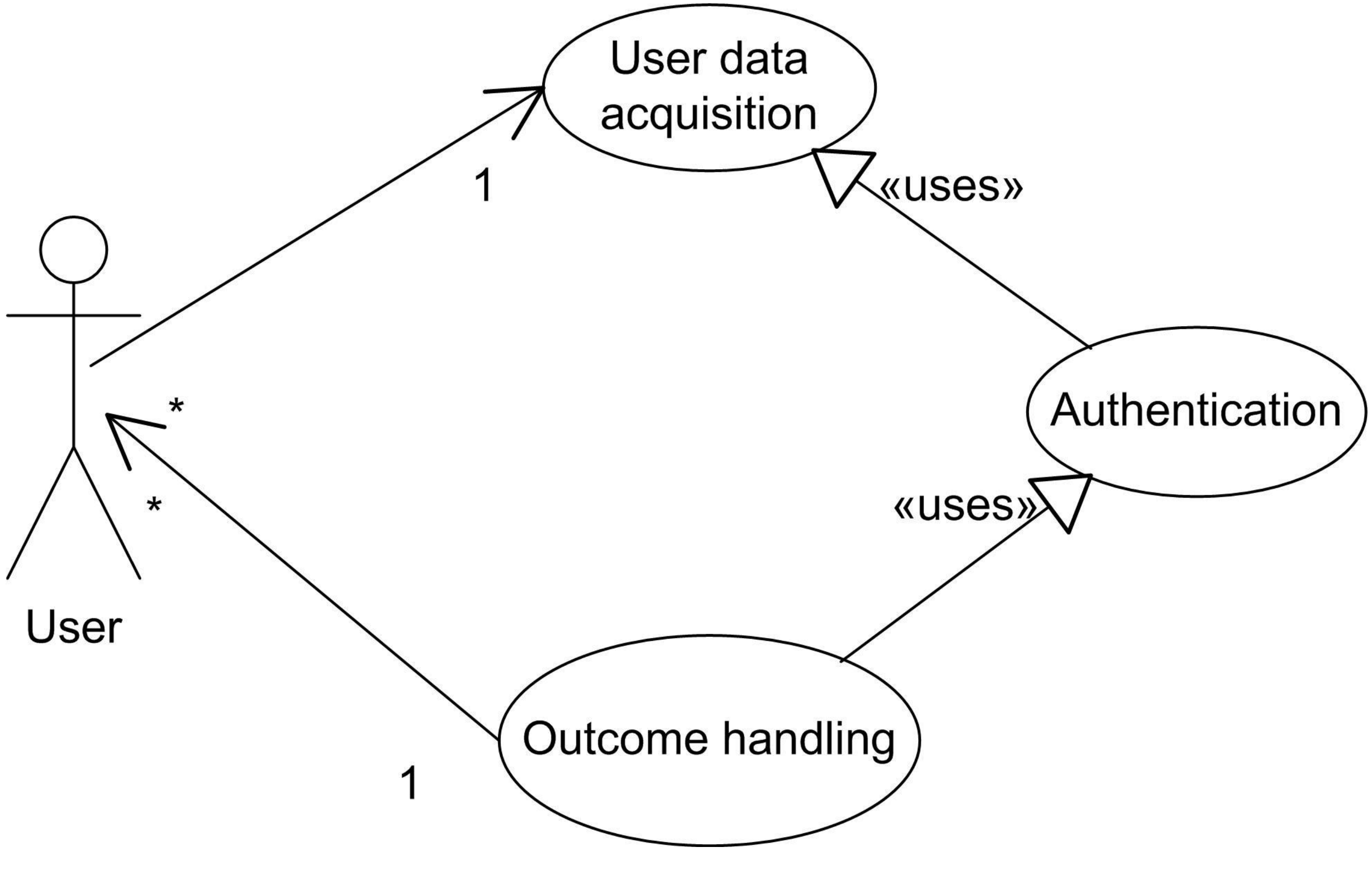
UC: Login authentication

- User
- Disable access - Enable access
- Logged in user = valid user
- Login delay; line security
- Behaviour modelled as:
  - activity diagram/s
  - decomposition in smaller UC diagrams
- Invalid login name; interrupt entry
- Rules modelled as:
  - activity diagram/s
- Log, pass prompts; authenticate
- examples, prototypes, etc.
- open questions and contacts
- other diagrams (realisations)

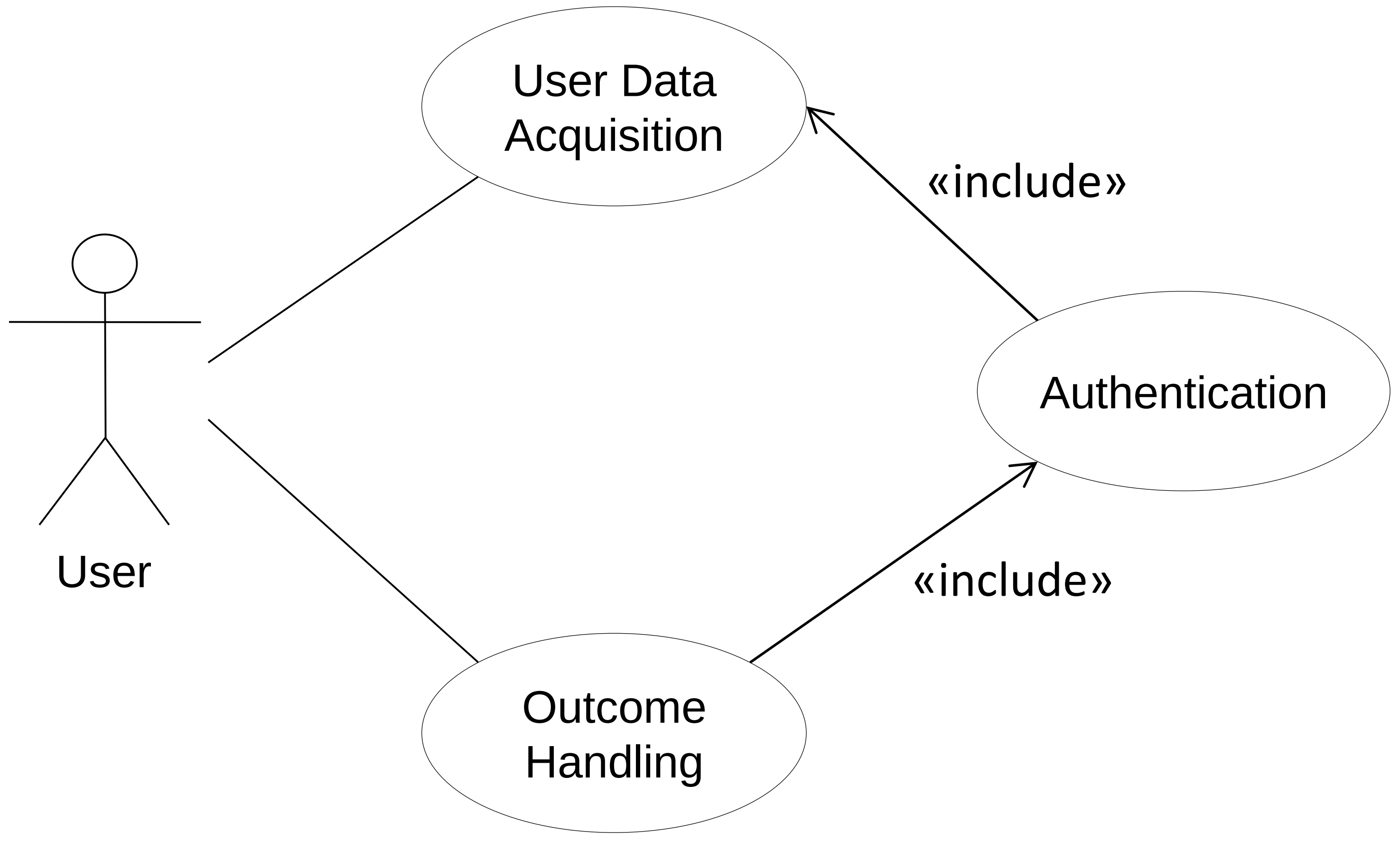
Activity Diagram from previous



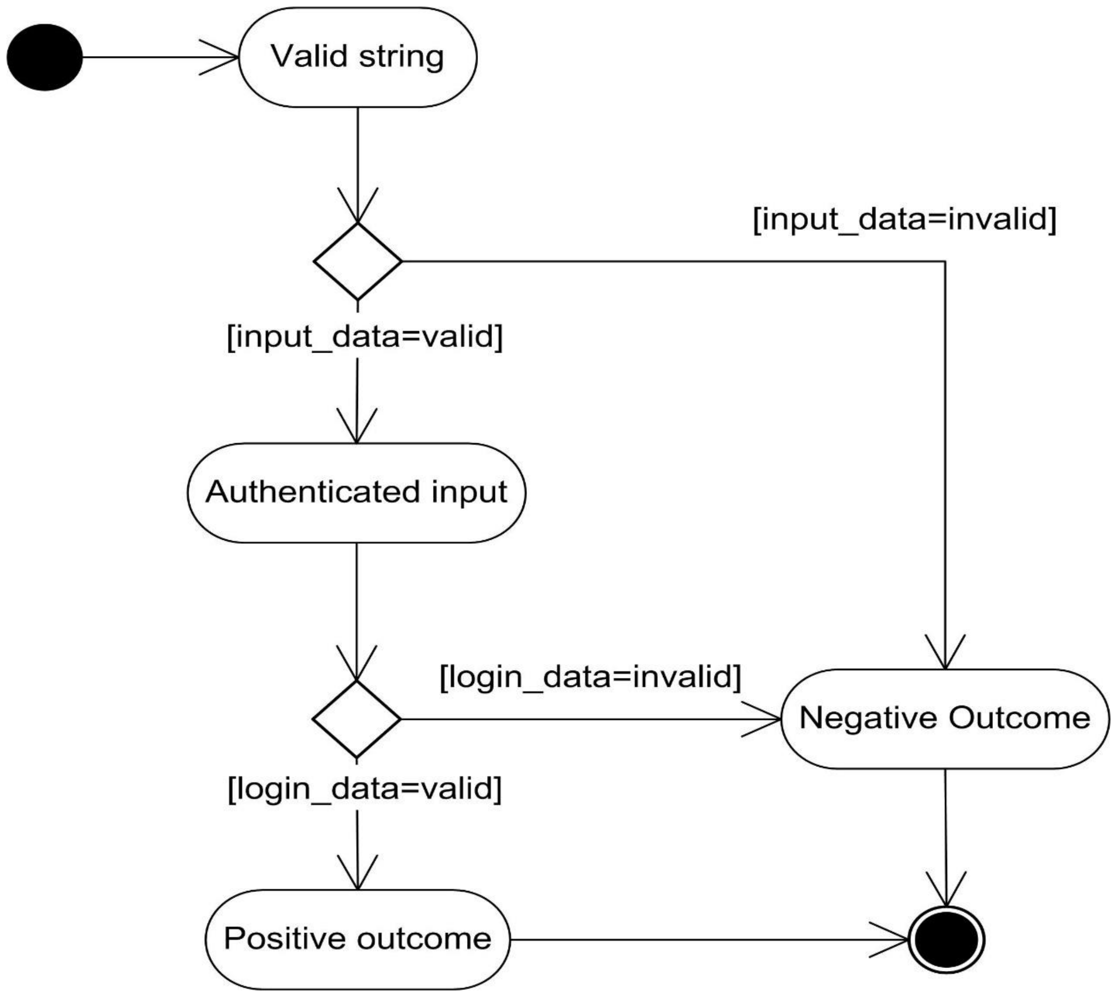
Sub-UCs to Login Example



Sub-UCs to Login Example



Rules Activity Diagram Example



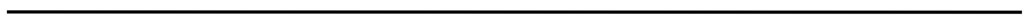
Consolidating UC Descriptions

Ask yourself these questions:

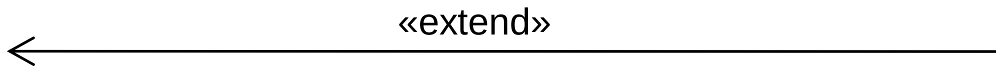
- Do all actors interacting with a given UC have communication association to it?
- Are there common roles amongst actors?
- Are there UC similarities?
- Are there special cases of a UC?
- Are all system functions catered for by UCs?

UCD Relationships (1/2)

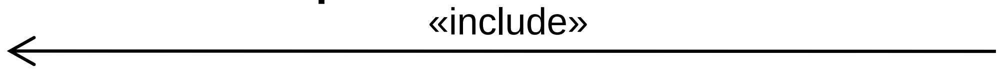
- Association relationship



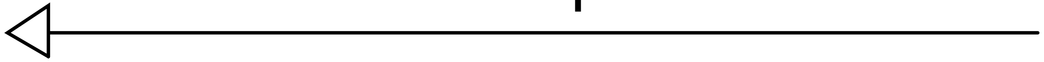
- Extend relationship



- Include relationship



- Generalisation relationship



UCD Relationships (2/2)

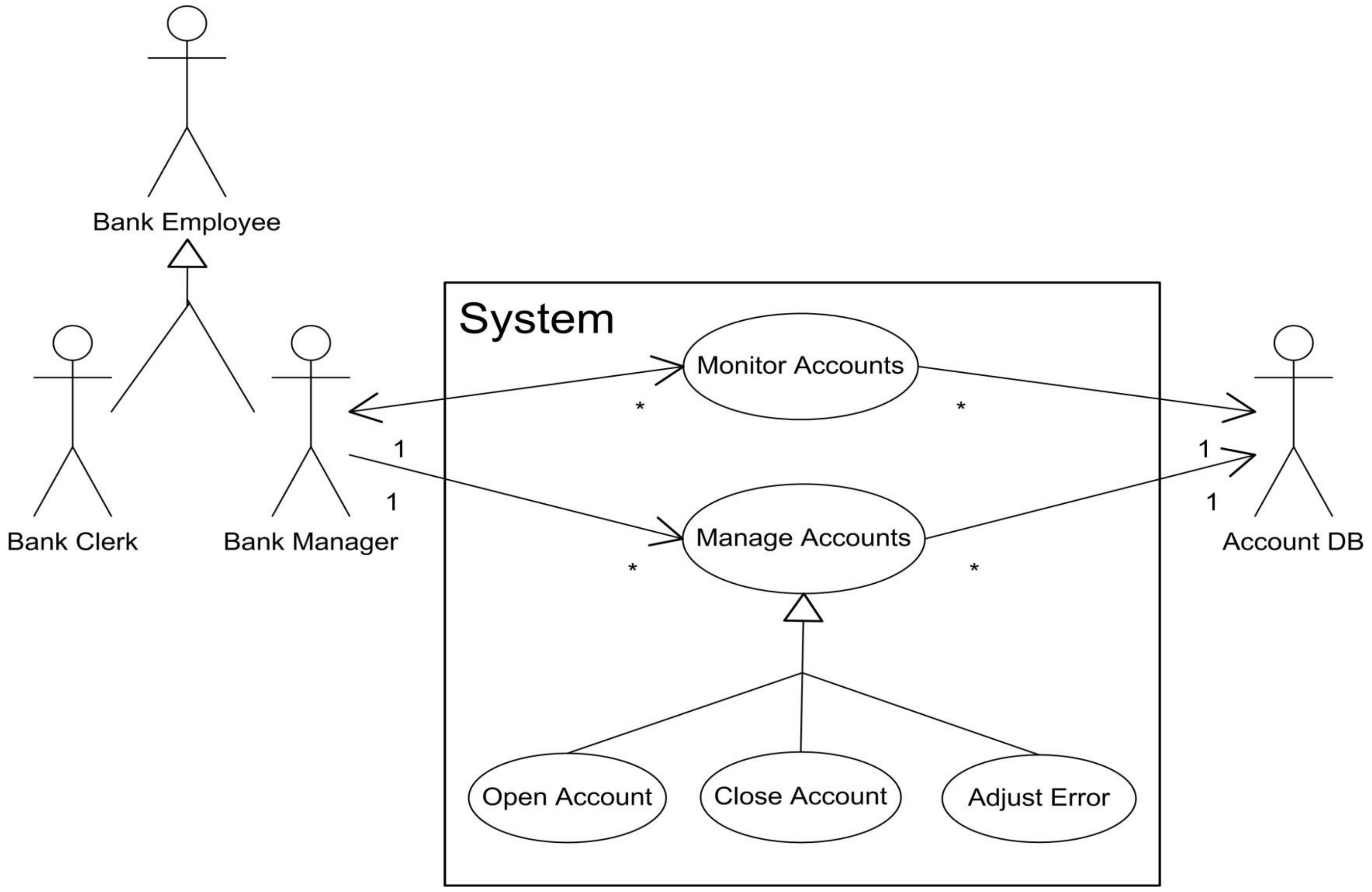
- Associations
  - Links actors to their UCs
- Use (or include)
  - Drawn from base UC to used UC, it shows inclusion of functionality of one UC in another (used in base)
- Extend
  - Drawn from extension to base UC, it extends the meaning of UC to include optional behaviour
- Generalisation
  - Drawn from specialised UC to base UC, it shows the link of a specialised UC to a more generalised one

UCD Definition Summary

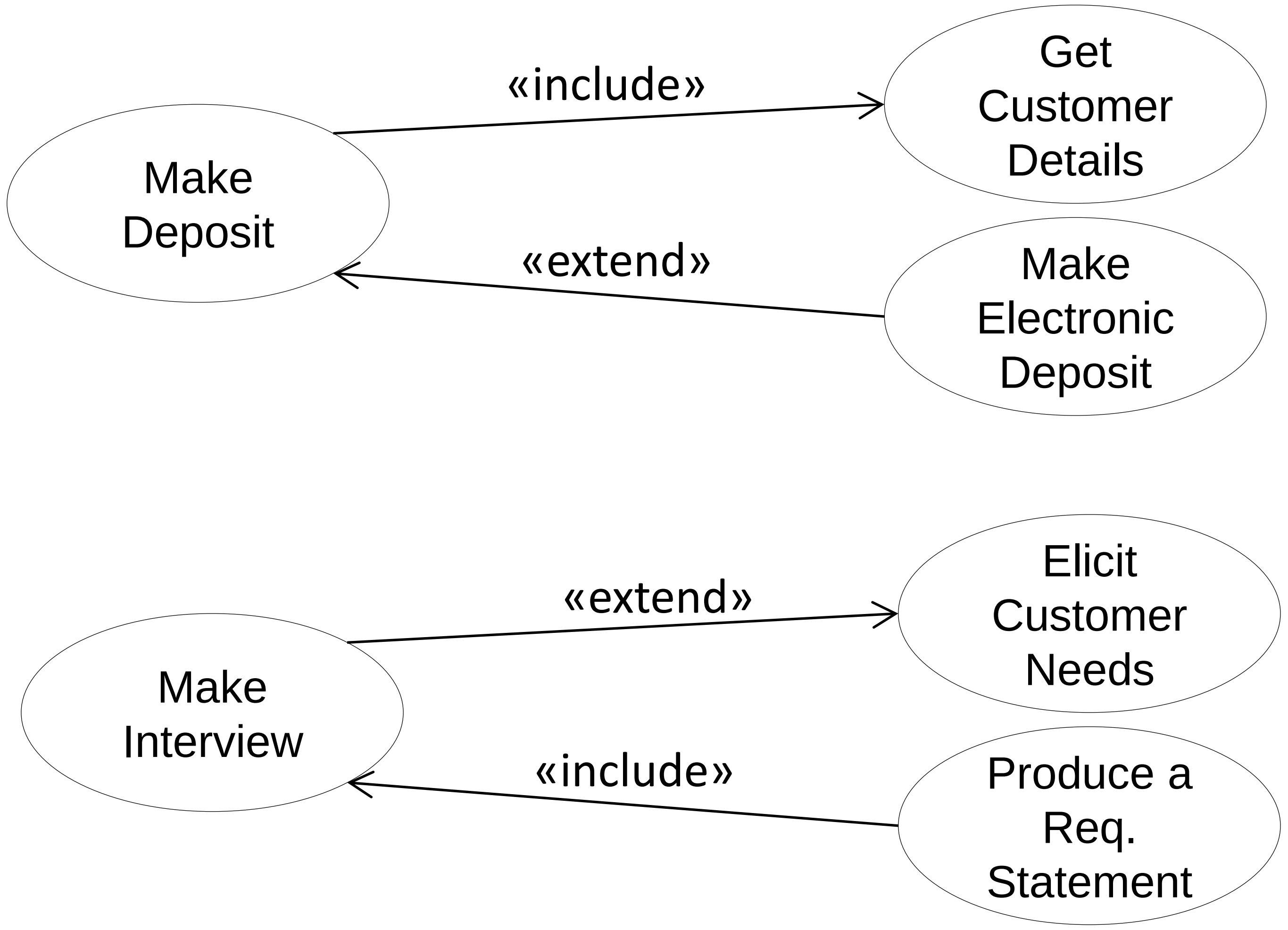
Use-Case diagrams:

- show use-cases and actors
- connected by “associations”
- refined by inheritance stereotypes
  - “uses”
    - re-use of a set of activities (use-cases)
    - partitioning of activities
    - points to the re-used use-case
  - “extends”
    - variation of a use-case
    - points to the standard use-case

UCD Relationship Example
(1/2)



UCD Relationship Example
(2/2)



What a UCD is - and what it isn't

- Attention focuser on the part of the business process that is going to be supported by the IS.
- It is the end-user perspective model.
- It is goal driven
- Helps to identify system services.
- Are not used as DFDs.
- Sequences, branching, loops, rules, etc. cannot (and should not) be directly expressed.
- Are often combined with activity diagrams, which serve as their refinement.

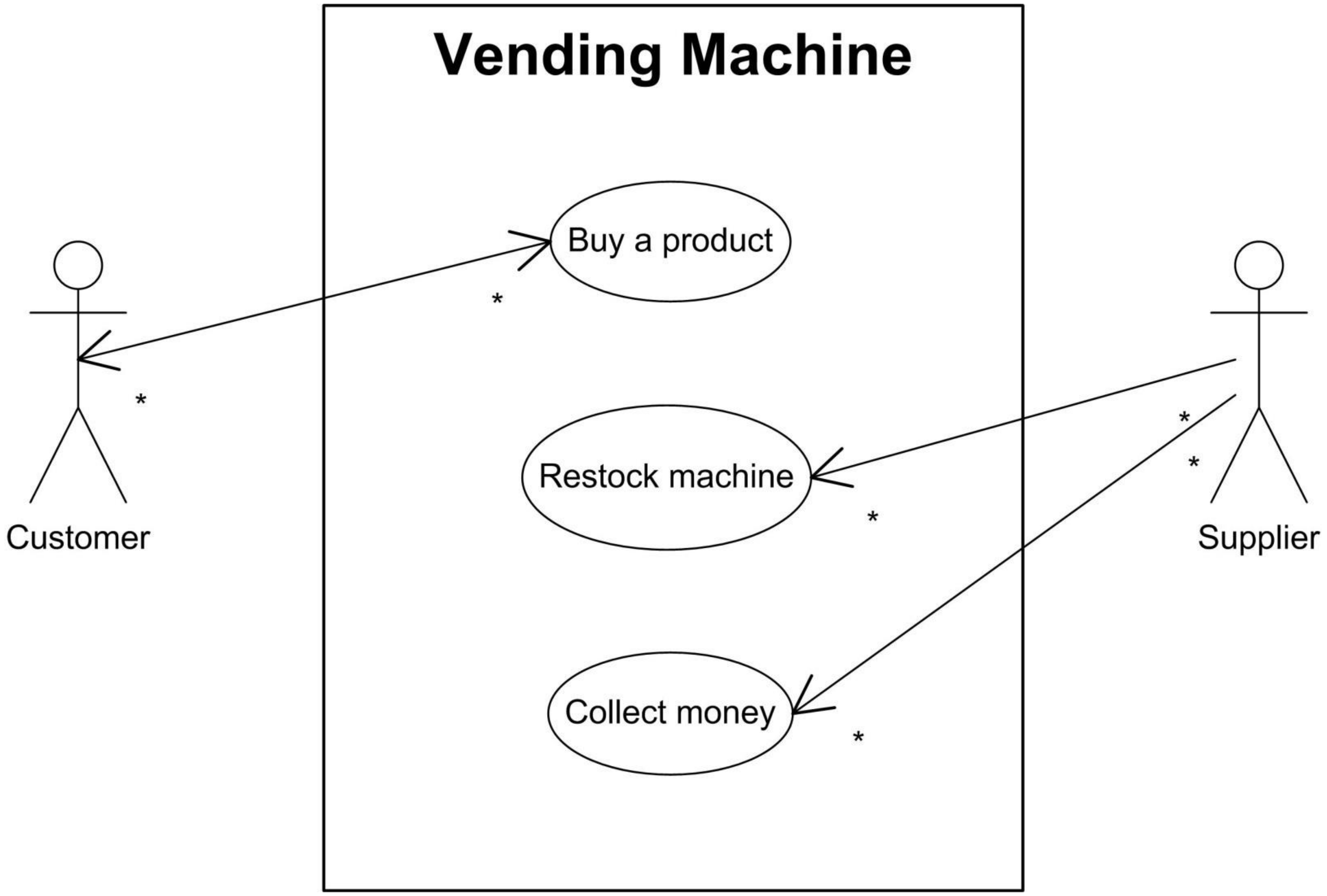
UCD Case Study (1/3)

Vending Machine

- After client interview the following system scenarios were identified:
  - A customer buys a product
  - The supplier restocks the machine
  - The supplier collects money from the machine
- On the basis of these scenarios, the following three actors can be identified:

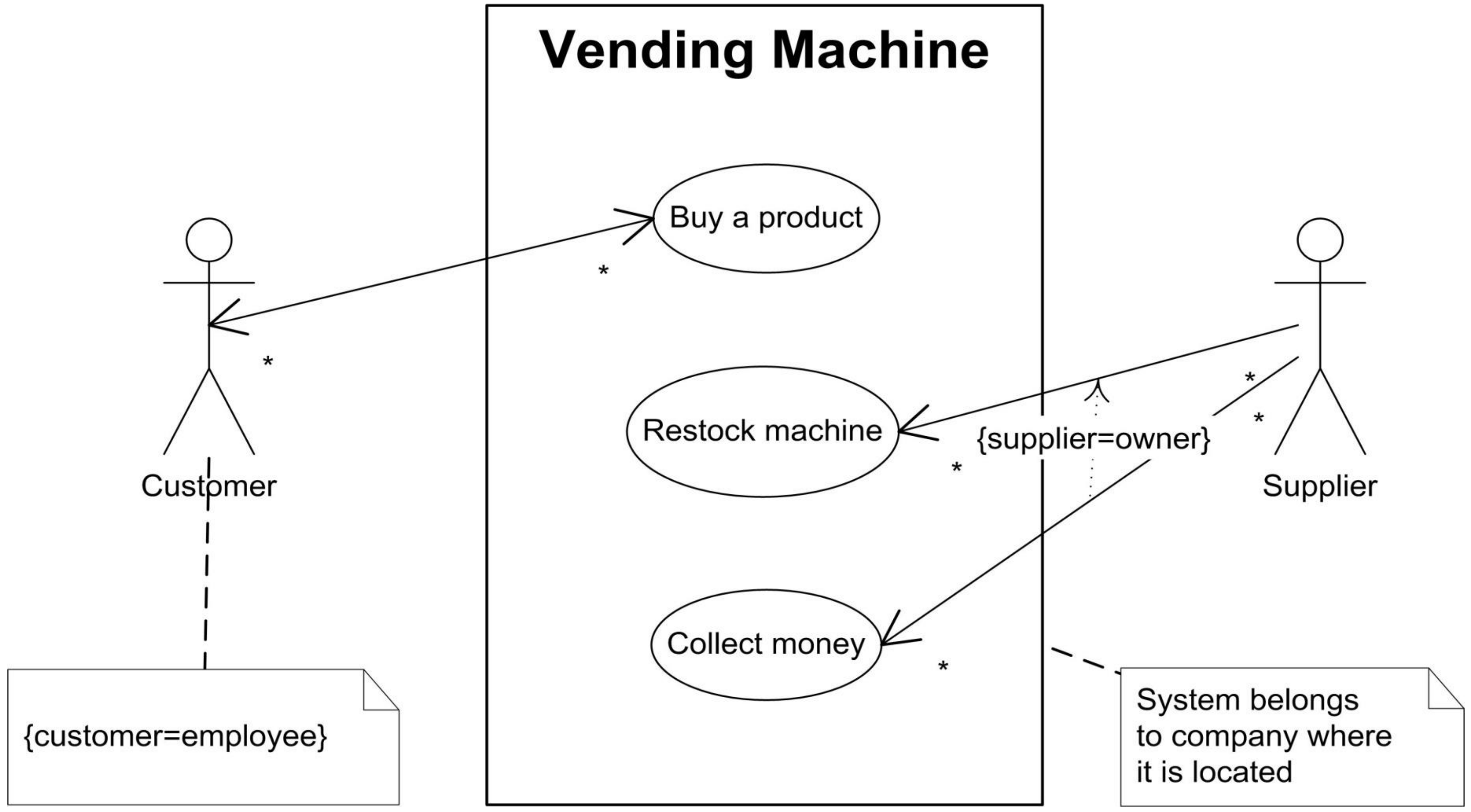
Customer; Supplier; Collector (in this case Collector=Supplier)

UCD Case Study (2/3)



UCD Case Study (3/3)

- Introducing annotations (notes) and constraints.



Testing UCs

- Verification
  - Confirmation of correct development according to system requirements.
- Validation *(only when working parts become available)*
  - Confirmation of correct system functionality according to end-user needs.
- Walking the UC
  - This is basically, interchangeable role play by the system developers.