SE463
Software Requirements Specification & Analysis

Business Use Cases

Readings:

Robertson, S. and Robertson, J., *Mastering the Requirements Process, 3ed.*, Chapter 4 "Business Use Cases"

AND

Larman, *Applying UML and Patterns*, Chapter 6: "Use Cases"

Furniture Design Exercise

Working in groups of 3 and 4, design an innovative piece of furniture that can be manufactured by individuals who have physical or mental impairments, disabilities, or conditions.

Name Requirements:

It can serve as a chair, footstool, room divider, coffee table, bookshelf, and end table.

Functional Requirements:

It can be used to sit on, prop feet on, divide rooms, and put books, lamps, magazines, and knickknacks on.

Constraints:

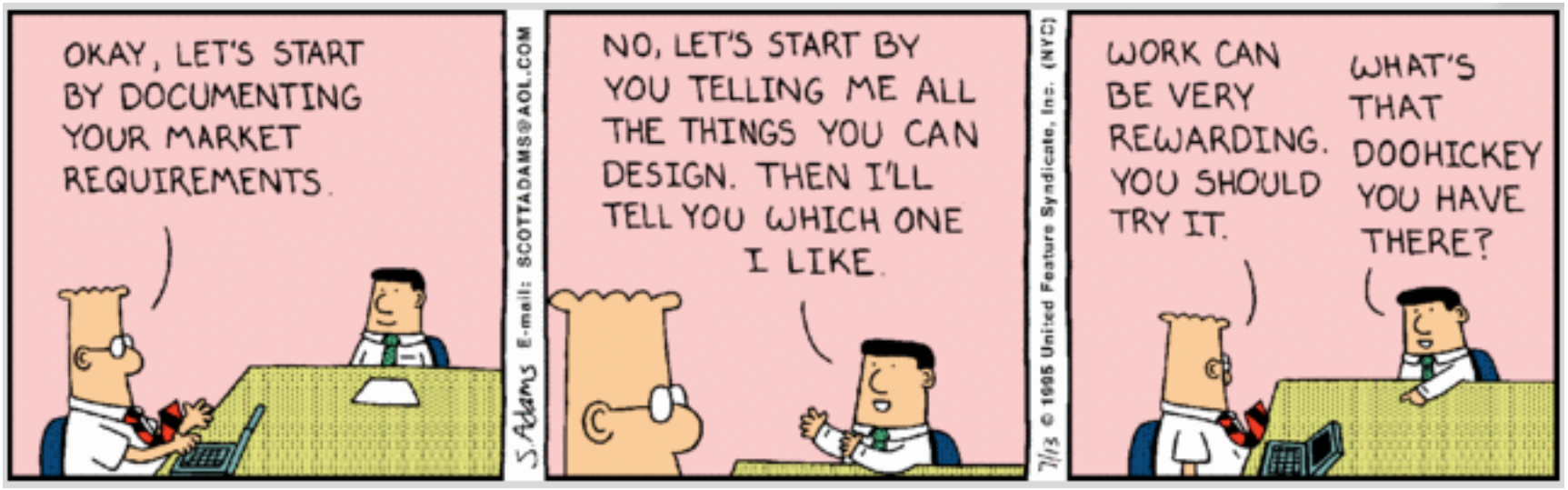
cost (must sell for under $50)

shipping (must ship unpackaged without fear of damage)

Evaluation Criteria:

innovative

Business Requirements



"If your software does not have to satisfy a need, then you can build anything.

However, if it is meant to satisfy a need, then you have to know what that need is to build the right software."

Robertson, Robertson, *Mastering the Requirements Process*, *3ed.,* 2012

Business Use Cases

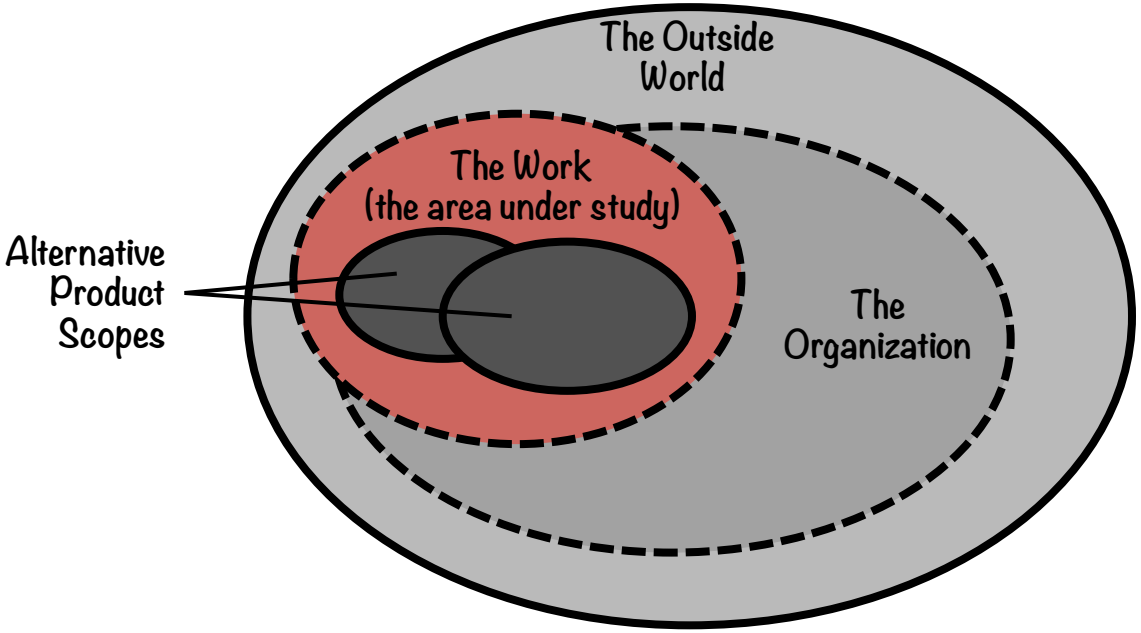
Understanding the work – understand the work that is to be improved before offering a solution or product.

Business Use Cases – decompose the work into work-pieces to manage complexity

Use Case Models – express business use cases in a manner that is easy for all stakeholders to visualize

- Big picture view of the problem

Understanding the Work



Robertson, Robertson, *Mastering the Requirements Process,* 2012, Figure 3.3.

Start by understanding the work that the owner does (or wants to do).

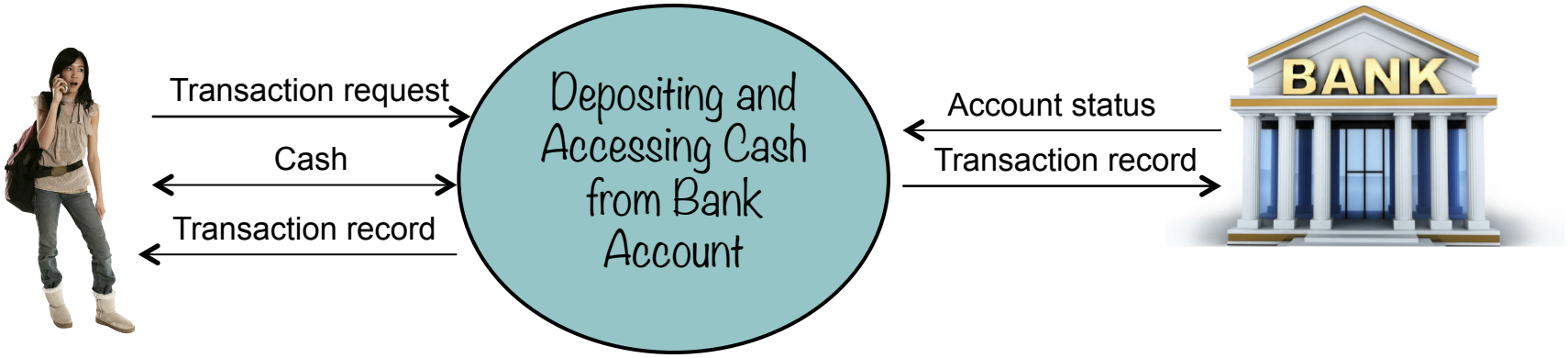
Have to understand the work itself, before determining what kind of product will improve the work.

Understanding the Work

Goal is to understand *what* the existing Work is

- not *how* it is implemented
- focus on externally observable behaviour
- abstract inputs and outputs

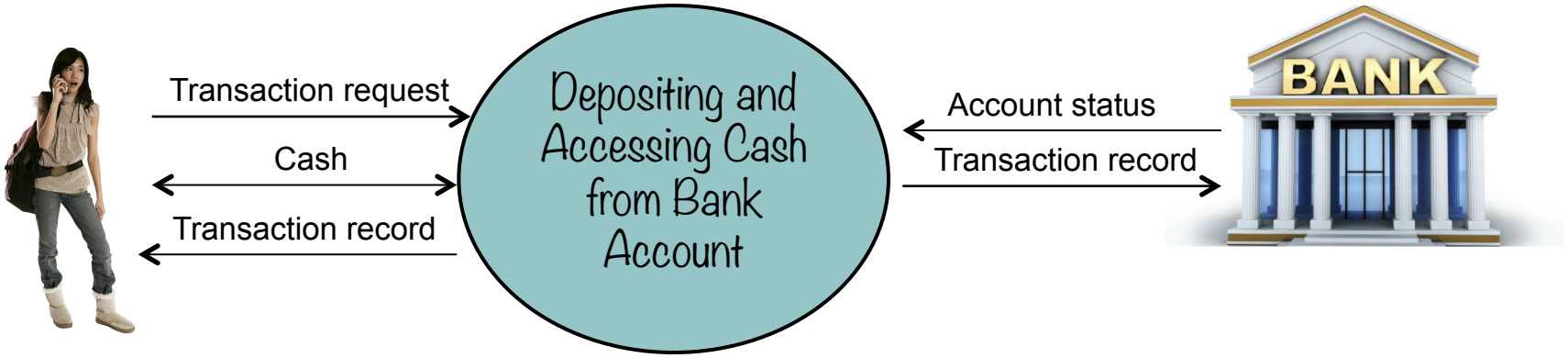
Example: Context diagram for Banking



Scope of Study

The Work to be studied must include

- anything that you are permitted to change
- anything you need to understand to decide what to change
- anything that can be affected by your product.



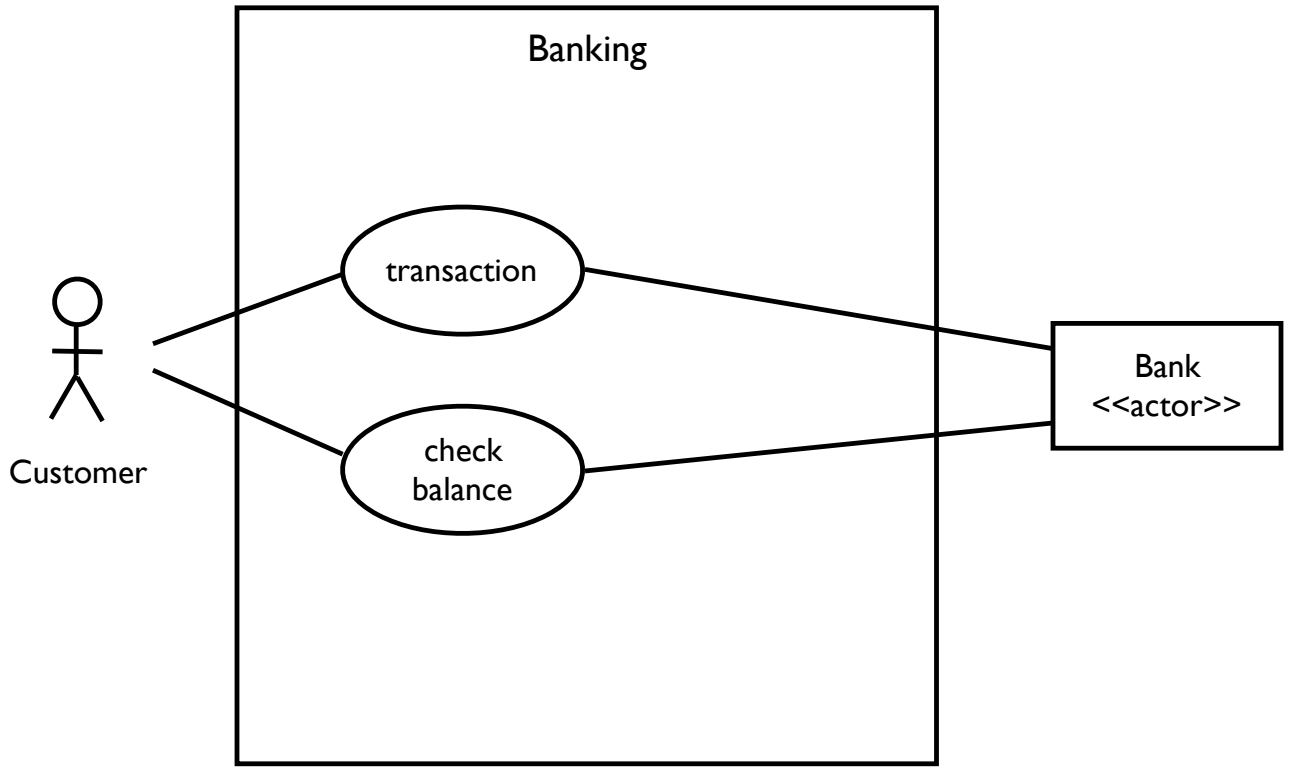
Business Use Cases

Decompose The Work into vertical slices to reduce complexity.

Each slice is called a use case

- represents some end-to-end functionality
- triggered by an external event (e.g., from adjacent system)
- captures a complete response to a triggering event
- use cases are (ideally) orthogonal to one another

Business Use Case

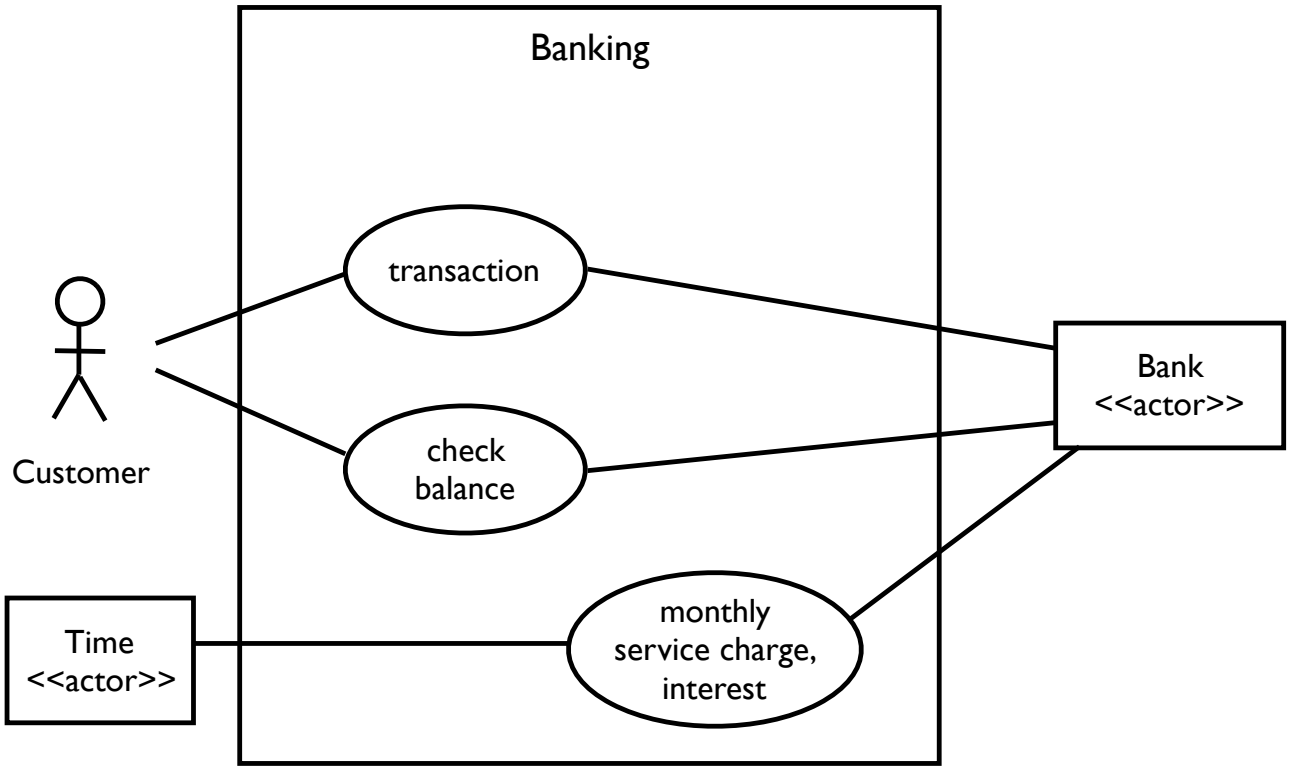


Captures:

- scope of Work
- adjacent systems
- use cases (= business events + responses)
- interactions

Time-Triggered Use Case

Time-triggered use cases are activated when a date or time comes to pass.



Example: Park User Fees

Suppose that the city of Waterloo decides to raise funds by instituting users fees for public parks.

Requirements:

R1: Collect $1 fee from each user on entry to the park.

R2: Ensure that anyone who has paid may enter the park.

R3: Ensure that no one may enter park without paying.

Example: Patient Monitoring System

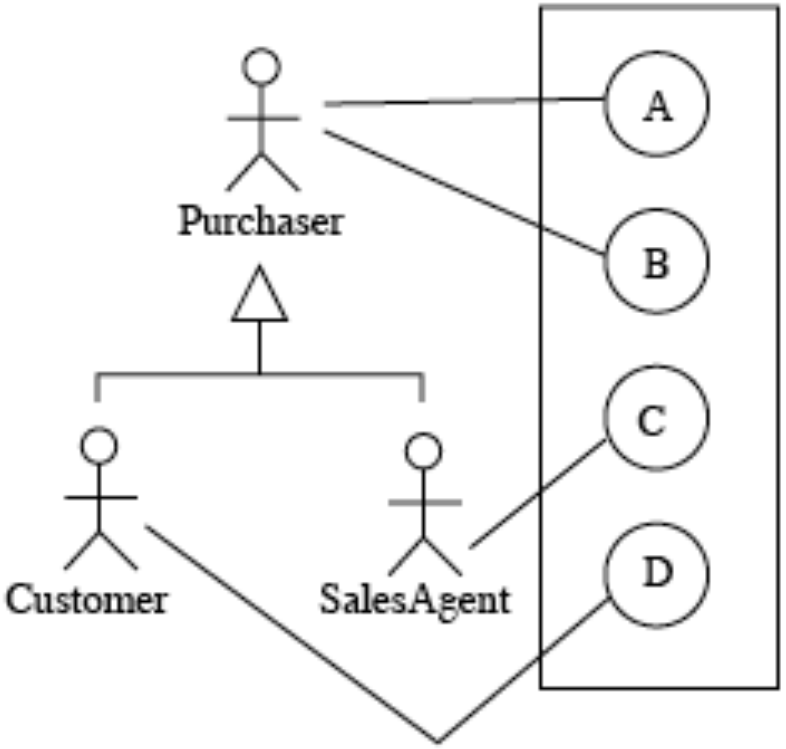
Patients in an intensive-care ward in a hospital are monitored by electronic analog devices attached to their bodies by sensors of various kinds.

- Through the sensors, the devices measure the patient's vital factors: pulse rate, temperature, blood pressure, and so on.
- A program is needed to read the factors, at a frequency specified for each patient, and store them in a database.
- The factors read are to be compared with safe ranges specified for each patient, and readings that exceed the safe ranges are to be reported by alarm messages displayed on the screen of the nurse's station.

Stevens, Myers, and Constantine, *IBM Systems Journal*, 1974

Actor Generalization

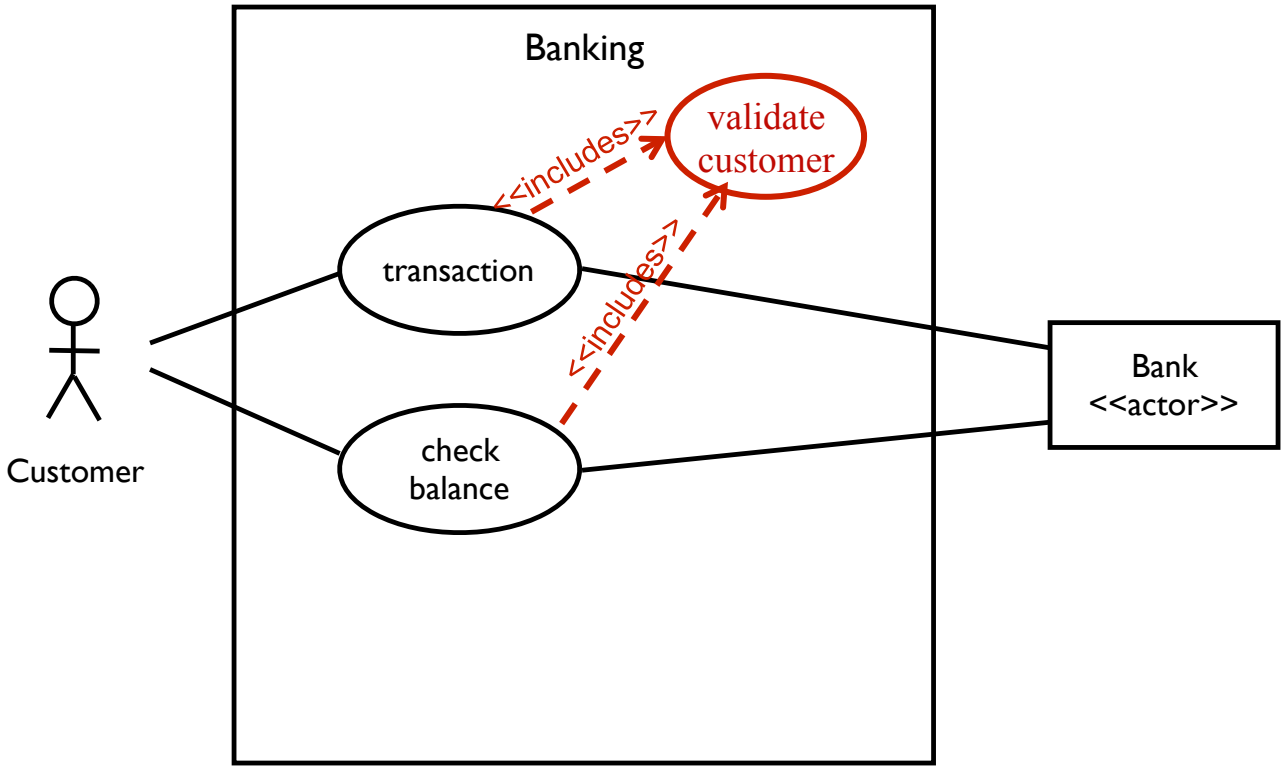
- Use actor generalization when actors have common *interesting* behaviour
  - i.e., they interact with many of the same use cases
- Factor out common behaviour as an abstract actor
  - Children inherit all relationships with use cases of the parent



«include»

- <<include>> - a sub use case that is used within multiple other use cases (like a procedure call)
  - Avoids repetition of the same steps in multiple use cases, improving readability
  - Specify point of inclusion in base use case
  - When sub use case completes, control returns to base use case

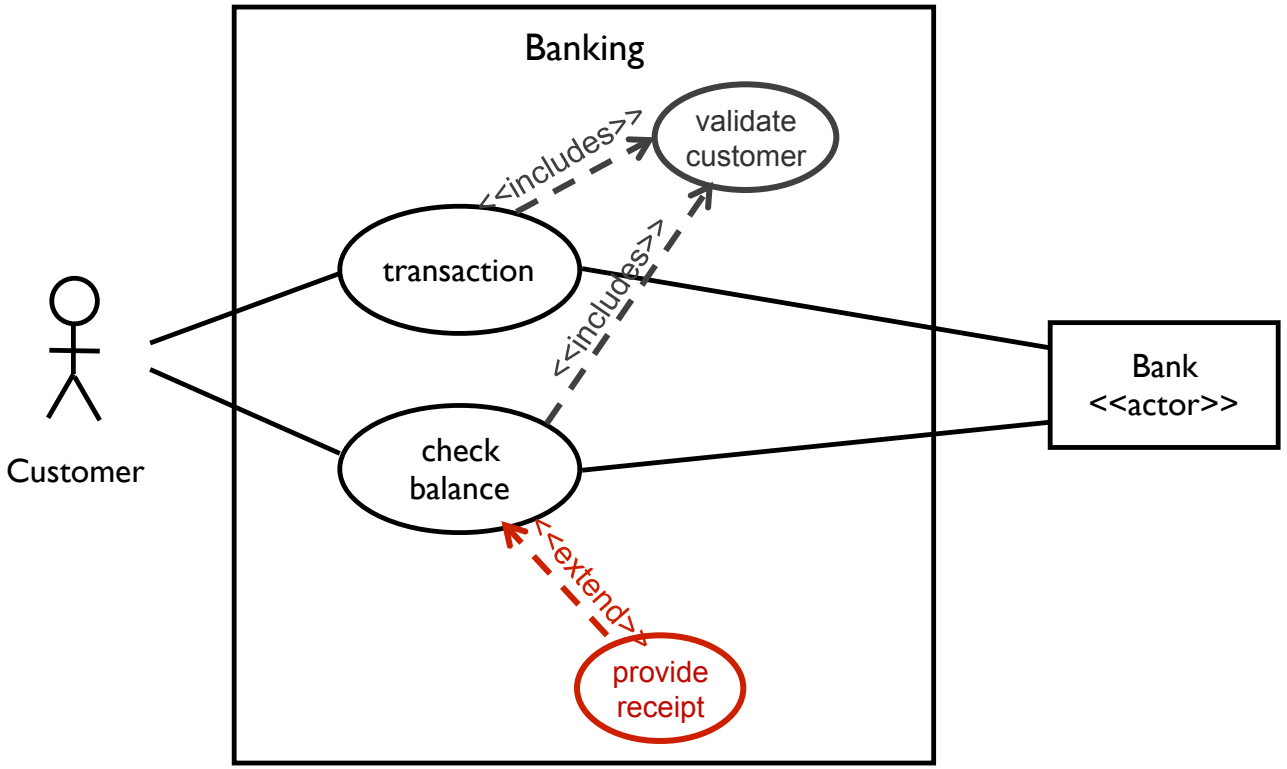
Example



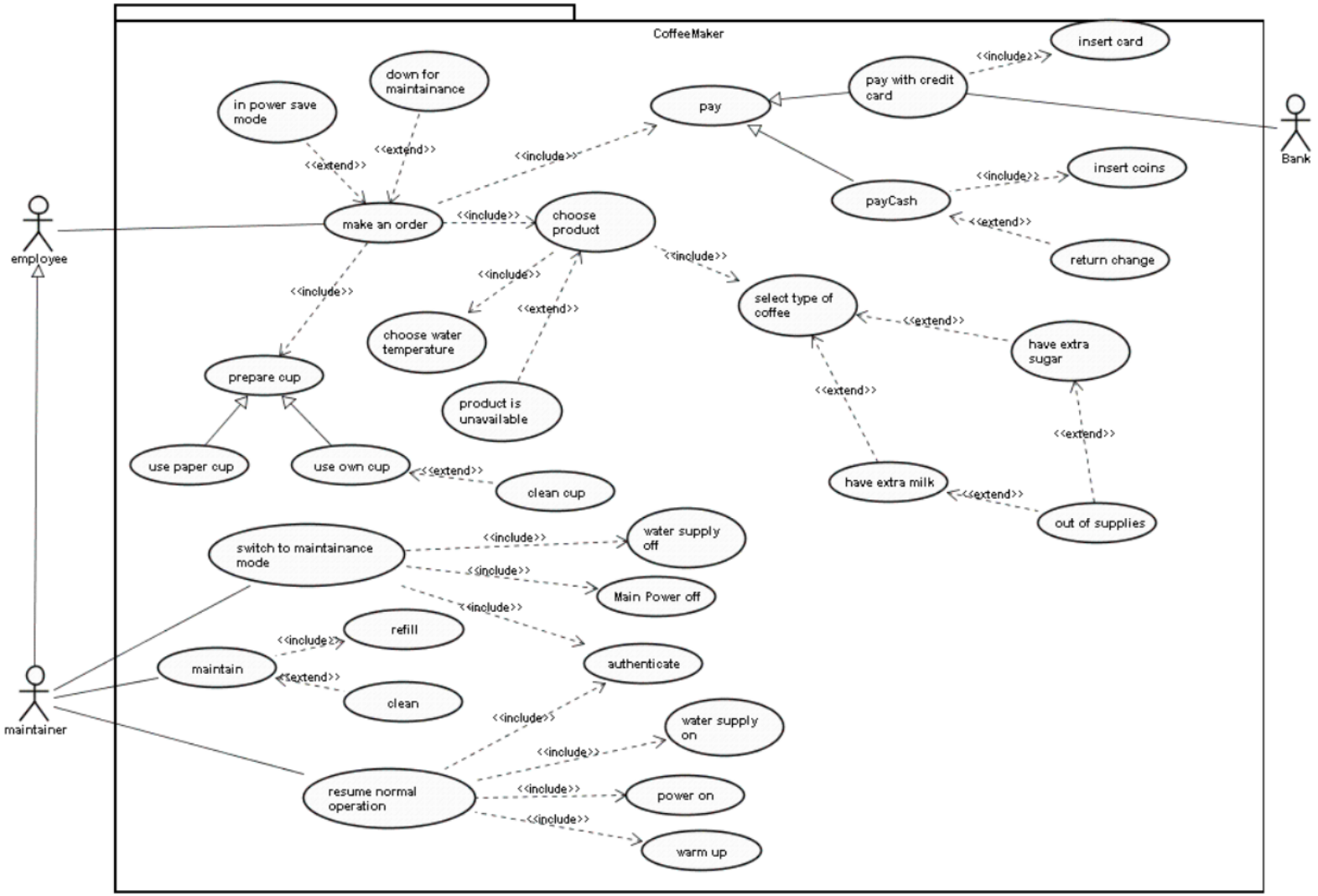
«extend»

- <<extend>> - a sub use case that extends or replaces the end of an existing use case
  - Base use case has hooks where it can be extended
  - Unlike «include», base case is complete without extension use case

Example



Bad Example (from the Web)



http://www.iai.uni-bonn.de/III/lehre/vorlesungen/SWT/OOSC06/exercises/exercise2.html

Summary

Before eliciting the requirements of the future system, one must

- Understand the current Work (to be improved)
- Decompose the work into business use cases (to reduce complexity)
- Capture all business use cases in a comprehensive Use Case Model