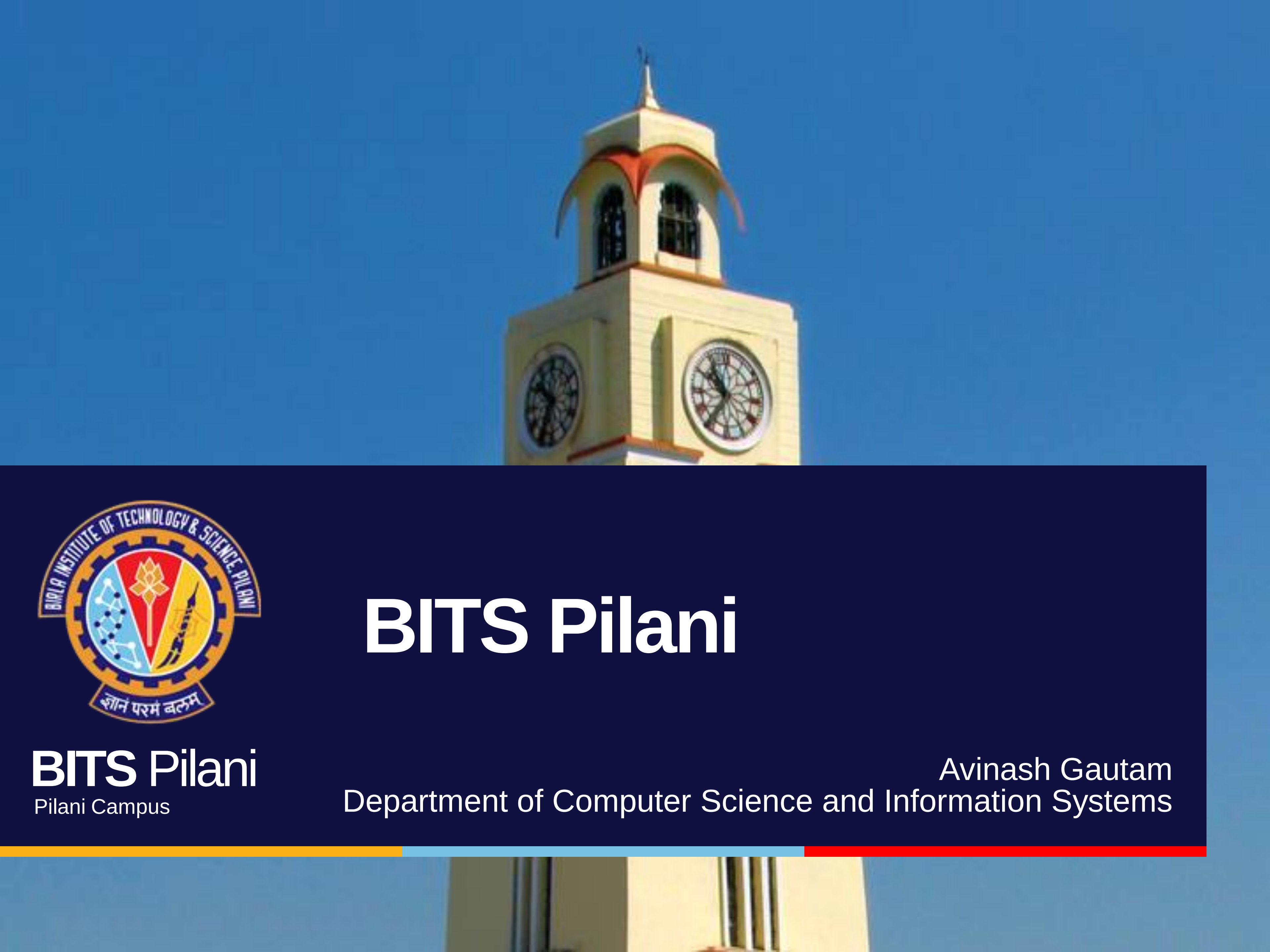


BITS Pilani

BITS Pilani
Pilani Campus

Avinash Gautam
Department of Computer Science and Information Systems

Use Case Model

Observable Result of Value

- "A key attitude in use case work is to focus on the question 'How can using the system provide observable value to the user, or fulfill their goals?', rather than merely thinking of requirements in terms of a list of features or functions"
- Focus on <u>business process</u>, not technology features

Black Box Use Cases

- Focus on "what", not "how"
- Good: *The system records the sale*
- Bad: *The system writes the sale to a database*
- Worse: *The system generates a SQL INSERT statement…*

Premature design decisions!

Analysis vs. Design = "What" vs. "How"

Degrees of Use Case Formality

- Brief: one-paragraph summary, main success scenario
- Casual: multiple, informal paragraphs covering many scenarios
- Fully-Dressed: all steps and variations written in detail with supporting sections
  - See: http://alistair.cockburn.us/usecases/usecases.html

NextGen POS Example

- A fully-dressed example in your textbook

Fully-Dressed Format

- Primary Actor
- Stakeholders & Interests
- Preconditions
- Success Guarantee (Postconditions)
- Main Success Scenario (Basic Flow)
- Extensions (Alternative Flows)
- Special Requirements
- Technology and Data Variations
- Frequency of Occurrence
- Open Issues

Details…

- Stakeholders and Interests
  - Important: defines what the use case covers ("all and only that which satisfies the stakeholders' interests")
- Preconditions
  - What must always be true before beginning a scenario in the use case
    (e.g., "user is already logged in")

Details…[2]

- Success Guarantees (Postconditions)
  - What must be true on successful completion of the use case; should meet needs of all stakeholders
- Basic Flow
  - Records steps: interactions between actors; a validation (by system); a state change (by system)

Details…[3]

- Extensions (Alternative Flows)
  - Scenario branches (success/failure)
  - Longer/more complex than basic flow
  - Branches indicated by letter following basic flow step number, e.g. “3a”
  - Two parts: *condition*, *handling*
- Special Requirements
  - Non-functional considerations (e.g., performance expectations)

Details...[4]

- Technology & Data Variations
  – Non-functional constraints expressed by the stakeholders
  (e.g., "must support a card reader")

Identify Use Cases

- Capture the specific ways of using the system as dialogues between an actor and the system.
- Use cases are used to
  - Capture system requirements
  - Communicate with end users and Subject Matter Experts
  - Test the system

Specifying Use Cases

- Create a written document for each Use Case
  - Clearly define intent of the Use Case
  - Define Main Success Scenario (Happy Path)
  - Define any alternate action paths
  - Use format of Stimulus: Response
  - Each specification must be testable
  - Write from actor’s perspective, in actor’s vocabulary

Identification of Use Cases [1]

- Method 1 - Actor based
  - Identify the actors related to the system
  - Identify the scenarios these actors initiate or participate in.
- Method 2 - Event based
  - Identify the external events that a system must respond to
  - Relate the events to actors and use cases
- Method 3 – Goal based
  - [Actors have goals.]
  - Find user goals. [Prepare actor-goal list.]
  - Define a use case for each goal.

Identification of Use Cases[2]

- To identify use cases, focus on *elementary business processes (EBP).*

- An EBP is a task performed by one person in one place at one time, in response to a business event. This task adds measurable business value and leaves data in a consistent state.

Use Cases and Goals

- Q1: “What do you do?”
- Q2: “What are your goals?”
- Which is a better question to ask a stakeholder?
- Answer: Q2. Answers to Q1 will describe current solutions and procedures; answers to Q2 support discussion of improved solutions

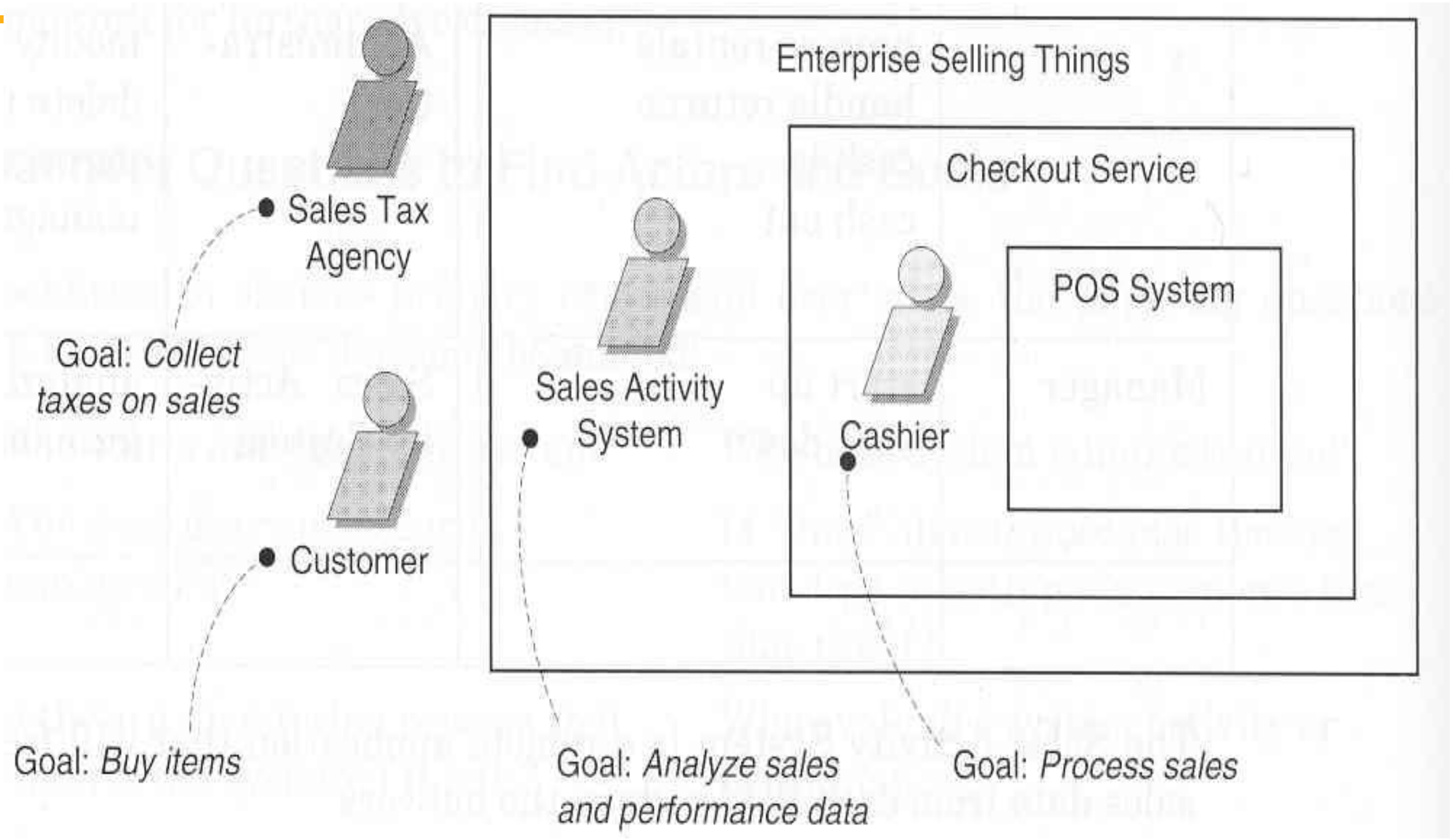
Finding Actors, Goals & Use Cases

- Choose the system boundary
  - Software, hardware, person, organization
- Identify the primary actors
  - Goals fulfilled by using the system
- Identify goals for each actor
  - Use highest level that satisfies EBP
  - Exception: CRUD (next slide)
- Define use cases that satisfy goals

Create, Retrieve, Update, Delete (CRUD)

- E.g., “edit user”, “delete user”, …
- Collapse into a single use case (e.g., *Manage Users*)
- Exception to EBP
  – Don’t take place at the same place/time
  – But since it’s so common, it would inflate the number of use cases to model each action as a separate use case

Actors, Goals & Boundaries



Back to POST - Actors

- Actors:
    - Cashier
    - Customer
    - Supervisor

- Choosing actors:
    - Identify system boundary
    - Identify entities, human or otherwise, that will interact with the system, from outside the boundary.
    - Example: A *temperature sensing device* is an actor for a temperature monitoring application.

POST - Use Cases: First Try

- Cashier
  - Log In
  - Cash out

- Customer
  - Buy items
  - Return items

Common mistake

- Representing individual steps as use cases.
  - Example: printing a receipt (Why is this being done ?)

High level vs. Low Level Use cases[1]

- Consider the following use cases:
  - Log out
  - Handle payment
  - Negotiate contract with a supplier

- These use cases are at different levels. Are they all valid? To check, use the EBP definition.

Use Cases can be defined at different levels of granularity

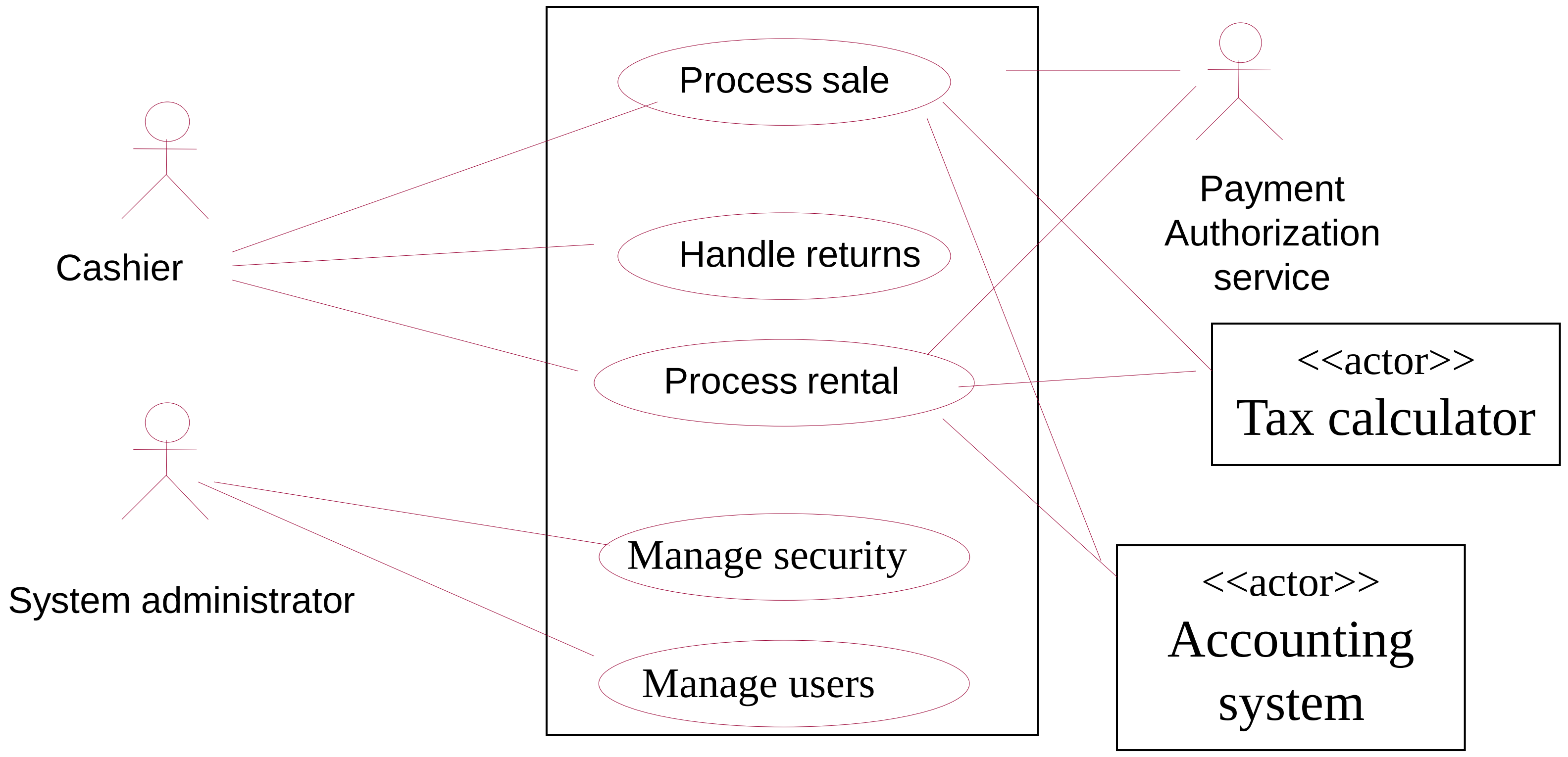
High level vs. Low Level Use cases [2]

- Log out: a secondary goal; it is necessary to do something but not useful in itself.

- Handle payment: A necessary EBP. Hence a primary goal.

- Negotiate contract: Most likely this is too high a level. It is composed of several EBPs and hence must be broken down further.

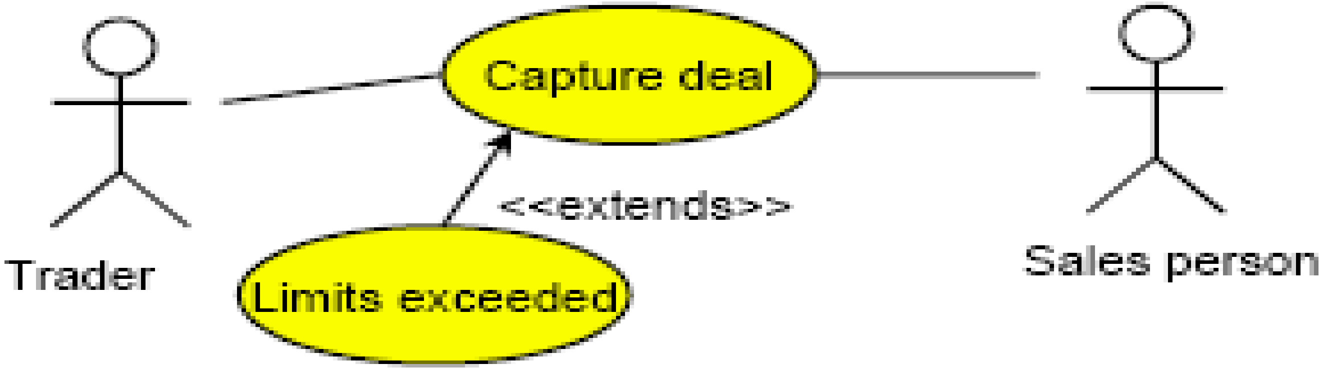
Use Case Diagram - Example



Use Case Diagram: illustrates a set of use cases for a system.

Extends

Use the **extends** relationship when you have a use case that is **similar** to another use case but does a bit more.



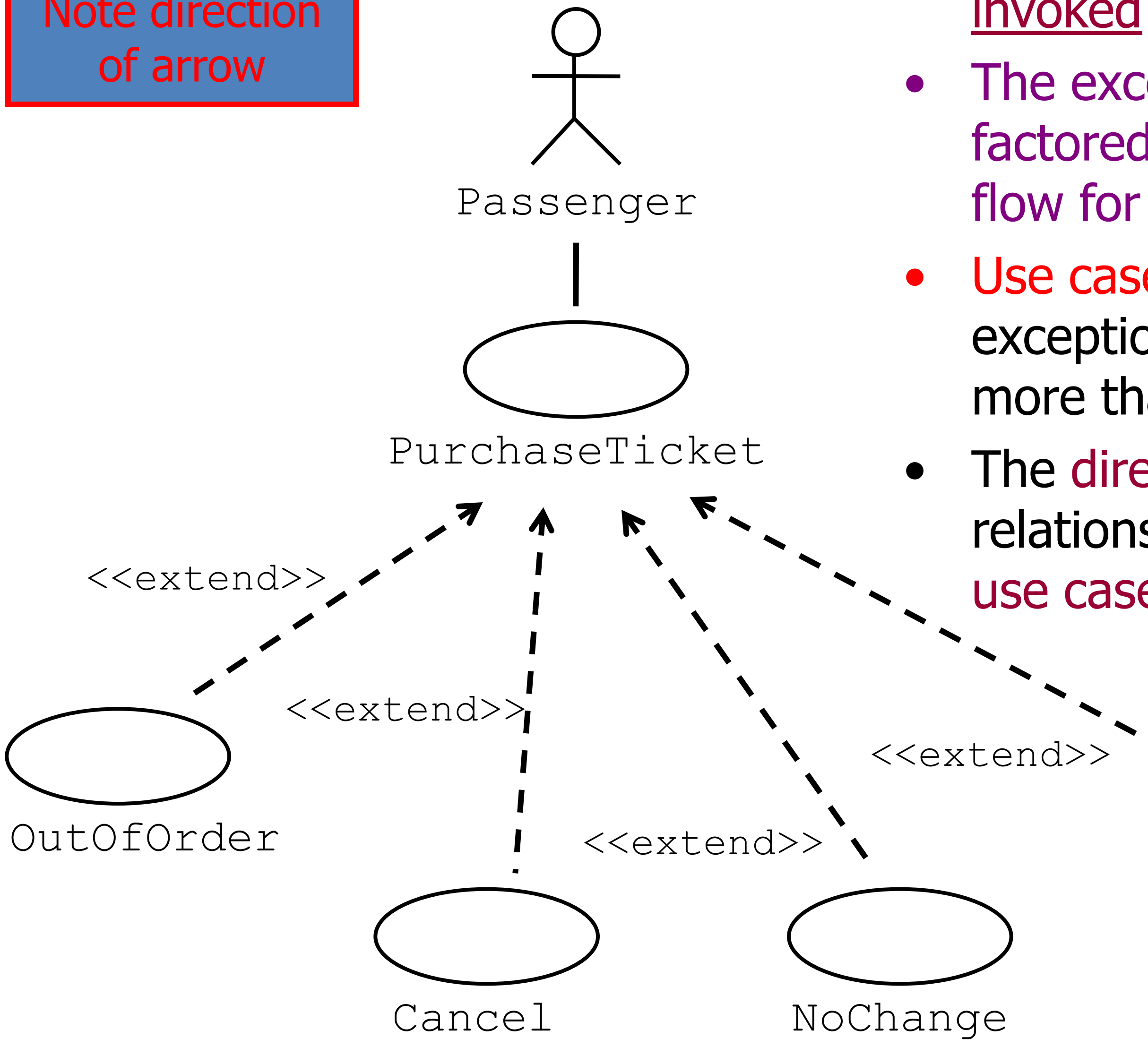
- In this example the basic use case is Capture deal.
- In case certain limits for a deal are exceeded the additional use case Limits exceeded is performed.

Extends

- Extensions are used instead of modeling every **extra** case by a single **use case**.
- Extensions are used instead of creating a **complex use case** that covers all variations.
- How do you address case variation?
  - Capture the simple, **normal** use case **first**.
  - For **every step** in that use case ask: “What could go wrong here?“
  - Plot all **variations** as extensions of the given use case.

The *<<extend>>* Relationship

Note direction of arrow

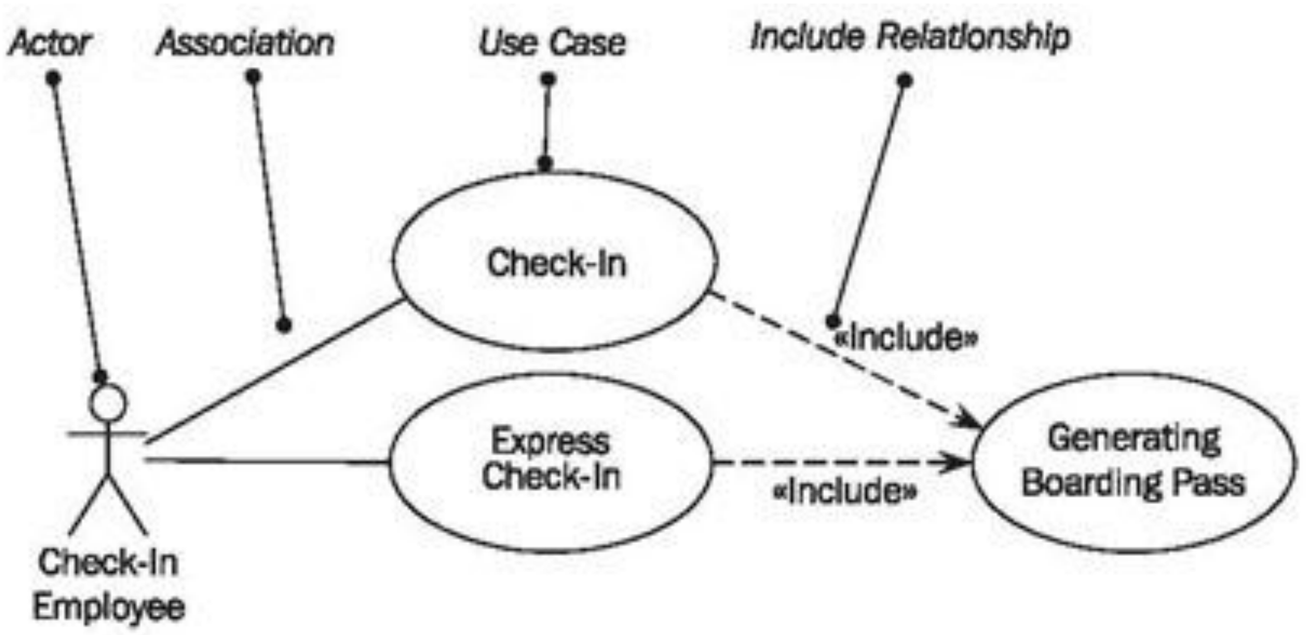


- *<<extend>>* relationships represent exceptional or seldom invoked cases.
- The exceptional event flows are factored out of the main event flow for clarity.
- Use cases representing exceptional flows can extend more than one use case.
- The direction of a *<<extend>>* relationship is to the extended use case

<<extend>> shown with dotted line

Includes

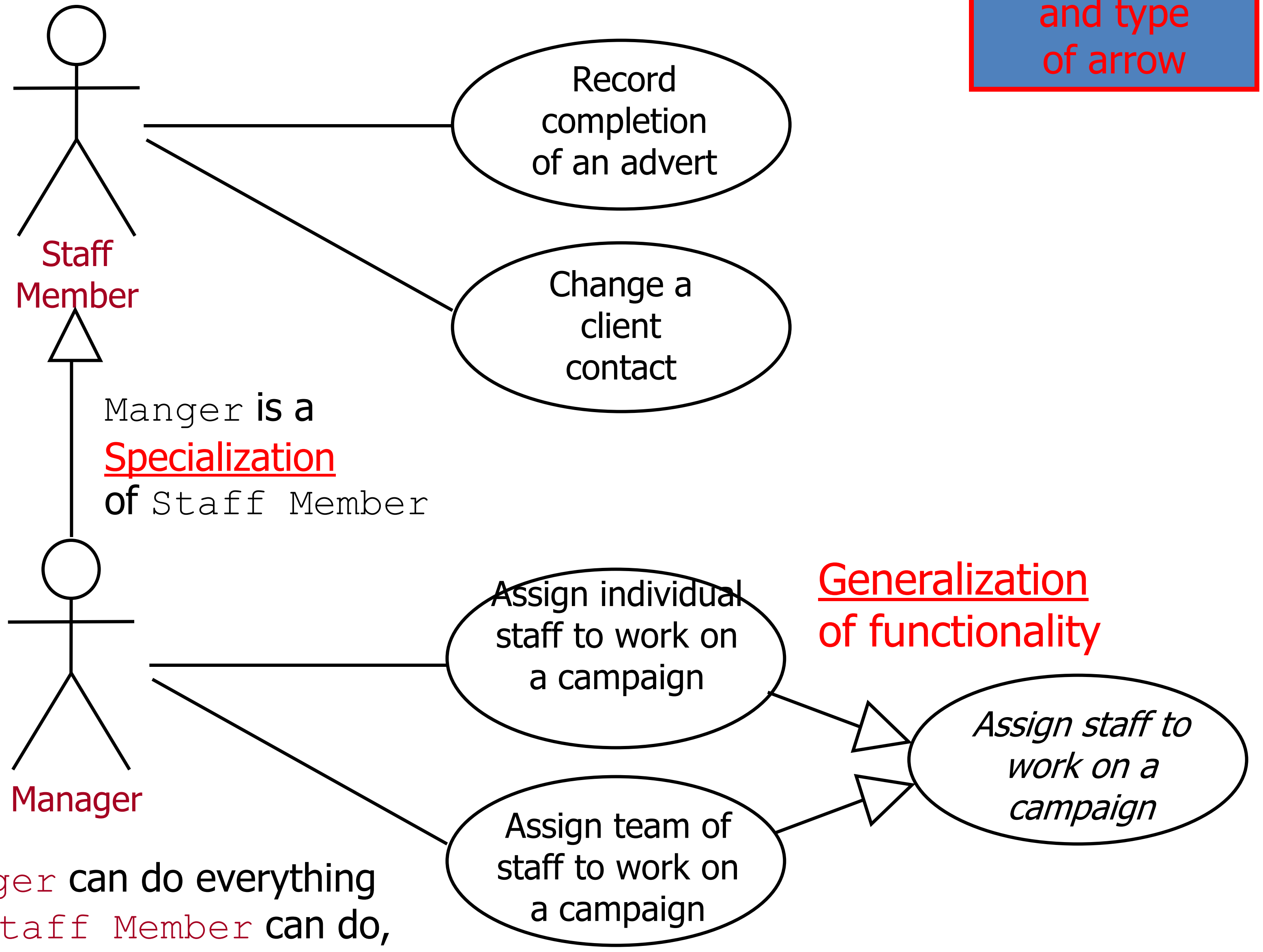
The **Includes** relationship occurs when you have a chunk of behavior that is **similar across** several **use cases**.



Includes and Extends

- The similarities between **extends** and **Includes** are that they both imply **factoring out** common **behavior** from several use cases to a single use case that is
  - used by several other use cases or
  - extended by other use cases.
- Apply the following rules:
  - Use **extends**, when you are describing a **variation** on normal behavior.
  - Use **Includes** when you want to split off **repeating** details in a use case.

Note direction and type of arrow

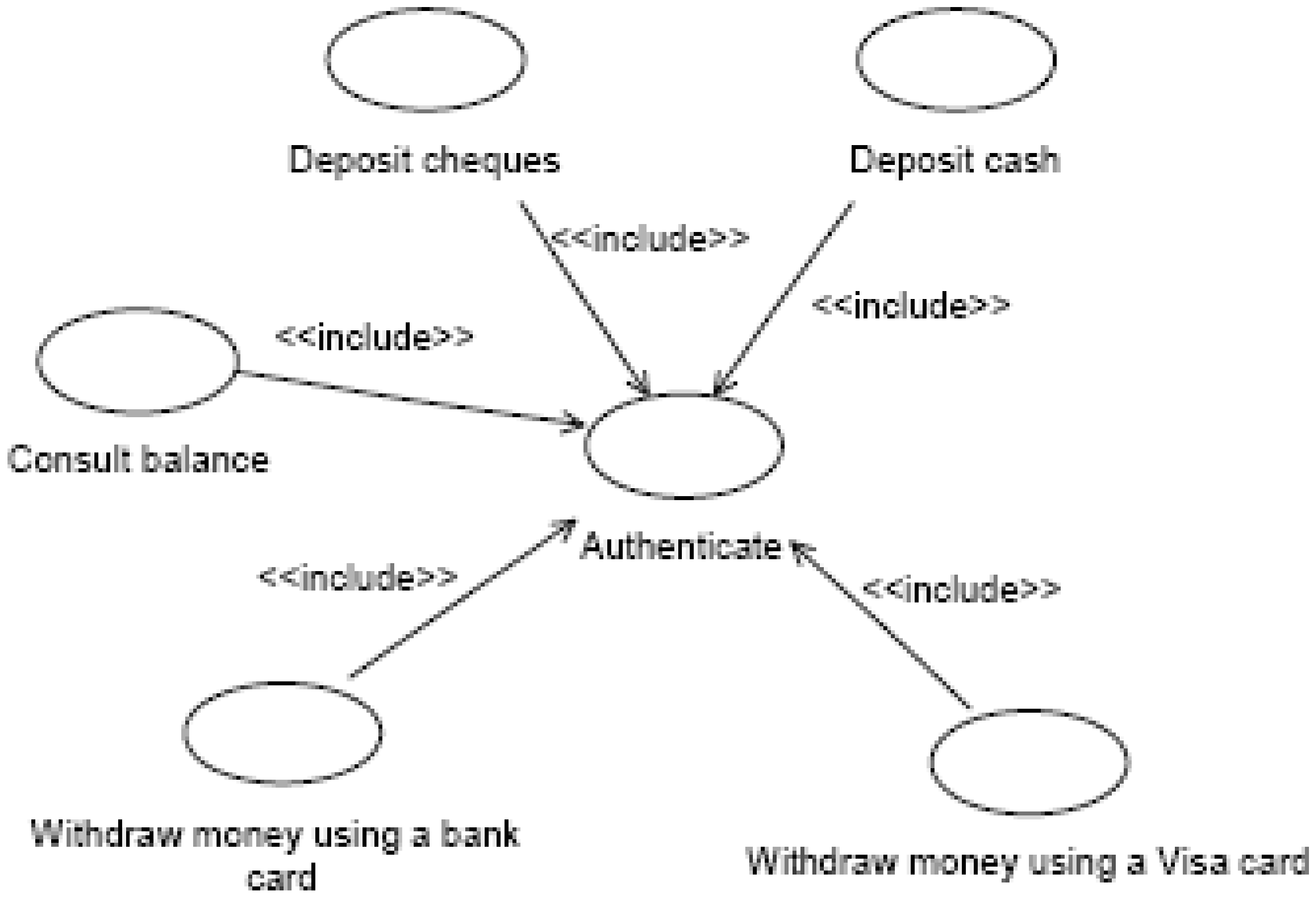

Staff Member
Record completion of an advert
Change a client contact
Manager
Assign individual staff to work on a campaign
Assign team of staff to work on a campaign
Assign staff to work on a campaign

`Manger` is a Specialization of `Staff Member`

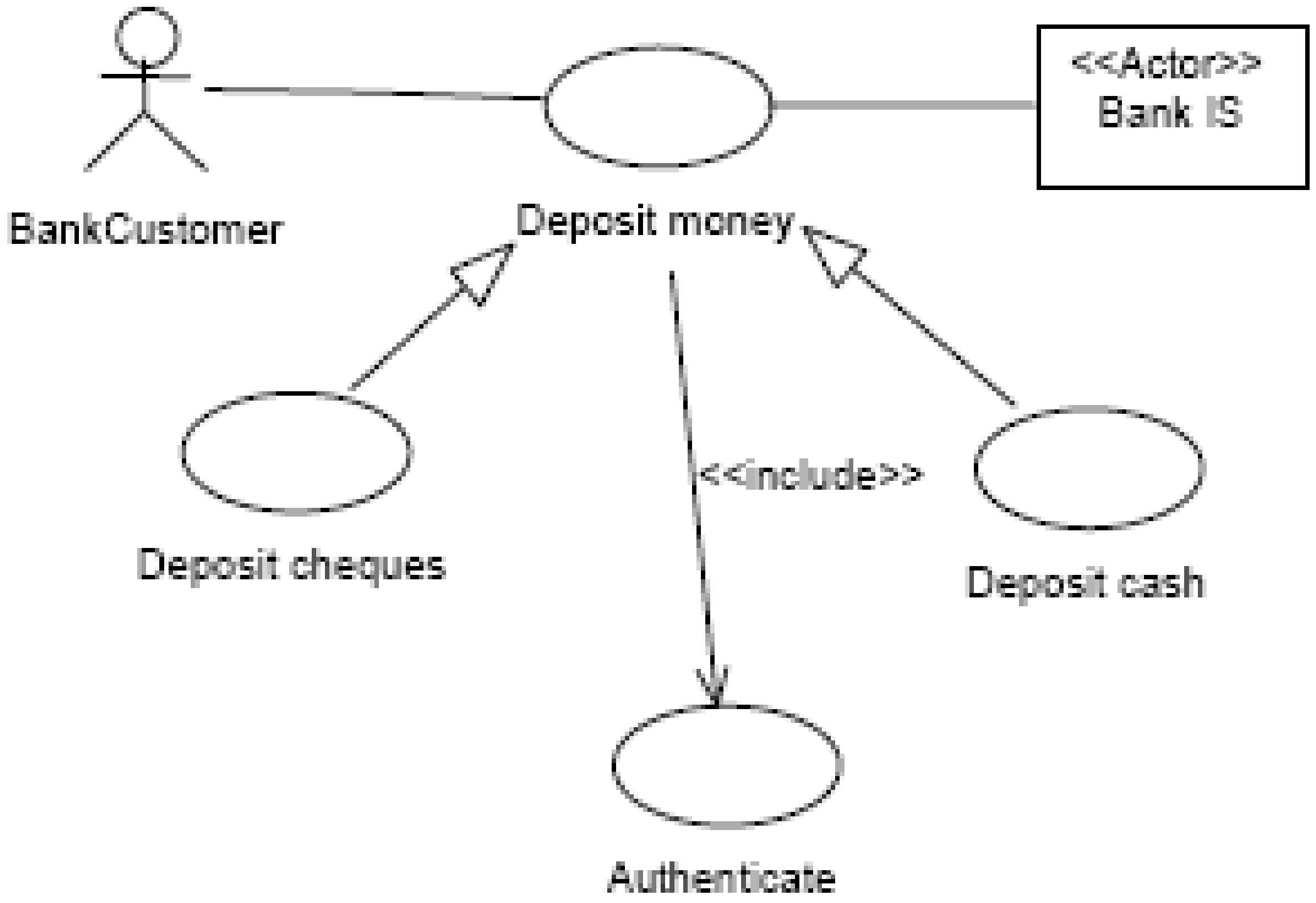
Generalization of functionality

`Manager` can do everything that `Staff Member` can do, and more.

Example of an include relationship



Example of a generalisation relationship



Developing a Use Case

- Ask yourself these questions:
  - What are the main tasks or functions that are performed by the actor?
  - What system information will the actor acquire, produce or change?
  - Will the actor have to inform the system about changes in the external environment?
  - What information does the actor desire from the system?
  - What changes or events will the actor of the system need to be informed about?

Use Case Description

- It includes
  - How the use case starts and ends
  - The context of the use case
  - The actors and system behavior described as intentions and responsibilities
  - All the circumstances in which the primary actor's goal is reached and not reached
  - What information is exchanged

More on Use Cases

- Narrate use cases independent of implementation
- State success scenarios (how do you determine the success of a use case).
- A use case corresponds to one or more scenarios.
- Agree on a format for use case description
- Name a use case starting with a verb in order to emphasize that it is a process (Buy Items, Enter an order, Reduce inventory)

Exception handling

- Document exception handling or branching
  - What is expected of the system when a "Buy Item" fails ?
  - What is expected of the system when a "credit card" approval fails ?

A sample Use Case

Use case:	Buy Items
Actors:	Customer, Cashier
Description:	A customer arrives at a checkout with items to purchase. The cashier records the purchase items and collects payment.

Ranking the Use cases

- The **developers** should consider the **architectural** risk.
  - Do not omit use cases which later cause you to do a lot of rework to fit them in.
  - Concentrate to the use cases which are technologically most challenging.
- The **developers** should be aware of the **schedule** risks.
  - “I’m pretty sure I know how long it will take.”
  - “I can estimate the time only to the nearest man-month.”
  - “I have no idea.”

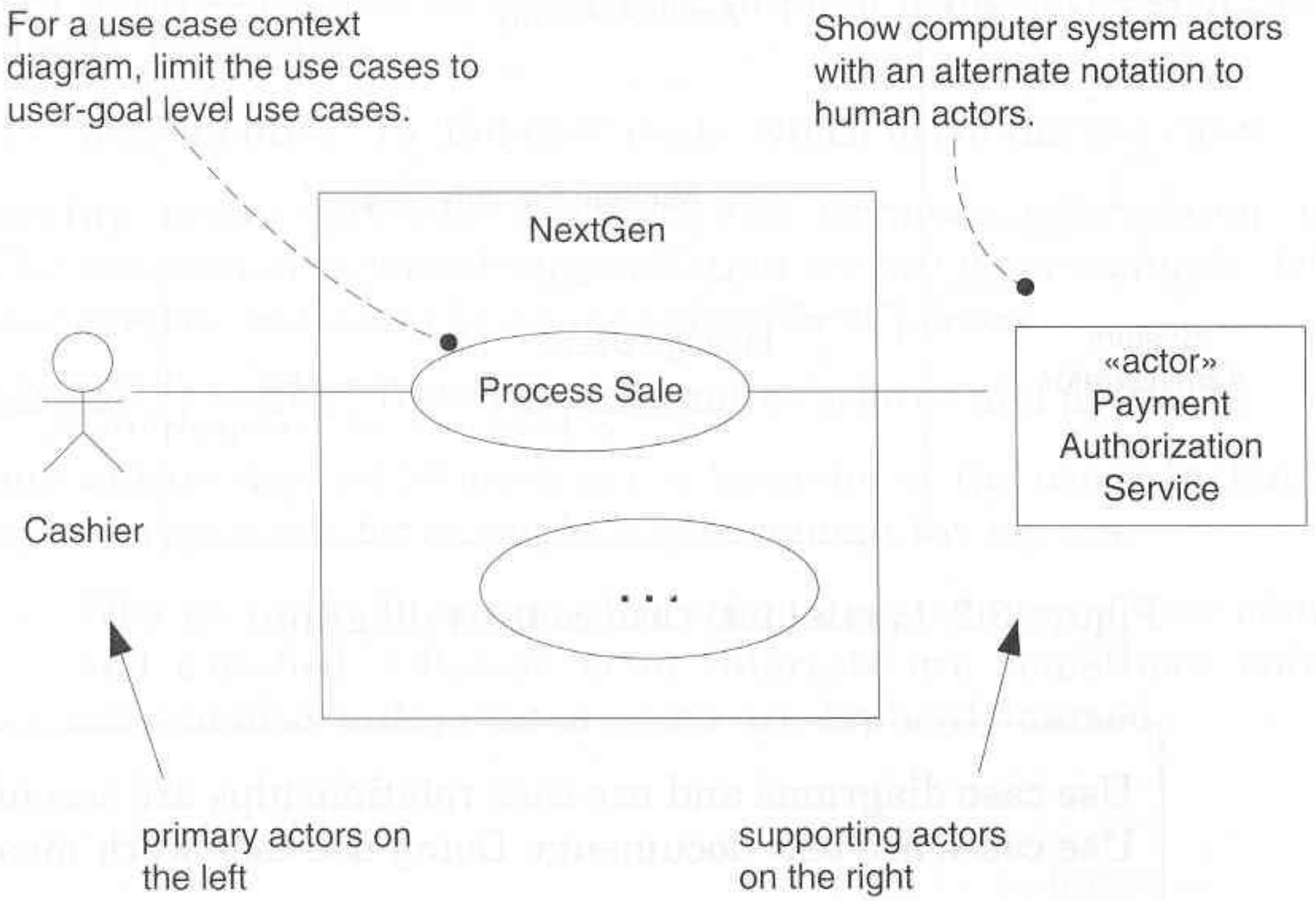
Ranking Use Cases

- Use some ordering that is customary to your environment
  - Example: High, Medium, Low
  - Example: Must have, Essential, Nice to have

- Useful when deciding what goes into an increment

- POST example:
  - Buy Items - High
  - Refund Items - Medium (Why?)
  - Shut Down POST terminal - Low

Notation Guidelines


For a use case context diagram, limit the use cases to user-goal level use cases.
Show computer system actors with an alternate notation to human actors.
NextGen
Process Sale
. . .
Cashier
«actor» Payment Authorization Service
primary actors on the left
supporting actors on the right

More on Use Cases

- Use cases drive the whole development process.
- Use cases are not object-oriented.
- Use cases represent an external view of the system.
- Use cases should be written and named at the domain level in requirements.
- Adjust the **granularity** of your use cases according to the complexity of the individual problem you are working on, bearing in mind that
  - **too many** use cases can be overwhelming while
  - **too few** use cases may hide important details.

More on Use cases

- Use cases force one to look at exceptional as well as normal behavior.
  - helps us to surface hidden requirements
- Use cases can help formulate system tests.
  - "Is this use case built into the system?"
- Use case templates facilitate interviewing and reviews.
- Ease an iterative development lifecycle
  - levels of precision for a use case by refinement
- Support an incremental development lifecycle
  - E.g. "Foobar" Release 1: use cases 1-20;
    "Foobar" Release 2: use cases 1-29.

More on Use Cases

- Use Simple Grammar (active voice)
  - GOOD “1. Actor requests System for rum-flavored popcorn.”
  - BAD “1. The System was requested by the Actor for some of the rum-flavored variety of popcorn.”
- Avoid "if" statements. Factor out into “alternative flows” instead.
- Look for ways that each step in the basic flow can fail.
  - a single step may have several alternatives

Use Case Tips

- Keep them Simple.
- Don’t worry about capturing all the use cases.
- Don’t worry about finding all the details
- They are going to change tomorrow.
- They are *Just in Time Requirement.*

Use Case Example

- Use case name: Login
- Goal: Authenticate user
- Precondition: System is ready
- Postcondition: User is logged in
- Actors: User
- Triggering event: User requests login
- Description: User supplies username and password. If the username is known and the password is valid, the user is logged in.
- Alternatives: 1) *User name is not valid*: Retry getting user name up to three times, then report failure, record error, and wait one minute before allowing next login.

  2) *Password is not valid*: Retry getting password up to three times, then report failure, records error and wait one minute before allowing next login.

Use Case Example

- **Use Case**: Deposit Money
- **Scope**: Bank Accounts and Transactions System
- **Level**: User Goal
- **Intention in Context**: The intention of the Client is to deposit money on an account. Clients do not interact with the System directly; instead, for this use case, a Client interacts via a Teller. Many Clients may be performing transactions and queries at any one time.
- **Primary Actor**: Client
- **Basic Flow**:

1. Client requests Teller to deposit money on an account, providing sum of money.