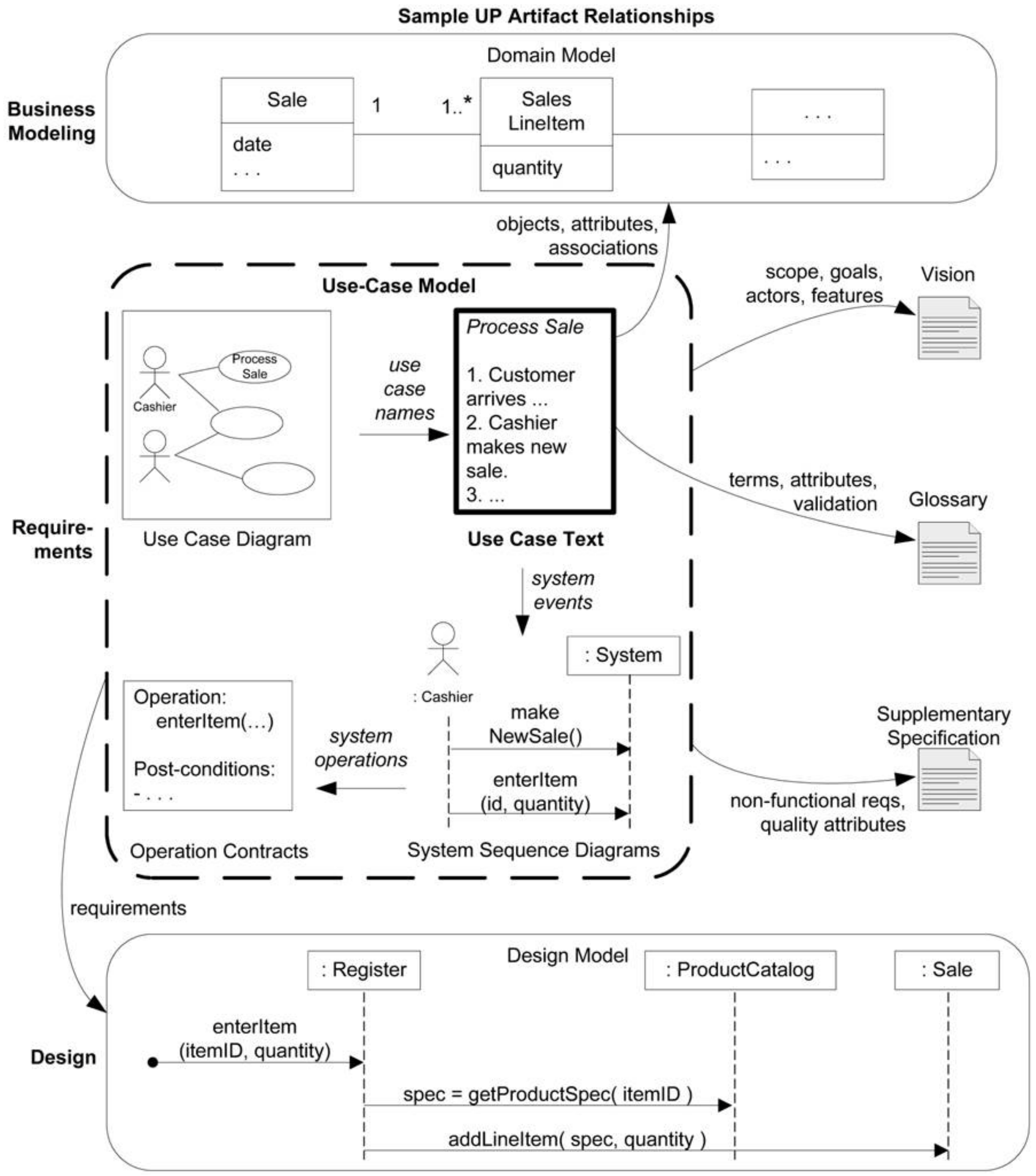
Chapter 6. Use Cases

UP Artifacts Relationships



Use Cases

- Informally, **use cases** are **text stories** of some actor using a system to meet goals.
  - A mechanism to capture requirements
  - For example: **Process Sale** (brief format)
    - **Process Sale**: A customer arrives at a checkout with items to purchase. The cashier uses the POS system to record each purchased item. The system presents a running total and line-item details. The customer enters payment information, which the system validates and records. The system updates inventory. The customer receives a receipt from the system and then leaves with the items.

Definition : Actors, Scenarios and Use Cases

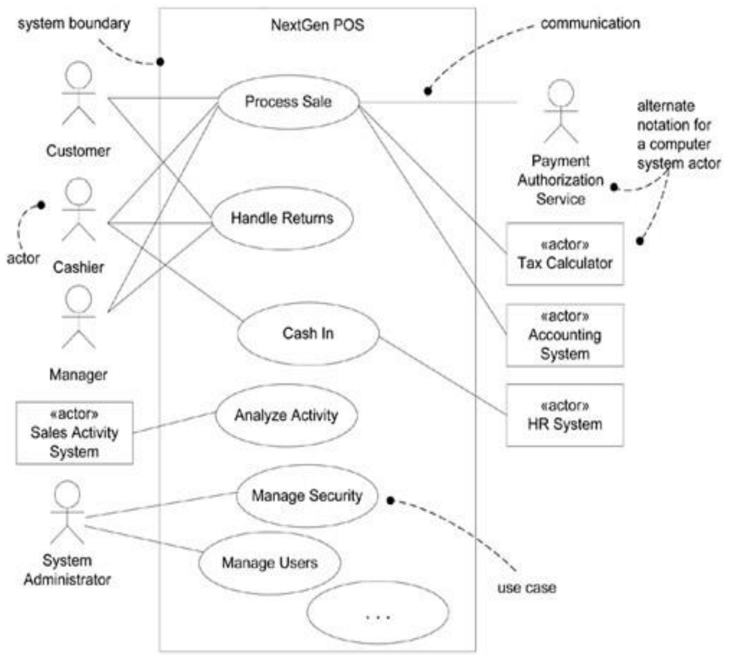
- **Actor** : An actor specifies a role played by a user or any other system that interacts with the system to develop.
- **Scenario** : a specific sequence of actions and interactions between actors and the system
  - Also called a use case instance

- **Use case** : a collection of related success and failure scenarios, which describe an actor using the system to achieve a goal
  - For example: **Handle Returns**
    - *Main Success Scenario*: A customer arrives at a checkout with items to return. The cashier uses the POS system to record each returned item …
    - *Alternate Scenarios*: If the customer paid by credit, and the reimbursement transaction to their credit account is rejected, inform the customer and pay them with cash…

“***A set of use-case instances, where each instance is a sequence of actions a system performs that yields an observable result of value to a particular actor***.”

Use Cases and the Use-Case Model

- UP defines the **Use-Case Model** within the requirements discipline.
  - A set of all written use cases
  - A model of the system's functionality and environment
- The use case model may optionally include **use case diagrams**:
  - Names of use cases and actors
  - A context diagram of system and its environment



Motivation: Why Use Cases?

- Use cases
  - Are a simple way of capturing user goals
    - help keep it simple, and make it possible for domain experts or requirement donors to themselves write (or participate in writing)
  - Emphasize the user goals and perspective
    - Instead of asking for a list of system features

- **Use Cases are requirements**, primarily **functional (behavioral)** requirements.
  - "**F**" (functional or behavioral) in terms of **FURPS+** requirements types
  - Reduce the importance or use of detailed old-style feature lists

Identifier	Requirement
REQ1	**The system shall keep the door locked at all times, unless commanded otherwise by authorized user. When the lock is disarmed, a countdown shall be initiated at the end of which the lock shall be automatically armed (if still disarmed).**
REQ2	**The system shall lock the door when commanded by pressing a dedicated button.**

Three Kinds of Actors

- **Primary Actor** : has user goals fulfilled through using services of the SuD (System under Discussion)
  - e.g., cashier
  - To find user goals, which drive use cases.
  - Typically, but not always, primary actor initiates the interaction
- **Supporting Actor** : provides a service to the SuD
  - e.g., payment authorization service
  - Clarify external interfaces and protocols.
- **Offstage Actor** : has an interest in the behavior of the use case, but is not primary or supporting
  - e.g., government tax agency
  - To ensure that all necessary interests are identified and satisfied.
  - Offstage actor interests are sometimes subtle or easy to miss unless these actors are explicitly named.

Three Common Use Case Formats

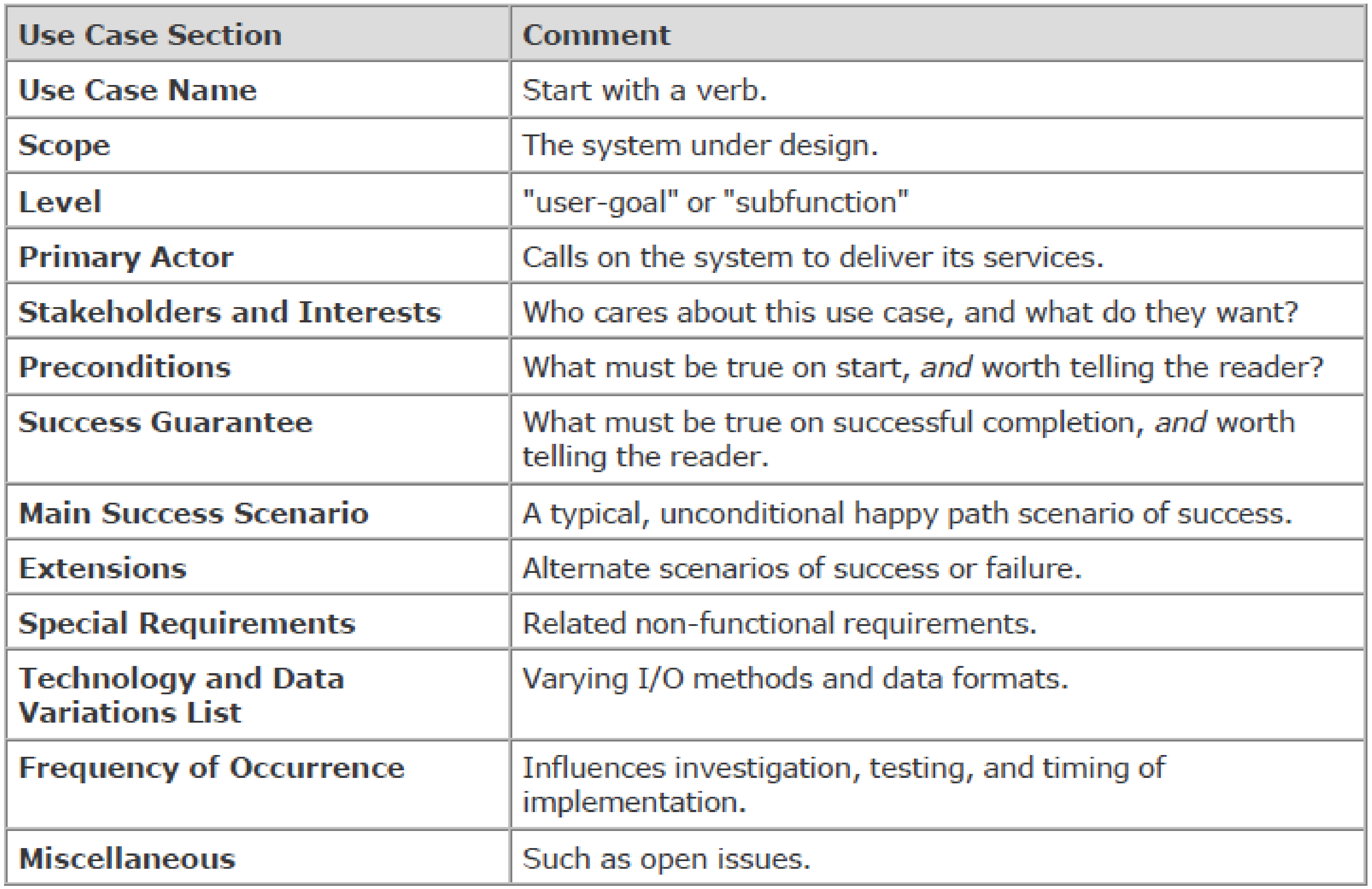
- **Brief**: Terse one paragraph summary, usually the main success scenario or happy path

- **Casual**: Informal paragraph format. Multiple paragraphs that cover various scenarios.

Handle Returns (Casual Style)

- Main Success Scenario:
  - A customer arrives at a checkout with items to return. The cashier uses the POS system to record each returned item …
- Alternate Scenarios:
  - If the customer paid by credit, and the reimbursement transaction to their credit account is rejected, inform the customer and pay them with cash.
  - If the item identifier is not found in the system, notify the Cashier and suggest manual entry of the identifier code (perhaps it is corrupted).
  - If the system detects failure to communicate with the external accounting system, …

- **Fully Dressed** : Includes all steps, variations and supporting sections (e.g., preconditions).



Use Case Section	**Comment**
Use Case Name	Start with a verb.
Scope	The system under design.
Level	"user-goal" or "subfunction"
Primary Actor	Calls on the system to deliver its services.
Stakeholders and Interests	Who cares about this use case, and what do they want?
Preconditions	What must be true on start, *and* worth telling the reader?
Success Guarantee	What must be true on successful completion, *and* worth telling the reader.
Main Success Scenario	A typical, unconditional happy path scenario of success.
Extensions	Alternate scenarios of success or failure.
Special Requirements	Related non-functional requirements.
Technology and Data Variations List	Varying I/O methods and data formats.
Frequency of Occurrence	Influences investigation, testing, and timing of implementation.
Miscellaneous	Such as open issues.

Example: Fully Dressed Style

Use Case UC1: Process Sale

- Scope: NextGen POS application
- Level: user goal
- Primary Actor: Cashier

- Stakeholders and Interests:
  - Cashier: Wants accurate, fast entry, and no payment errors, as cash drawer shortages are deducted from his/her salary.
  - Salesperson: Wants sales commissions updated.
  - Customer: …
  - Company: …
  - Manager: …
  - Government Tax Agencies: …
  - Payment Authorization Service: …

- Preconditions: Cashier is identified and authenticated.
- Success Guarantee (or Postconditions): Sale is saved. Tax is correctly calculated. Accounting and Inventory are updated. Commissions recorded. Receipt is generated. Payment authorization approvals are recorded.

Example: Fully Dressed Style

<u>Main Success Scenario (or Basic Flow):</u>

1. Customer arrives at POS checkout with goods and/or services to purchase.
2. Cashier starts a new sale.
3. Cashier enters item identifier.
4. System records sale line item and presents item description, price, and running total. Price calculated from a set of price rules.

Cashier repeats steps 3-4 until indicates done.

5. System presents total with taxes calculated.
6. Cashier tells Customer the total, and asks for payment.
7. Customer pays and System handles payment.
8. System logs completed sale and sends sale and payment information to the external Accounting system (for accounting and commissions) and Inventory system (to update inventory).
9. System presents receipt.
10. Customer leaves with receipt and goods (if any).

Example: Fully Dressed Style

Extensions (or Alternative Flows):

*a. At any time, Manager requests an override operation:

1. System enters Manager-authorized mode.
2. Manager or Cashier performs one Manager-mode operation. e.g., cash balance change, resume a suspended sale on another register, void a sale, etc.
3. System reverts to Cashier-authorized mode.

*b. At any time, System fails:

To support recovery and correct accounting, ensure all transaction sensitive state and events can be recovered from any step of the scenario.

1. Cashier restarts System, logs in, and requests recovery of prior state.
2. System reconstructs prior state.
   - 2a. System detects anomalies preventing recovery:
     1. System signals error to the Cashier, records the error, and enters a clean state.
     2. Cashier starts a new sale.

Example: Fully Dressed Style

1a. Customer or Manager indicate to resume a suspended sale.
   1. Cashier performs resume operation, and enters the ID to retrieve the sale.
   2. System displays the state of the resumed sale, with subtotal.
      2a. Sale not found.
         1. System signals error to the Cashier.
         2. Cashier probably starts new sale and re-enters all items.
   3. Cashier continues with sale (probably entering more items or handling payment).

2-4a. Customer tells Cashier they have a tax-exempt status (e.g., seniors, native peoples)
   1. Cashier verifies, and then enters tax-exempt status code.
   2. System records status (which it will use during tax calculations)

….

For Reference

Main Success Scenario (or Basic Flow):
1. Customer arrives at POS checkout with goods and/or services to purchase.
2. Cashier starts a new sale.
3. Cashier enters item identifier.
4. System records sale line item and presents item description, price, and running total. Price calculated from a set of price rules.

What Do the Sections Mean?

- **Scope**: bounds the system (or systems) under design.
  - System use case: describes the use of one software system
  - Business use case: enterprise-level process description

- **Level**: user-goal level or the subfunction level
  - **User-Goal level**: a common kind that describes the scenarios to fulfill the goals of a primary actor (corresponds to an Elementary Business Process(EBP))
  - **Subfunction level**: substeps required to support a user goal. Factors out duplicate substeps in many use cases (e.g., Pay by Credit).

- **Primary Actor**:
  - The principal actor that calls upon system services to fulfill a goal

What Do the Sections Mean?

- **Stakeholders and Interests list**: suggests and bounds what the system must do
  - Very important since identifying all stakeholders and their interests to answers the questions ‘What should be in the use case?’ : ‘That which satisfies all stakeholders’ interests’.

- **Preconditions and Success Guarantees (Postconditions)**:
  - Used only when you are stating something non-obvious and noteworthy
  - **Preconditions**: state what must always be true before a scenario is begun
    - Imply completion of another scenario (e.g., login)
    - Don’t mention them if obvious (e.g., the system has power)
  - **Success guarantees (Postcondition)**: state what must be true on successful completion of the use case.
    - Should meet the needs of all stakeholders

What Do the Sections Mean?

- **Main Success Scenario (Basic Flow)** :
  - Also called the 'happy path' scenario
  - Describes the typical success path that satisfies the interests of the stakeholders
  - Often does not include any conditions of branching
    - More comprehensible and extensible to defer all conditional handling to the Extensions section
  - The scenario records the steps, of which there are three kinds:
    1. An interaction between actors
    2. A validation (usually by the system)
    3. A state change by the system (for example, recording or modifying something)
  - While the first step of a use case does not always fall into this classification.
    - But, it indicates the trigger event that starts the scenario.

What Do the Sections Mean?

- **Extensions (or Alternate Flows)** :
  - Branches from the main success scenario with respect to its steps
    - Comprise the majority of the text, indicating all other scenarios or branches, both success and failures (besides the main success scenario).
  - Main success scenario + extensions: Should satisfy "nearly" all stakeholders' interests.
    - Non-functional requirements are expressed in the Supplementary Specification.
  - At the end of the extension handling, by default the scenario merges back with the main success scenario, unless the extension indicates otherwise.
  - Complicated extension points are usually expressed as separate use cases.

Guideline: Write in an Essential UI-Free Style

- Essential writing style is expressing user intentions and system responsibilities, rather than concrete actions.
  - Concrete use cases may be useful during GUI design in a later phase, but are better avoided during early requirements analysis.
  - For example: *Manage Users* use case

Essential Style

1. Administrator identities self.
2. System authenticates identity.
3. …

Concrete Style

1. Administrator enters ID and PW in dialog box.
2. System authenticates Administrator.
3. System displays the "edit user" window.
4. …

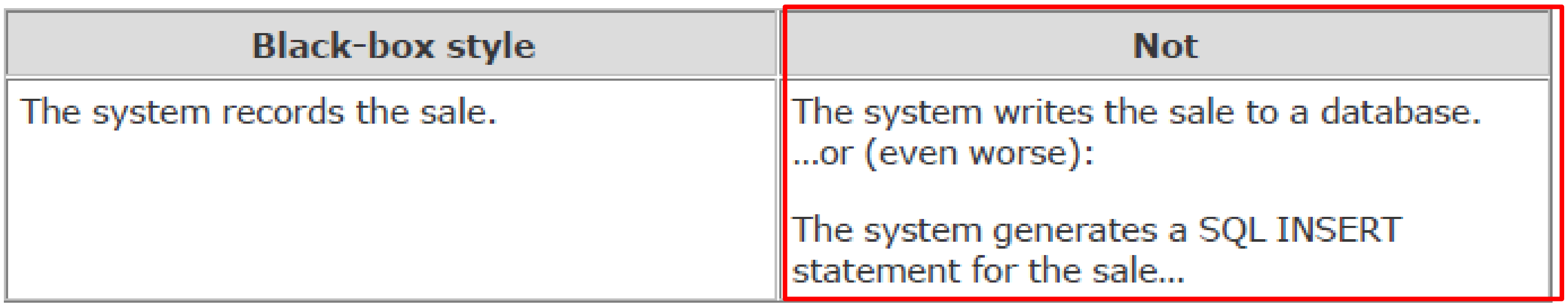
Guideline: Write Terse Use Cases

- Keep your use cases short and to the point.
- Avoid noise words.

Guideline: Write Black-Box Use Cases

- Don’t describe the internal working of the system, its components or design.

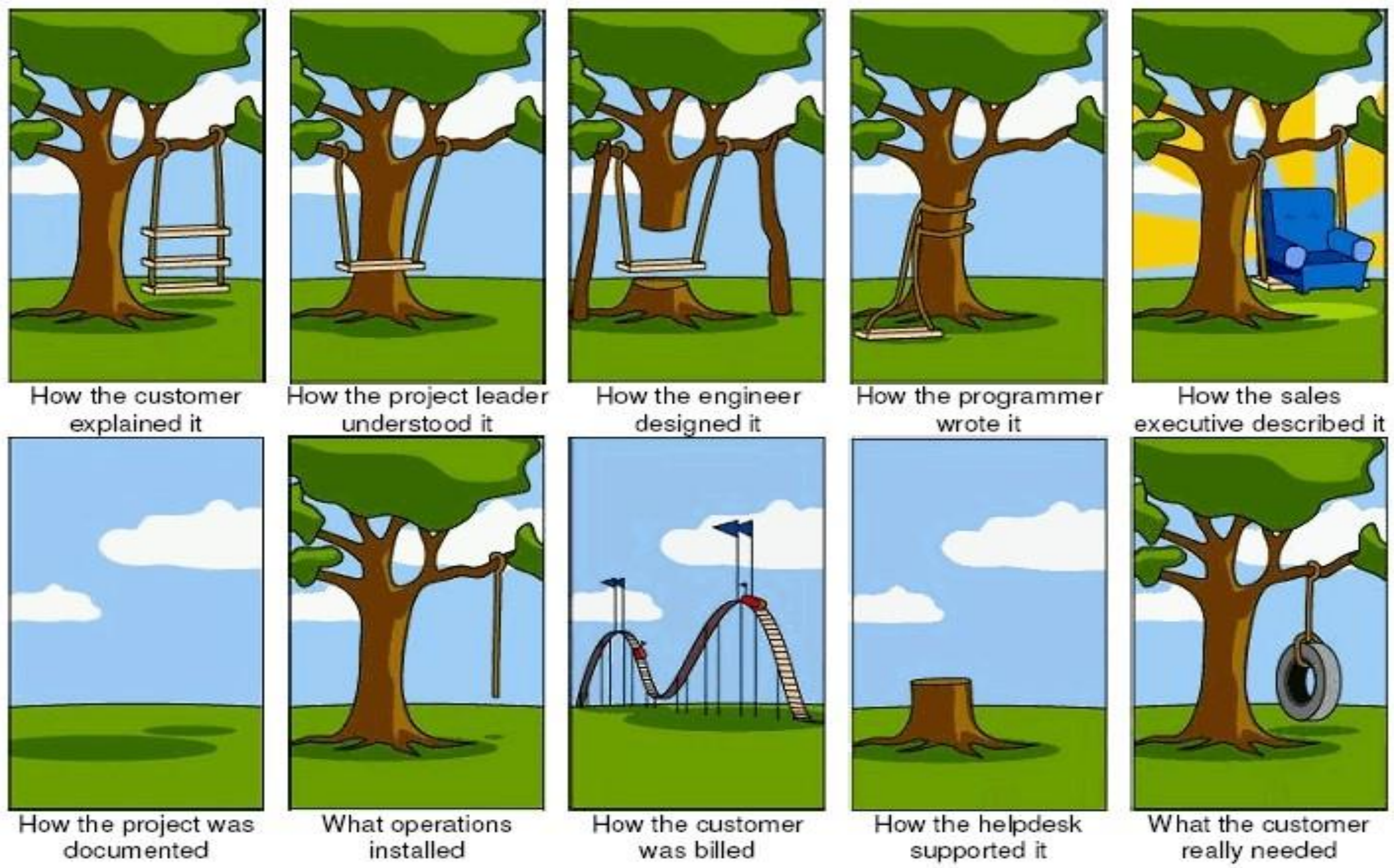
- Concentrate on responsibilities.
  - Define what the system does (analysis), rather than how it does it (design).



Black-box style	Not
The system records the sale.	The system writes the sale to a database. ...or (even worse): The system generates a SQL INSERT statement for the sale...

Guideline: Take an Actor and Actor-Goal Perspective

- Write requirements focusing on the users or actors of a system, asking what about their goals and typical situations.
- Focus on understanding what the actors considers a valuable result.


How the customer explained it
How the project leader understood it
How the engineer designed it
How the programmer wrote it
How the sales executive described it
How the project was documented
What operations installed
How the customer was billed
How the helpdesk supported it
What the customer really needed

Guideline: How to Find Use Cases

- Use cases are defined to satisfy the goals of the primary actors.

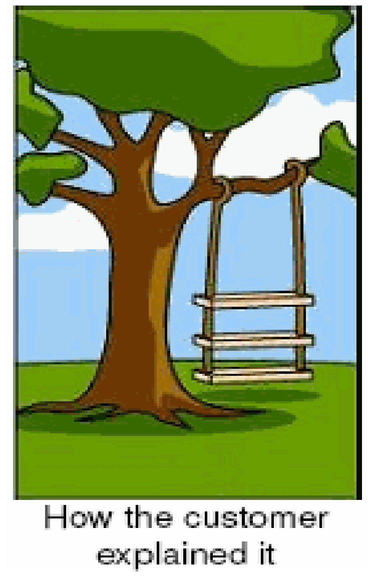
- The basic procedure is:
  1. Choose the system boundary
  2. Identify the primary actors
  3. Identify the goals for each primary actor
  4. Define use cases that satisfy user goals

- Why ask about actor goals rather than use cases?
  - Asking about goals helps discovering real user requirements instead of current practices and the complications that come with them.
  - **The difference between two questions in a requirements workshop:**

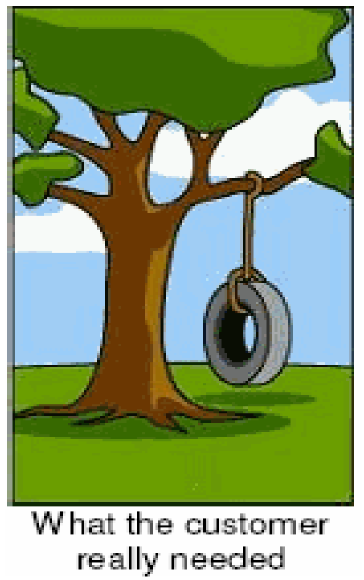
    "What do you do?"

    vs.

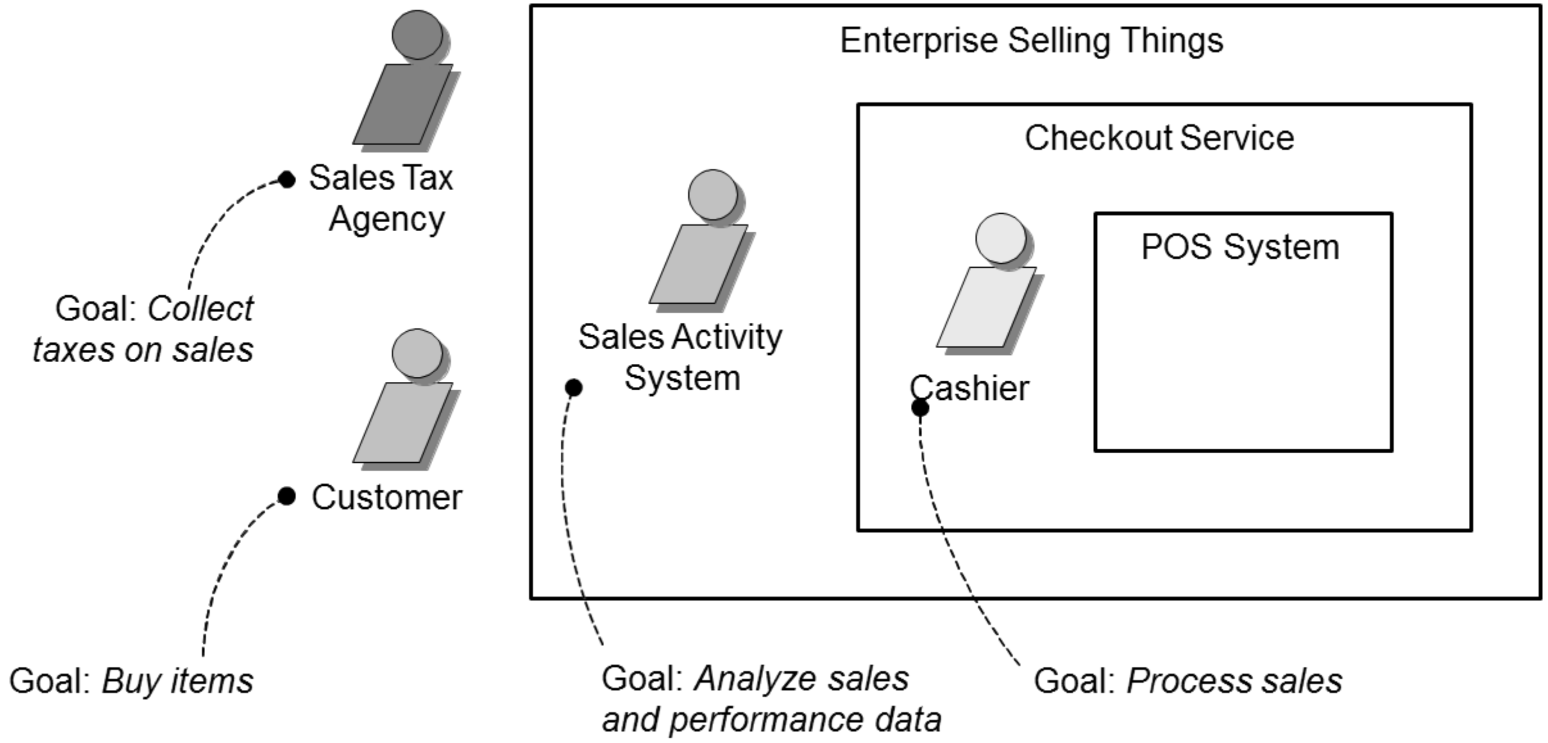
    "What are your goals whose results have measurable value?"



vs.



- Who is the primary actor?
  - It depends on the context.



Applying UML: Use Case Diagrams

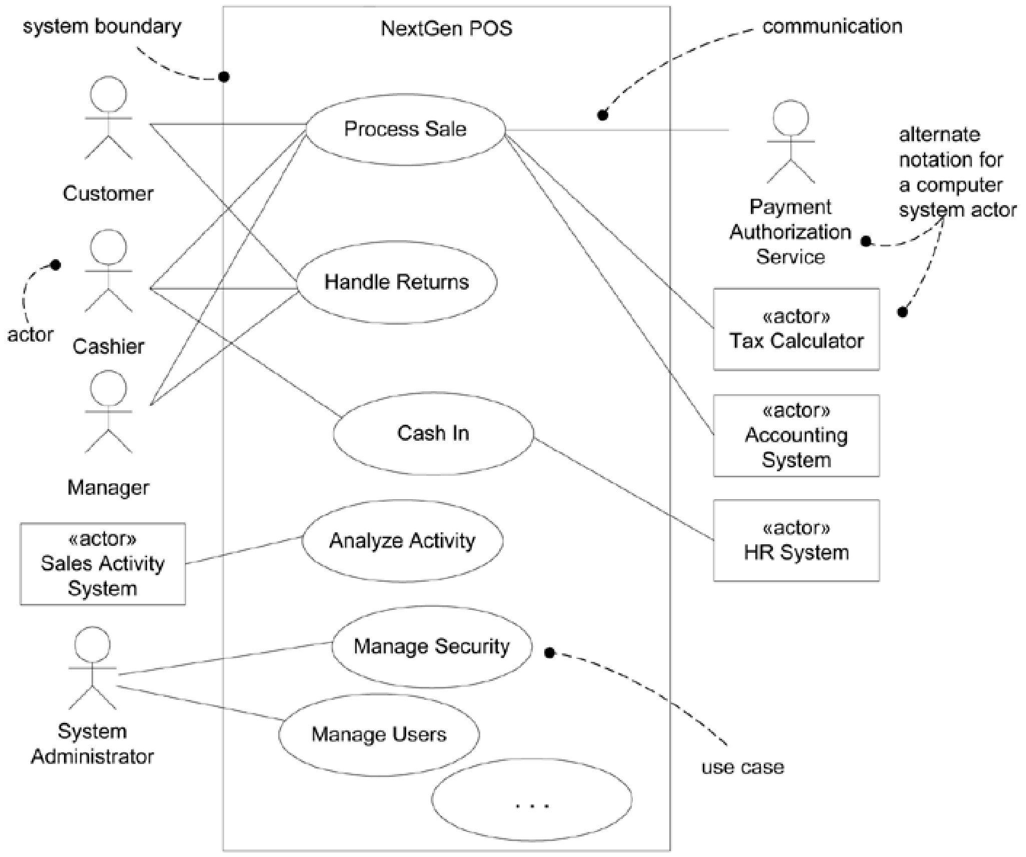
- **Use case diagram** illustrates the name of use cases and actors, and the relationships between them.
  - Use case diagrams and use case relationships are secondary in use case work.
  - Use cases are text documents.

- **A simple use case diagram** provides a succinct visual context diagram for the system, illustrating the external actors and how they use the system.
  - A common sign of a novice (or academic) use case modeler is a preoccupation with use case diagrams and use case relationships, rather than writing text.
- Keep use case diagrams short and simple.

Draw a simple use case diagram in conjunction with an actor-goal list.

Use Case Diagrams as System Context

- A use case diagram is an excellent picture of the **system context**.
  - Shows boundary of a system, what lies outside and how it gets used.
  - Summarizes the behavior of a system and its actors.



Discipline	Artifact	Incep (1 week)	Elab1 (3 weeks)	Elab2 (3 weeks)	Elab 3 (3weeks)	Elab 4 (3 weeks)
Requirements	Use-Case Model	2-day requirements workshop. Most use cases are identified by name, and summarized in a short paragraph. Pick 10% from the high-level list to analyze and write in detail. This 10% will be the most architecturally important, risky, and high-business value.	Near the end of this iteration, host a 2-day requirements workshop. Obtain insight and feedback from the implementation work, then complete 30% of the use cases in detail.	Near the end of this iteration, host a 2-day requirements workshop. Obtain insight and feedback from the implementation work, then complete 50% of the use cases in detail.	Repeat, complete 70% of all use cases in detail.	Repeat with the goal of 80~90% of the use cases clarified and written in detail. Only a small portion of these have been built in elaboration; the remainder are done in construction.
Design	Design Model	none	Design for a small set of high-risk architecturally significant requirements.	repeat	repeat	Repeat. The high risk and architecturally significant aspects should now be stabilized.
Implementation	Implementation Model (code, etc.)	none	Implement these.	Repeat. 5% of the final system is built.	Repeat. 10% of the final system is built.	Repeat. 15% of the final system is built.
Project Management	SW Development Plan	Very vague estimate of total effort.	Estimate starts to take shape.	a little better...	a little better...	Overall project duration, major milestones, effort, and cost estimates can now be rationally committed to.

When Various UP Artifacts are Created?

Discipline	Artifact	Inception	Elaboration	Construction	Transition
Business Modeling	Domain Model		s		
Requirements	Use-Case Model	s	r		
	Vision	s	r		
	Supplementary Specification	s	r		
	Glossary	s	r		
Design	Design Model		s	r	
	SW Architecture Document		s		

Summary

- Motivations of Use Case
- Actors
- Use Case Formats
- Finding Use Cases
- Related Artifacts