Use Cases

Reference: Craig Larman, Applying UML and Patterns, Ch. 6

Use Case

- What it is:
  - Text story
  - Widely used to discover and record (mostly functional) requirements
- What is it about:
  - Some actor(s) using a system to meet specific goals
  - Answering questions:
    - Who is using the system, what are their typical scenarios of use, and what are their goals?
- What it is NOT:
  - Not object-oriented
  - Not a diagram
    - UML use cases diagrams are “secondary-value” artifacts
- Focus: use cases, not use case diagrams

Example: Point of Sale

1. Customer arrives at a checkout (+goods).
2. Cashier uses POS system to record items.
3. System presents a running total and line-item details.
4. Customer enters payment information, which the system validates and records.
5. System updates inventory.
6. Customer receives receipt from the system and leaves.

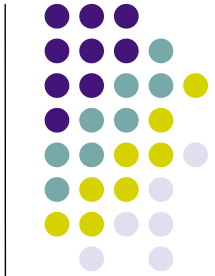
Actors, Scenarios, and Use Cases

- **Actor**: entity that shows a behavior,
  - e.g.: a person (role), computer system, or organization
- **Scenario:** specific sequence of actions and interactions between actors and a system
  - use case instance
  - singe path of using the system
  - e.g., purchasing 10 items with cash (or even more detailed)
- **Use case:** collection of related success & failure scenarios that describe an actor using a system to support a **goal**

Use Case Example with Scenarios (casual format)

- **UC 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 ...
  - If the item identifier is not found in the system …
  - If the system detects failure to communicate with the external accounting system …

Use-Case Model



- Set of all written use cases
- Model of the system's functionality and environment
- Unified Process (UP) defined artifact within the requirements discipline
  - UP also requires glossary.
- May optionally include a UML use case diagram
  - use cases, actors, and their relationships
  - context diagram

Three Kinds of Actors

- Primary actor
  - has user goals fulfilled through using services of the system under discussion
  - drives the use cases
- Supporting actor
  - provides a service to the system under discussion
  - e.g., payment authorization service
    - implies: clarification of external interfaces and protocols needed
- Offstage actor
  - has an interest in the behavior of the use case, but is not primary or supporting
  - e.g., a government tax agency

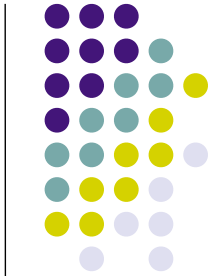
Use Case Format

- **Brief**
  - Succinct one-paragraph summary
  - usually the main success scenario
  - done during early requirements analysis
  - should take only a couple of minutes
- **Casual**
  - informal paragraph format
  - multiple paragraphs covering various scenarios
- **Fully dressed**
  - details all steps and variations
  - includes supporting sections such as preconditions and success guarantees
  - mainly done after many use cases are identified and during early requirements workshop for high-value and high-risk requirements (e.g., core architectural)

A Template for Fully Dressed Style

- **Use case name**
  - start with a verb
- **Scope**
  - the system under design
- **Level**
  - user goal or subfunction level
- **Primary actor**
  - calls on the system to deliver a service
- **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 (postcondition)**
  - 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 and failure
- **Special requirements**
  - related non-functional requirements
- **Technology and data variations list**
  - varying I/O methods and date formats
- **Frequency of occurrence**
- **Miscellaneous**

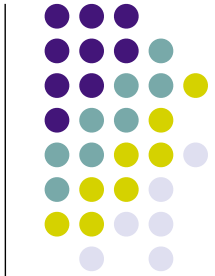
Coffee Maker Example



Example of a “semi” fully dressed use case

CoffeeMaker

http://agile.csc.ncsu.edu/SEMaterials/tutorials/coffee_maker/



Template for a fully dressed use case

Write in an Essential Style (early phase)

- Keep the user interface out
- Focus on actor intent
- User's intentions and system's responsibilities rather than their concrete actions
- Example
  - Manage Users:
  - 1. Administrator identifies self.
  - 2. System authenticates identity.
  - Another is concrete style that embeds user interface decisions - avoid during early analysis
  - Example
  - 1. Administrator enters ID and Password in a dialog box

Write Black-Box Use Cases

- Focus on what the system must do,
  - i.e., the behavior or functional requirements
  - Not on how it will do (the design)
- Examples:
  - Good: The system records the sale
  - Bad: The system writes the sale to the database.
  - Worse: System generates SQL INSERT statement for the sale

Take an Actor and Actor-Goal Perspective

- Use case definition by Jacobson
  - 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** [Jacobson]
- Write requirements focusing
  - on the users/actors of a system,
  - asking about their goals and typical situations
  - and what they consider a valuable result

Actor-Goal List

Actor	Goal		Actor	Goal
Cashier	process sales process rentals handle returns cash in cash out ...		System Administrator	add users modify users delete users manage security manage system tables ...
Manager	start up shut down ...		Sales Activity System	analyze sales and performance data
...	...		...	...

One Column vs Two Column Format
Two column emphasizes interaction

Use Case UC1: Process Sale

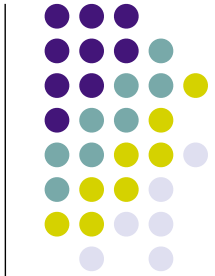
Primary Actor: ...

... as before ...

Main Success Scenario:

Actor Action (or Intention)	System Responsibility
1.Customer arrives at a POS checkout with goods and/or services to purchase.	
2.Cashier starts a new sale.	
3.Cashier enters item identifier.	4.Records each sale line item and presents item description and running total.
Cashier repeats steps 3-4 until indicates done.	5.Presents total with taxes calculated.
6.Cashier tells Customer the total, and asks for payment.	
7.Customer pays.	8.Handles payment.
	9.Logs the completed sale and sends information to the external accounting (for all accounting and commissions) and inventory systems (to update inventory). System presents receipt.
...	...

How to Find Use Cases?



- Choose the system boundary
  - what you are building?
  - who will be using the system?
  - what else will be used that you are not building?
- Find primary actors and their goals
  - brainstorm the primary actors first
  - who starts and stops the system?
  - who gets notified when there are errors or failures?
- Define use cases that satisfy user goals
  - prepare an actor-goal list (and not actor-task list)
  - in general, one use case for each user goal
  - name the use case similar to the user goal

What Tests Can Help Find Useful Use Cases?

- Which of these are valid use cases?
  - Negotiate a Supplier Contract
  - Handle Returns
  - Log in
  - Move Piece on the Game Board

What Tests Can Help Find Useful Use Cases?

- Which of these are valid use cases?
  - Negotiate a Supplier Contract
  - Handle Returns
  - Log in
  - Move Piece on the Game Board

- All of these can be use cases
  - at different levels,
  - depending on the system, boundary, actors, and goals

What Tests Can Help Find Useful Use Cases?

- Rather than asking
  - ”What is a valid use case?”
- More practical question:
  - “What is a useful level of focus to express use cases for application requirements analysis?”
- Rules of thumb
  - The Boss Test
  - The EBP Test
  - The size test

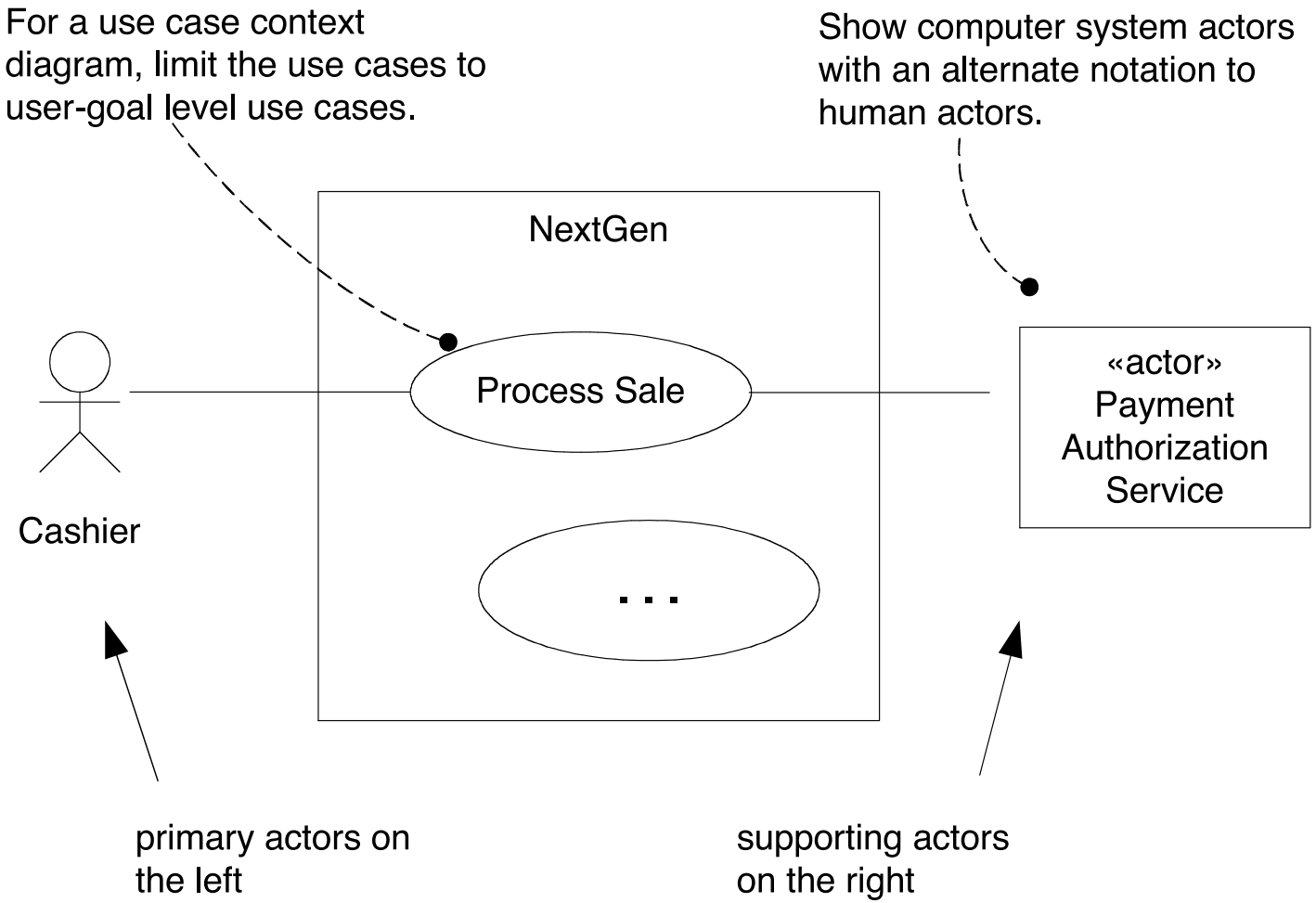
What Tests Can Help Find Useful Use Cases?

- The boss test
  - "What have you been doing all day?"
    - Your reply "logging in!"
      - Is your boss happy? No value? No good use case!
- The Elementary Business Process (EBP) test
  - a task performed by one person in one place at one time, in response to a business event, which adds measurable business value and leaves data in a consistent state
  - Good Examples: Approve Credit or Price Order
  - Bad Examples: delete a line item or print the document
- The size test
  - Just a single step in a sequence of others -> not good!

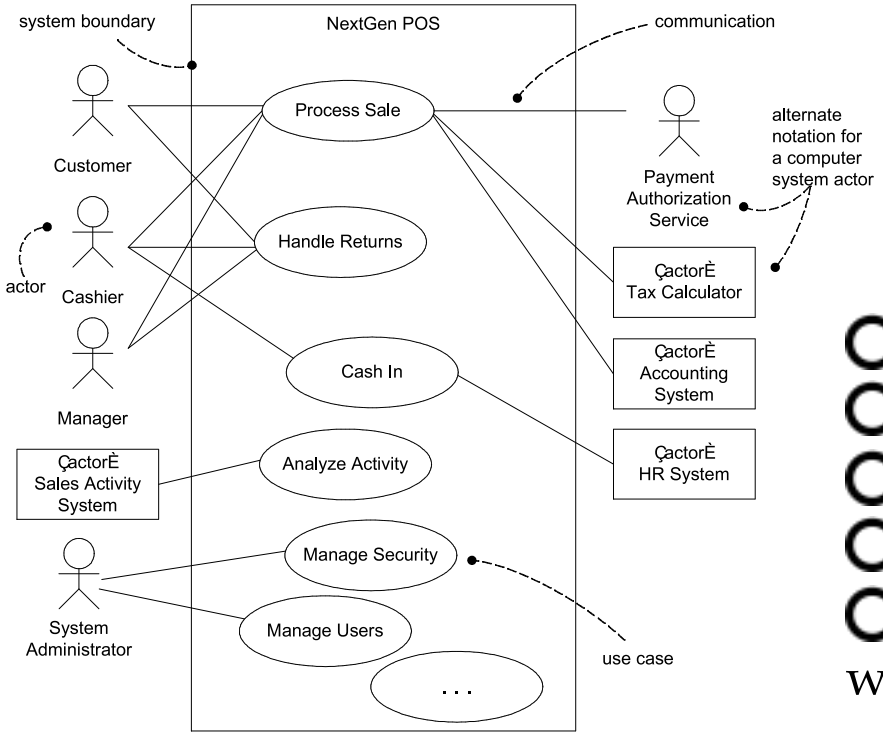
Applying Tests

- Negotiate a supplier contract
  - Much broader than EBP, rather a business use case
- Handle returns
  - OK with the Boss. EBP. Size is good.
- Log in
  - Boss is not happy is this is all you do all day!
- Move piece on game board
  - Single step – fails the size test.

Use Case (Context) Diagrams: Suggested Notation



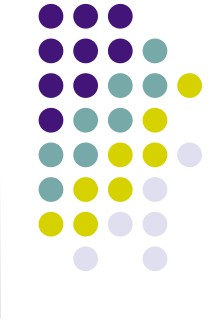
Use Case Diagrams

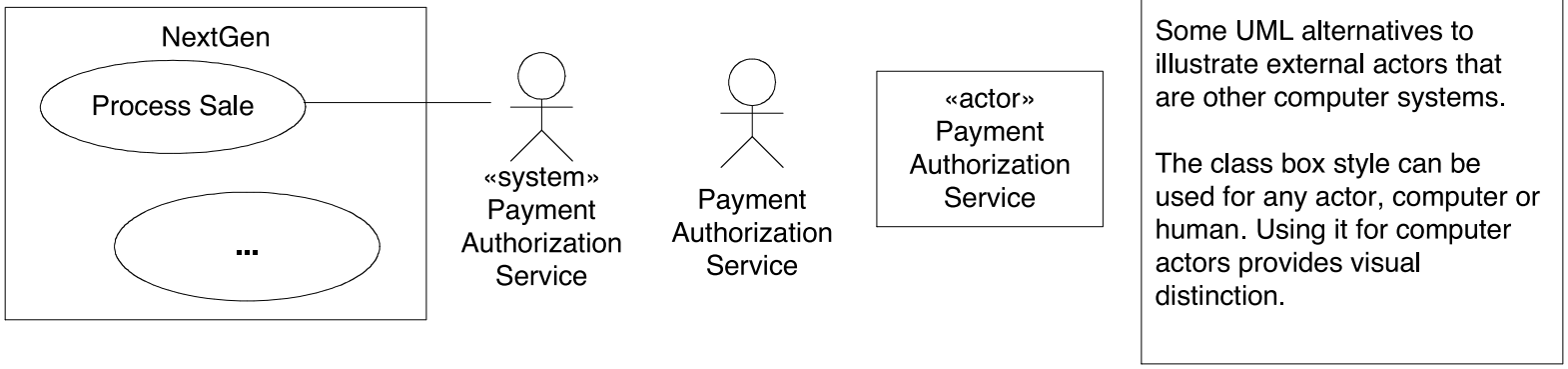


- Context diagram of the system
- Shows boundary
- What lies outside of it
- How it gets used
- Should be done in conjunction with an actor-goal list

- Downplay diagramming, Keep it short and simple
- Focus on text
- Do not focus on use case relationships

Alternative Actor Notation

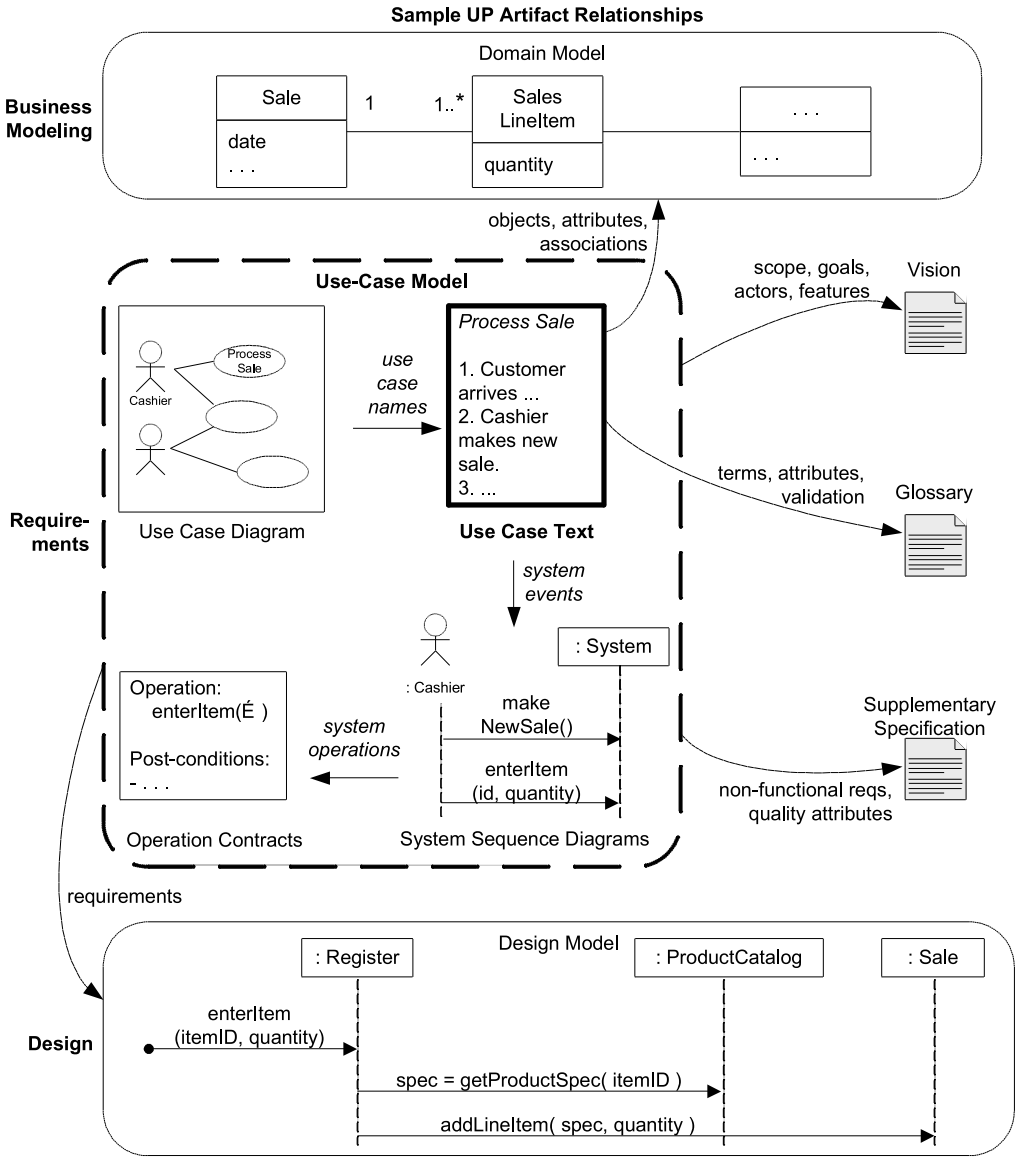




Some UML alternatives to illustrate external actors that are other computer systems.

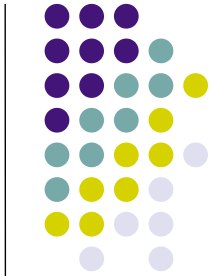
The class box style can be used for any actor, computer or human. Using it for computer actors provides visual distinction.

Use Cases form Basis for Others



Use Cases in Iterative Development

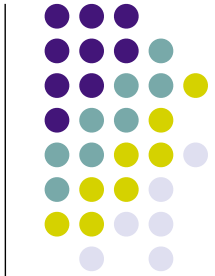
- Functional requirements are primarily captured in use cases
- Use cases drive the iteration planning and work
- Easy for users to understand
- Influence user manual/documentation
- Functional or system testing corresponds to the scenarios of use cases
- Independent of implementing technology
- UI shortcuts for most common scenarios



More examples?

iTrust

http://agile.csc.ncsu.edu/iTrust/wiki/doku.php



- Questions / Comments / Thoughts?

Credits

- Contents adopted from Applying UML and Patterns – Larman and www.craiglarman.com