Applying UML & Patterns (3rd ed.)

Chapter 6

USE CASES

Dr. Glenn L. Ray

School of Information Sciences

University of Pittsburgh

gray@sis.pitt.edu 412-624-9470

Use Case Overview

- UCs capture a system's functional reqs
  - Functional = algorithmic
  - Think 'workflow' or 'procedure'
  - UC is a textual description of an algorithm
    - Can be graphical but usually its textual

Use Case Overview

- Official model in the UML
- An accepted best practice & widely used
- But there is no generally accepted way of writing UCs!!
- How can UCs be best practice when there's no official specification of a well-formed UC?

Use Case Overview

- How to quality control UCs?
- How do users verify UCs?
- How do UCs integrate with other models?
- Indicates how much SW engineering needs to evolve

Use Case Overview

- Jacobson's definition- A set of UC instances (scenario), where each instance (scenario) is a sequence of actions a system performs that yields an observable result of value to a particular actor
- Encourages focus on achieving user goals
- A UC is a dialogue between an actor (aka 'user') and the system .
- The dialogue is presented as an *ordered* sequence of steps
- A complete sequence of steps is a use case scenario
  - A scenario forms a complete path thru the UC, aka UC instance.

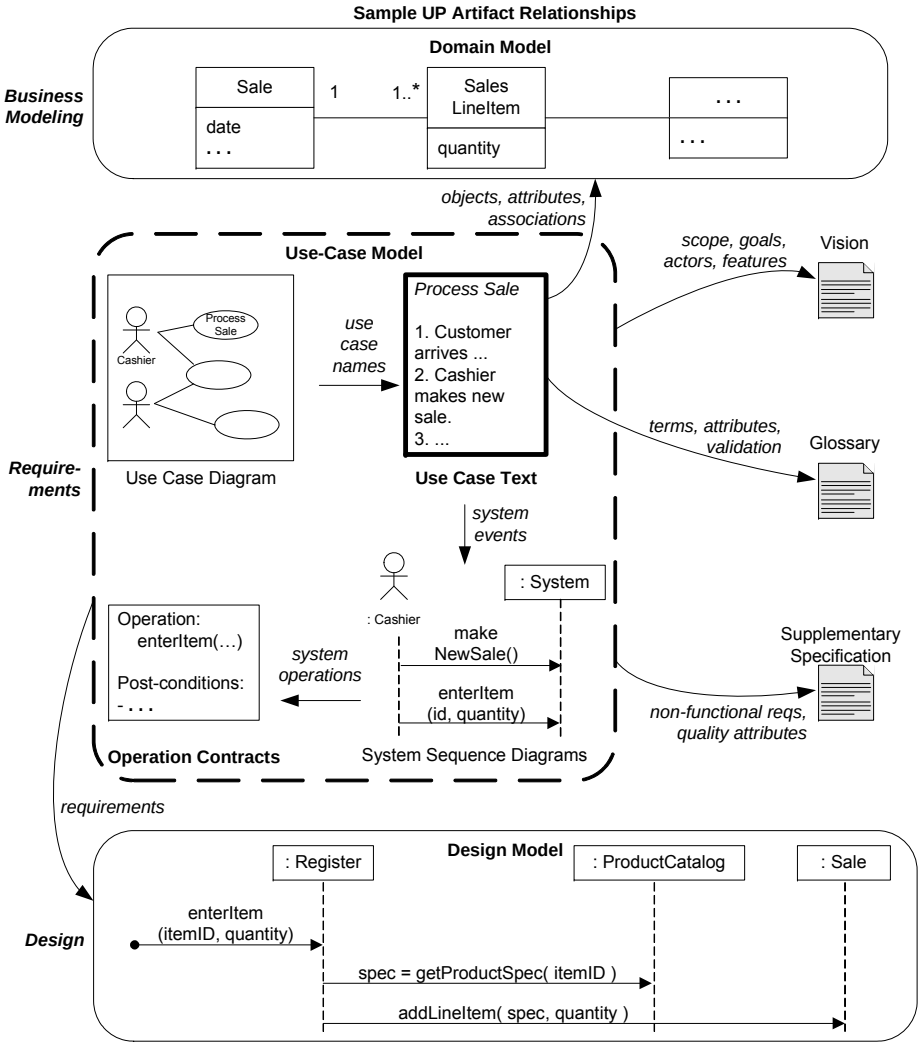
Use Cases

- UC can contain multiple scenarios (i.e., >1 path thru UC)
- Can range from simple (brief summary) to elaborate (detailed steps using adopted document template)
- UCs are NOT object-oriented artifacts! They feed into other OO models

Use Cases

- UCs are part of UML but it only specifies UC diagrams, which are low value
- UCs are written in simple concise text
- Complex UCs should probably be depicted using UML activity diagrams (we'll cover activity diagrams later)

Fig. 6.1



Use Cases

- UC strengths & uses
  - Emphasize user goals & objectives
  - Decompose system functionality into a set of discrete tasks (divide & conquer)
  - Easy for users to understand
  - Can be reused for user documentation
  - Basis for planning (estimating) work during each iteration
  - Guide developers during implementation
  - Test cases can be taken directly from UCs
  - Independent of implementing technology

Use Cases

- Actor is anything with behavior that triggers UC execution
  - Primary Actor accomplishes goal via UC
    - Typically a human user, also concepts like Time
  - Secondary Actor provides a service to UC
    - 3rd party system
  - Offstage actor is interested in UC results
    - Regulatory agency

Use Case Structure

- You can find lots of UC templates in books & on the WWW
  - Alistair Cockburn has published on UCs
- Some UC advise is contradictory
- Specific format depends on needs of project
- If company has an official template, use it!
- Here is an acceptable general purpose UC structure to be used in this course

Generic UC Structure

- UC Name in ‘Verb Noun’ format
  - Open Account, Enroll Student
- Description
  - 1 paragraph high-level description of the UC
- Actors
  - List all Actors who can trigger use case
- Preconditions
  - What must be true before use case can be triggered
  - Perhaps another UC must precede this one
- Postconditions
  - What must be true upon completion of the UC
- Base Scenario
  - Steps describing the sequence of actions that must occur assuming no alternative flows or error conditions
  - ‘Sunny day scenario’

Generic UC Structure (Cont'd)

- Alternative Scenarios
  - State condition that precipitates branching
  - Specify sequence of steps that occur as a result
- Additional Information
  - Any other information or constraints that impact the UC
- Key terms should be defined in a glossary, or possibly data dictionary

Generic UC Structure (Cont'd)

- **Please note**:
  - All UCs written for this course must use the UC template document on the course website!

Use Cases-Base Scenario

- **Essential course guidelines for writing UCs**
  - Minimize verbosity; keep it simple, stupid!
  - Use terminology from business domain
  - Avoid IT jargon / implementation concepts
  - **Never** refer to mouse clicks, GUI widgets, screens, databases, etc.
  - Each step begins with ‘System’ or ‘User’
  - Avoid using ‘System’ & ‘User’ in same step
  - Avoid articles (‘a’, ‘an’, ‘the’)
  - Present tense
  - Active voice
  - 1-2 actions/step w/ no if/else logic
  - Adhere to triplet structure (discussed below)

Use Cases

- Quality control guidelines
  - No violations of writing guidelines
  - It is functionally correct & complete
    - No missing scenarios
    - For each scenario, all steps in proper sequence with no missing steps
  - Users can verify with confidence

Use Cases

- What's wrong with these steps?
- How can they be improved?

  - 'Cashier enters item identifier'
  - 'System displays main menu'
  - 'User enters an item identifier'
  - 'System requests password from user
  - 'Password is entered by user'
  - 'System generated error log'
  - 'User clicks on submit button'
  - 'System executes load_DB module'
  - 'System asks user for password'
  - 'Rating engine calculates the premium due'
  - 'System checks input for errors'

Use Cases

- Larman's Process Sale UC (p. 68) is very elaborate and complex
- Actors/roles are hard-coded into UC
  - What if they change?
  - Big maintenance headache
- Contains many UC scenarios
- Difficult to comprehend in its entirety
- Should either decompose into smaller UCs
- Or, depict in an activity diagram

Larman's UC-Concise Alternate Version

UC: Process Sale

1. User selects new sale option
2. System requests item identifier
3. User enters item identifier
4. System records sale of item, and
5. System displays item description, price, current total
6. *Steps 2-5 repeated until user finished*
7. User selects sale finished option
8. System displays total and taxes due
9. User selects payment option
10. System requests payment information
11. User enters payment information
12. System handles payment
13. System logs completed sale and sends sale information to Accounting System and Inventory System
14. System generates receipt

Use Cases

- Alternative process sale UC
  - Is this UC more readable?
  - Can you suggest improvement?

Use Cases

- Scenarios exhibit a logical sequence of triplets

    1. System requests information….
    2. User enters information…
    3. System responds to information

- Does Larman's UC follow this style?
- What's missing?

Use Cases

- Scenario triplets are conceptual
  - At least 3 steps / triplet, possibly more
  - Some steps, e.g., a loop, don't belong to a triplet

1. System requests item identifier & quantity
2. User enters item identifier & quantity
3. System verifies item identifier and records sale of item
4. System presents item description, price, and current total

- System response is split into 2 steps to improve readability

Use Cases

- Use bullet list to simplify step that includes several data items
- Poor example
  1. System requests customer info
  2. User enters first name, middle initial, last name, street address, city, state, zip
  3. System verifies info and creates a new customer
- Harder to QA items in comma-separated list

Use Cases

- Use bullet list to simplify step that includes several data items.. No need to repeat list in step 3
- Better example
  1. System requests customer info
  2. User enters
     - First name
     - Middle initial
     - Last name
     - Street address
     - City
     - State
     - zip
  3. System verifies info and creates new customer

Use Cases

- The following version of Process Sale is written in triplet style but it contains a major error, raise your hand when you find it

Process Sale UC—My Version

UC: Process Sale

1. System provides make new sale option
2. User selects make new sale
3. System creates new sale
4. System requests item identifier & quantity
5. User enters item identifier & quantity
6. System verifies item identifier and records sale of item
7. System presents item description, price, and current total
8. *Steps 4-7 repeated until all items entered*
9. System requests 'is sale finished?'
10. User confirms sale is finished
11. System requests payment information
12. User enters payment information
13. System handles payment
14. System logs completed sale and sends sale information to Accounting and Inventory Systems
15. System generates receipt

Use Cases

- Time's up!!
- Let try this again, Here's another version of the same use case with the same error, raise you hand when you find it

Process Sale UC—My Version

UC: Process Sale

1. System provides make new sale option
2. User selects make new sale
3. System creates new sale

4. System requests item identifier & quantity
5. User enters item identifier & quantity
6. System verifies item identifier and records sale of item
7. System presents item description, price, and current total

8. Steps 4-7 repeated until all items entered

9. System requests 'is sale finished?'
10. User confirms sale is finished

11. System requests payment information
12. User enters payment information
13. System handles payment
14. System logs completed sale and sends sale information to Accounting and Inventory Systems
15. System generates receipt

Process Sale UC—My Version- Corrected!

UC: Process Sale

1. System provides make new sale option
2. User selects make new sale
3. System creates new sale

4. System requests item identifier & quantity
5. User enters item identifier & quantity
6. System verifies item identifier and records sale of item
7. System presents item description, price, and current total

8. Steps 4-7 repeated until all items entered

9. System requests 'is sale finished?'
10. User confirms sale is finished
11. System updates sale status to 'closed'

12. System requests payment information
13. User enters payment information
14. System handles payment
15. System logs completed sale and sends sale information to Accounting and Inventory Systems
16. System generates receipt

Use Cases

- Inserting a blank line between triplets improves ability to find basic errors!
  - Similar to using whitespace to make a program more readable

1. System requests….
2. User enters…
3. System responds….

1. System requests…
2. User enters…
3. System responds …

1. System requests…

Use Cases

- UCs can invoke other UCs
  - Very similar to calling a function
  - Permits reuse of UCs
  - Avoids duplication & simplifies maintenance
  - Use hyperlinks in UC to link to invoked UC
  - See UC template for correct format

Common UC Issues

- Granularity
  - Use elementary business process as guide

    EBP: A task performed by 1 user in 1 place at 1 time in response to a business event, that adds measurable value to the business and leaves data in a consistent state.

- Base scenario's first step
  - Writing steps that precede the functionality to be implemented clutters the UC; may confuse developers
  - Start UC at the boundary of the system

Common UC Issues

- Write ‘black box’ UCs
  - Defer implementation details
  - Avoid reference to specific technologies

- Finding UCs
  - Define system boundary
  - Identify Actors and their goals
  - Construct simple Actor-Goal Matrix (p. 65 in pdf, p. 84 in Lar 3rd ed)

Common UC Issues

- Questions to help find actors…
  - Who uses system
  - Who gets info from system
  - Who provides info to system
  - Who supports, maintains, administers system
  - What other systems interact with this system
  - Where in the enterprise is the system used

UC Strategy

- UCs are usually developed (and implemented!) iteratively during a project
- Different UCs may be in varying stages of detail
  - Attack high-value & high-risk UCs first!

- Here is a sensible iterative plan…
  1. Develop List of possible UCs & Actors
     - Name appropriately, e.g., 'Approve Application'
  2. Write high-level summary (~ 1 paragraph)
  3. Write fully dressed UC w/ base scenario only
  4. Write alternative UC scenarios, possibly iteratively

Fig. 6.2

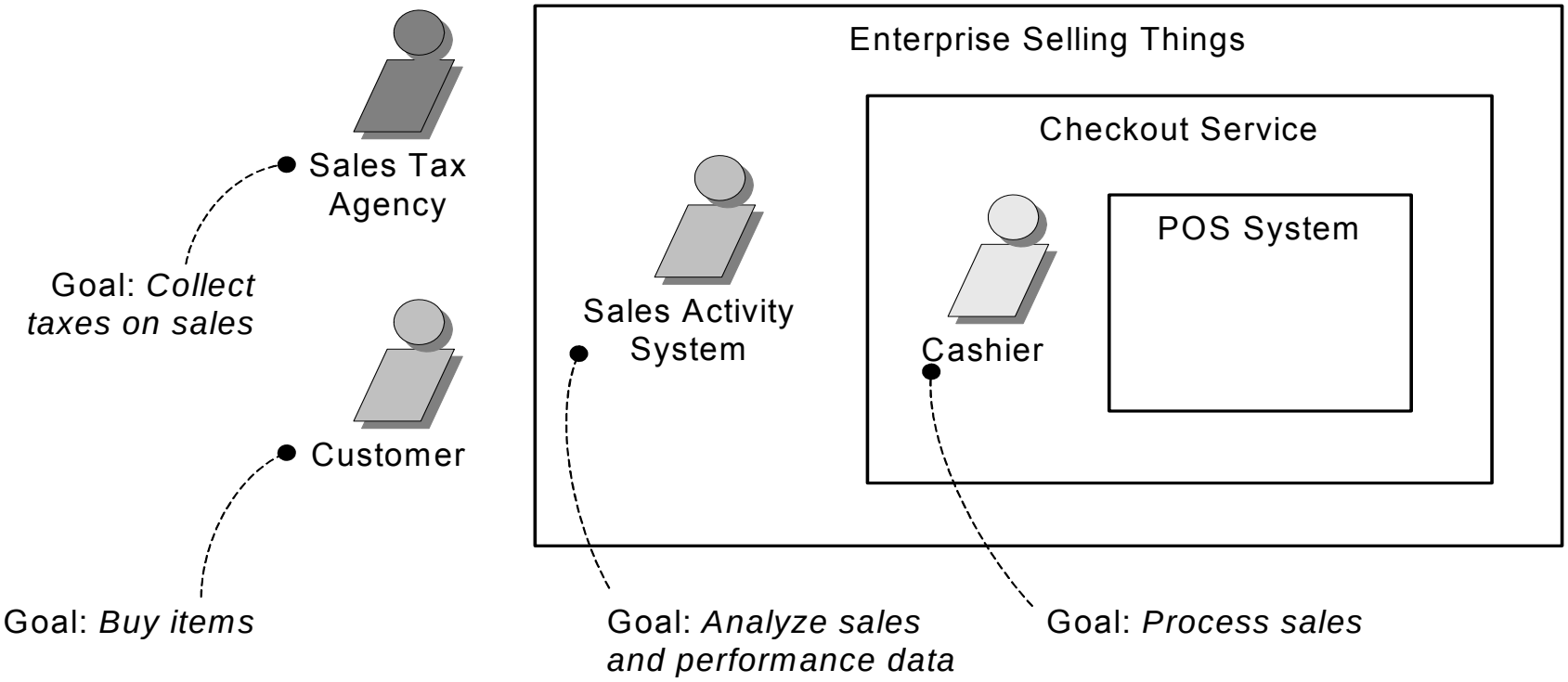


Fig. 6.3: UC Context Diagram

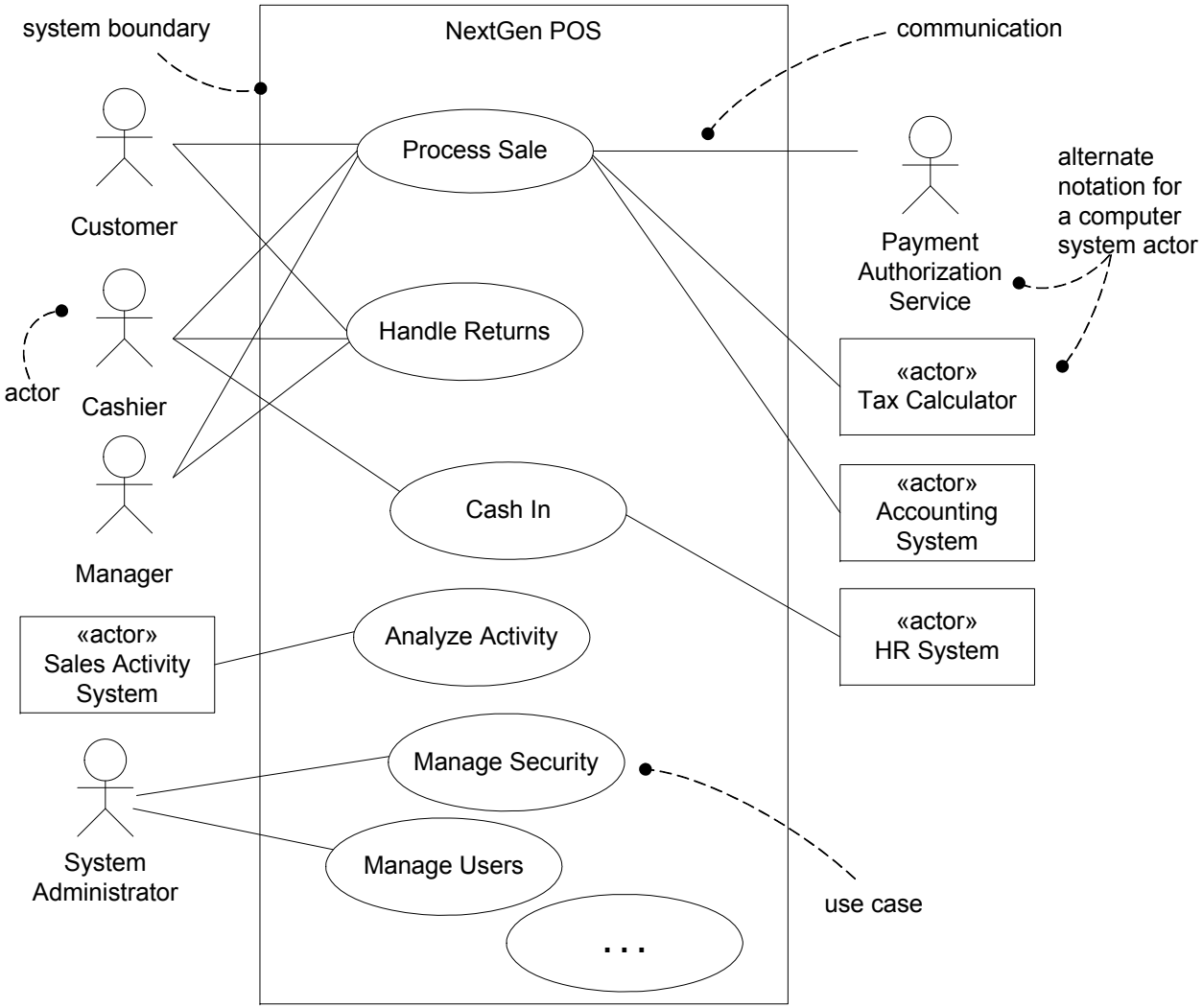


Fig. 6.4

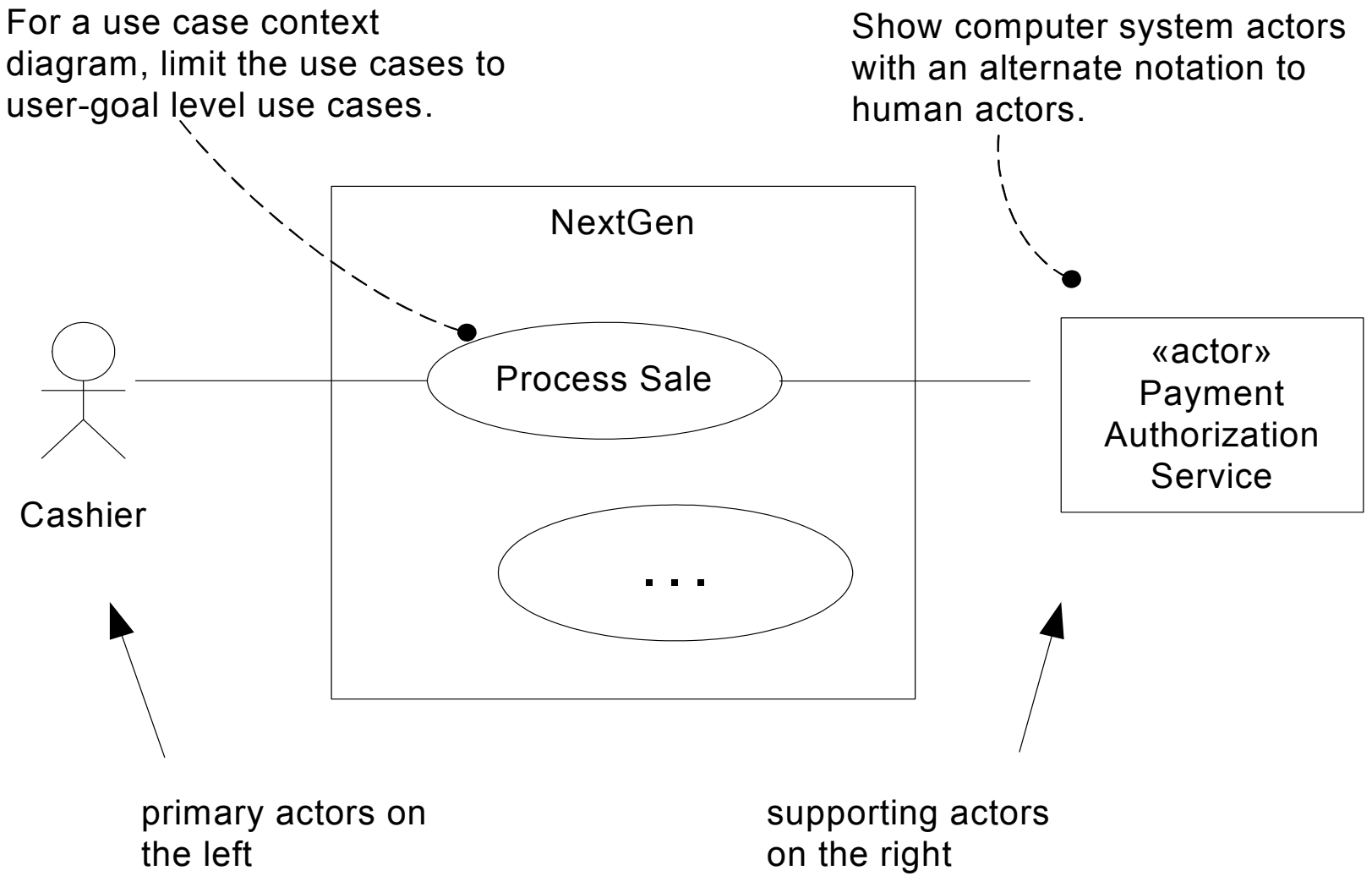


Fig. 6.5

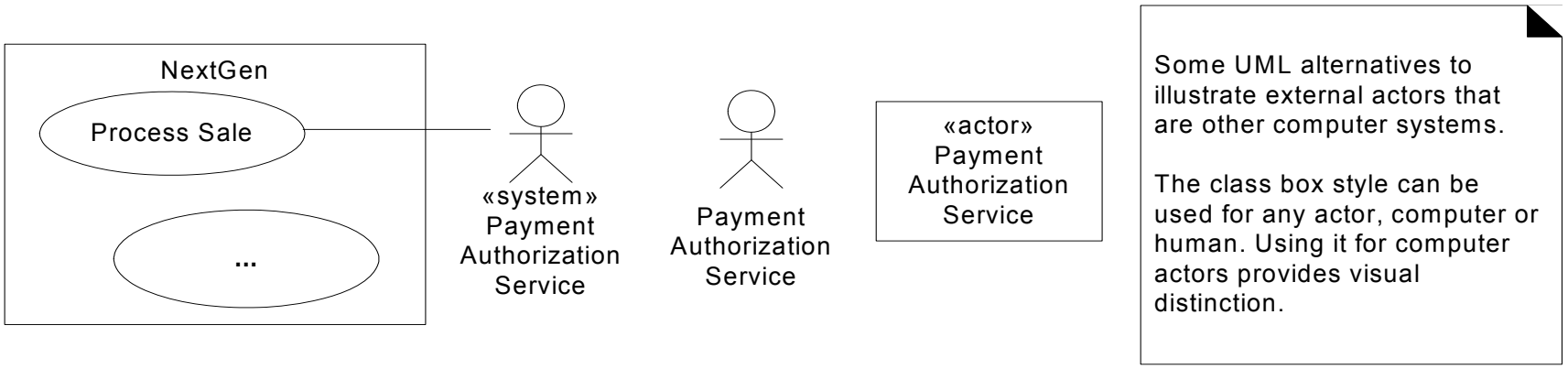


Fig. 6.6

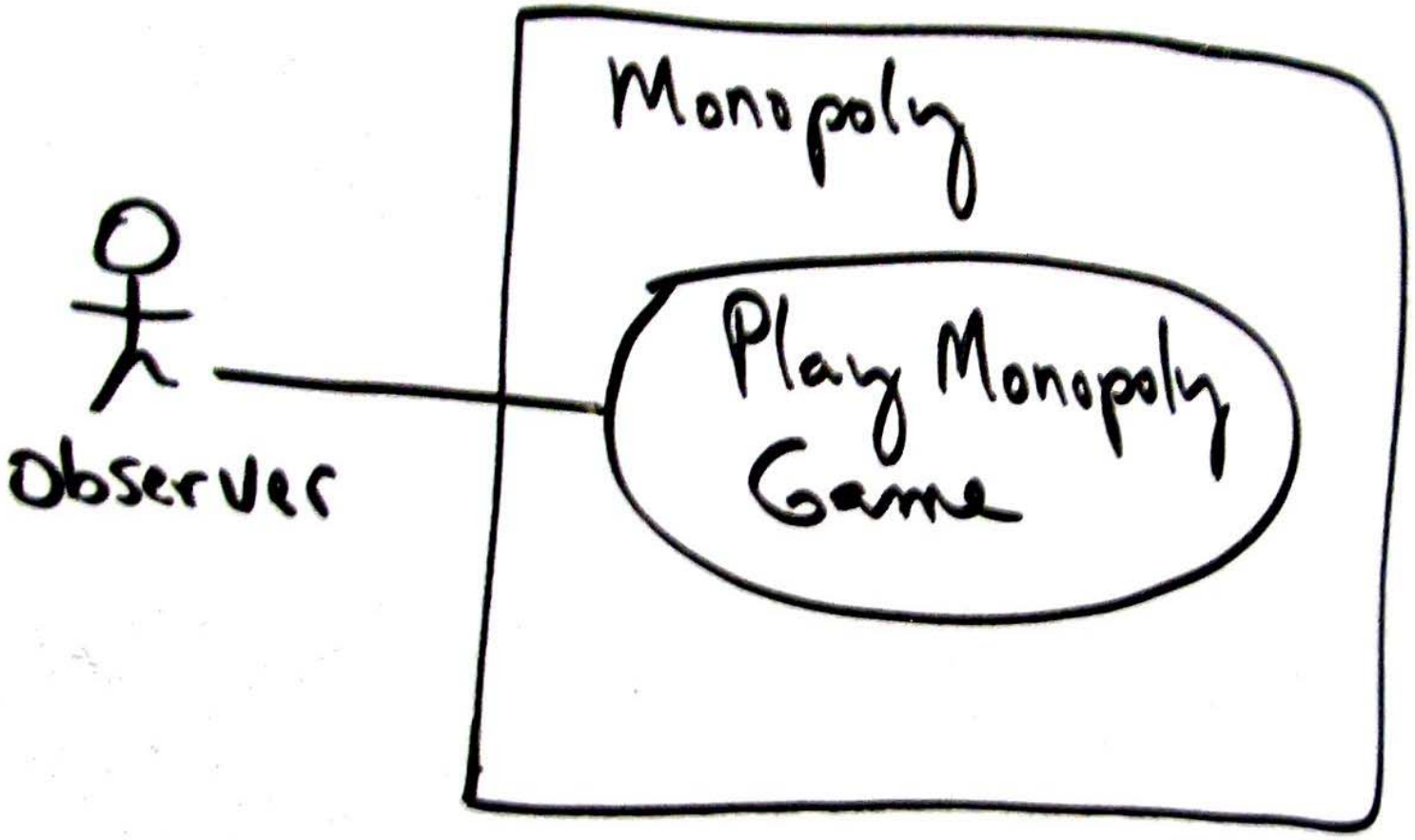


Fig. 6.6

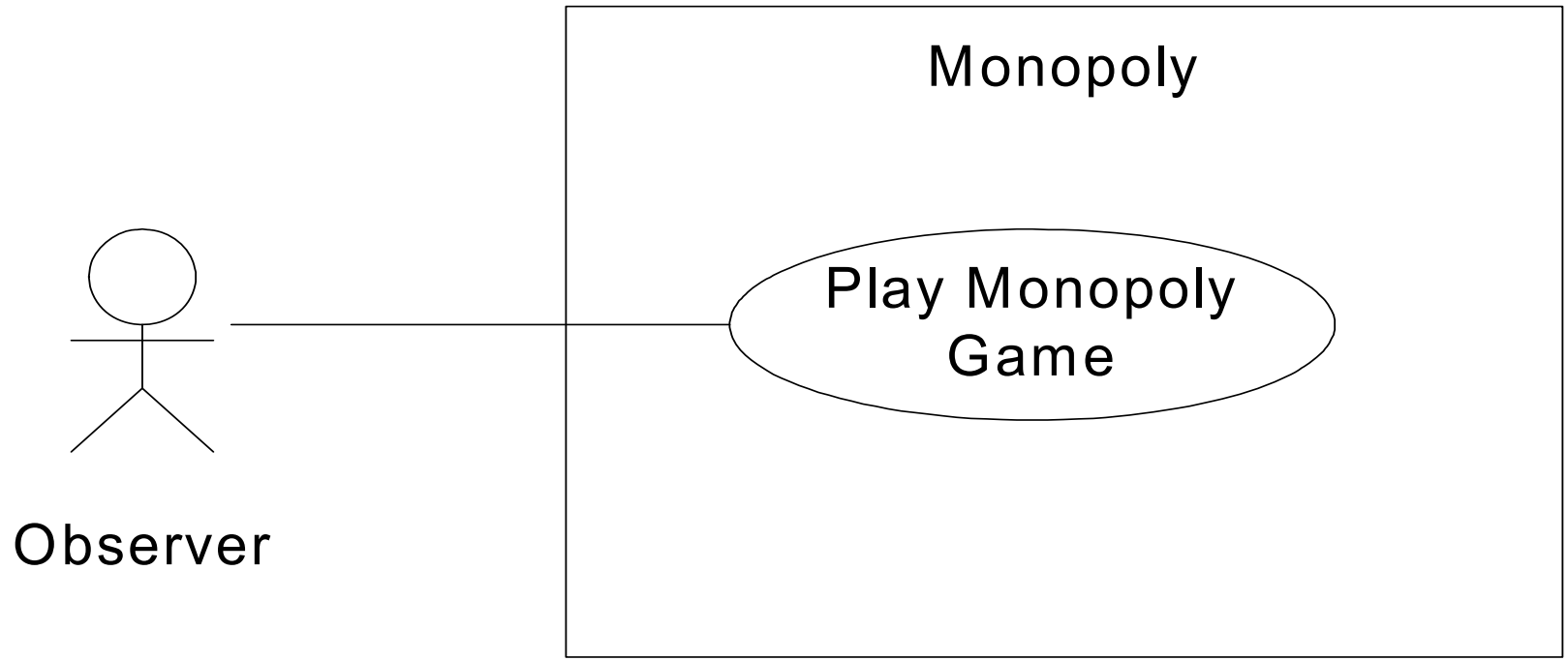


Fig. 6.7