SWEN30006
Software Modelling and Design

ITERATION 2—MORE PATTERNS

Larman Chapter 23

Objectives

On completion of this topic you should be aware of:

- The requirements for iteration-2 of the textbook case studies.

SWEN30006
Software Modelling and Design

Quick Analysis Update

Larman Chapter 24

Any sufficiently advanced bug is indistinguishable from a feature.

—Rich Kulawiec

Objectives

On completion of this topic you should be aware of:

- Analysis artefact changes for the textbook case studies, especially in the Monopoly domain model.

Case 1: NextGen POS System

Point-Of-Sale (POS) is a system to record sales and handle payments.

Includes hardware such as a register with bar code scanner and credit card reader, as well as software.

Features include:

- Interfaces to service applications, e.g. tax calculator, inventory control
- Fault-tolerant: at least capture sales and handle cash payments
- Flexibility in client-side terminals and interfaces
- Able to support different clients with different business rules, e.g. discounting policies

NextGen POS Requirements (I2)

1. Support for variations in 3rd-party external services, e.g. tax calculator, each with
   - a different API and protocol

   for a core of common functions.
2. Complex pricing rules
3. Refresh GUI when sale total changes

Case Study: NextGen POS (I2)

Use Cases:

- No change or refinement to existing cases

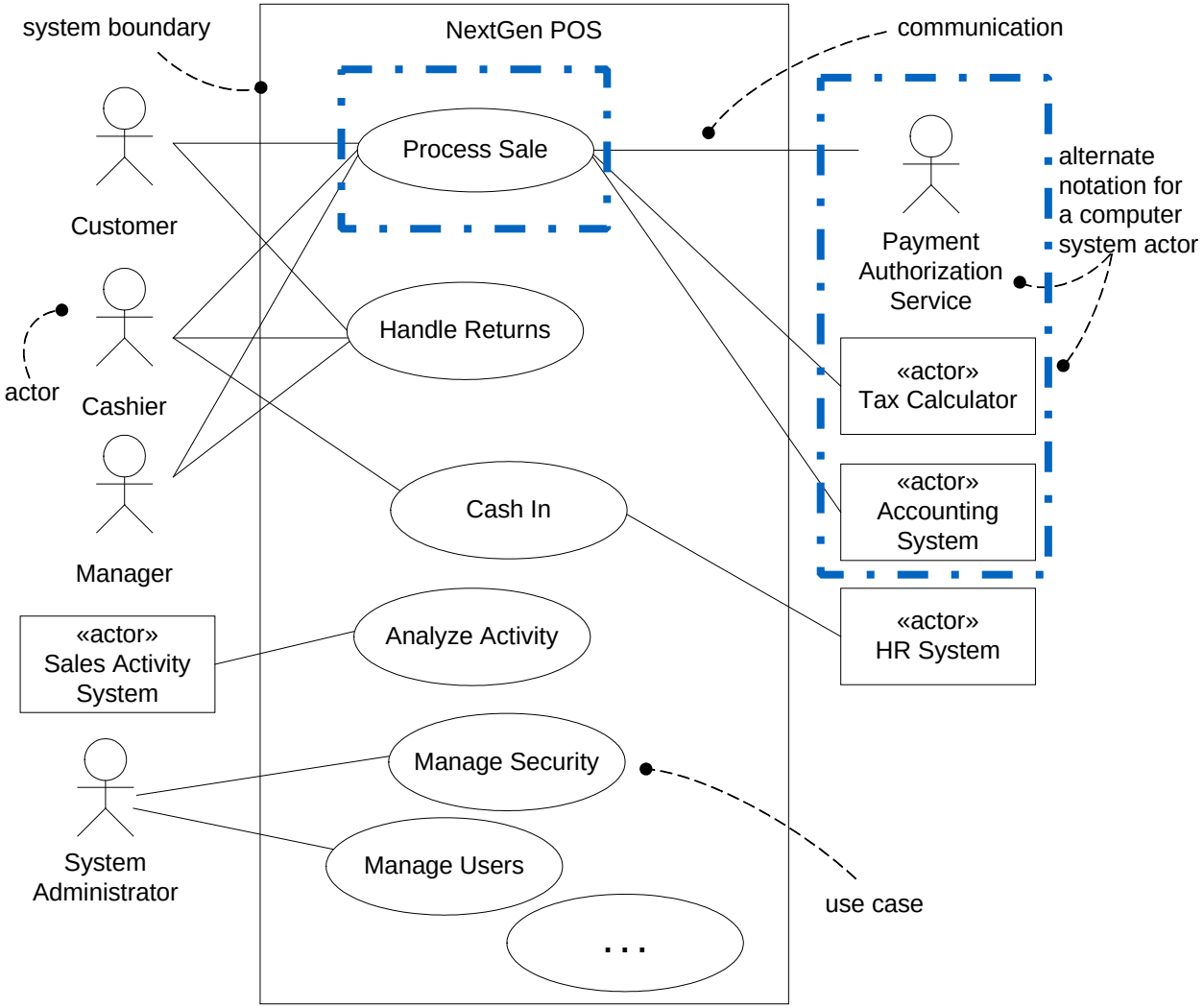
SSDs:

- Adding support for 3rd-party external systems
  - Eg. *TaxCalculator*, *CreditAuthorization*, *Accounts*
  - Intersystem collaboration → New system-level events

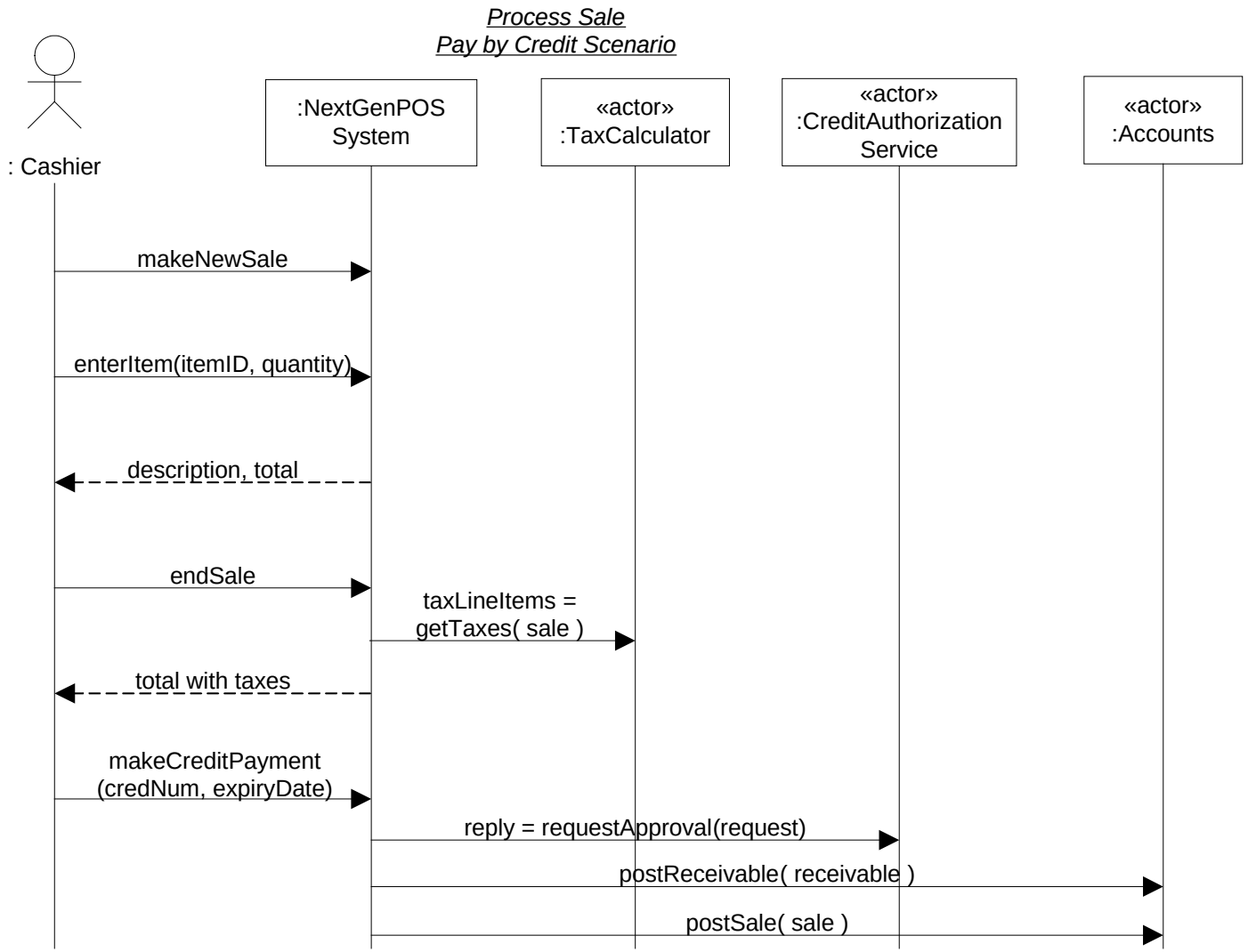
Domain Model:

- Additional external systems
- *PriceRule*, …

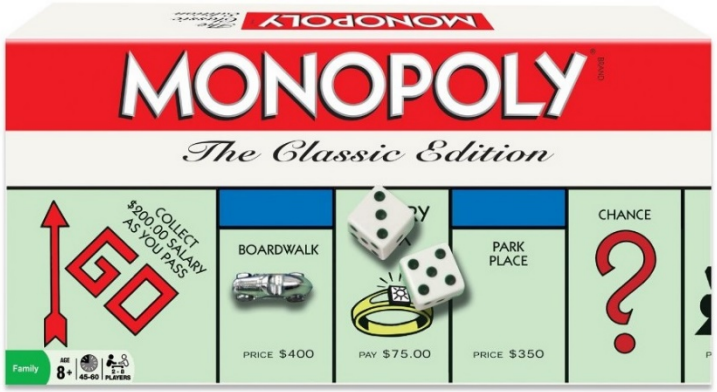
Use Case Diagram: NextGen POS



NextGen POS SSD: External Systems



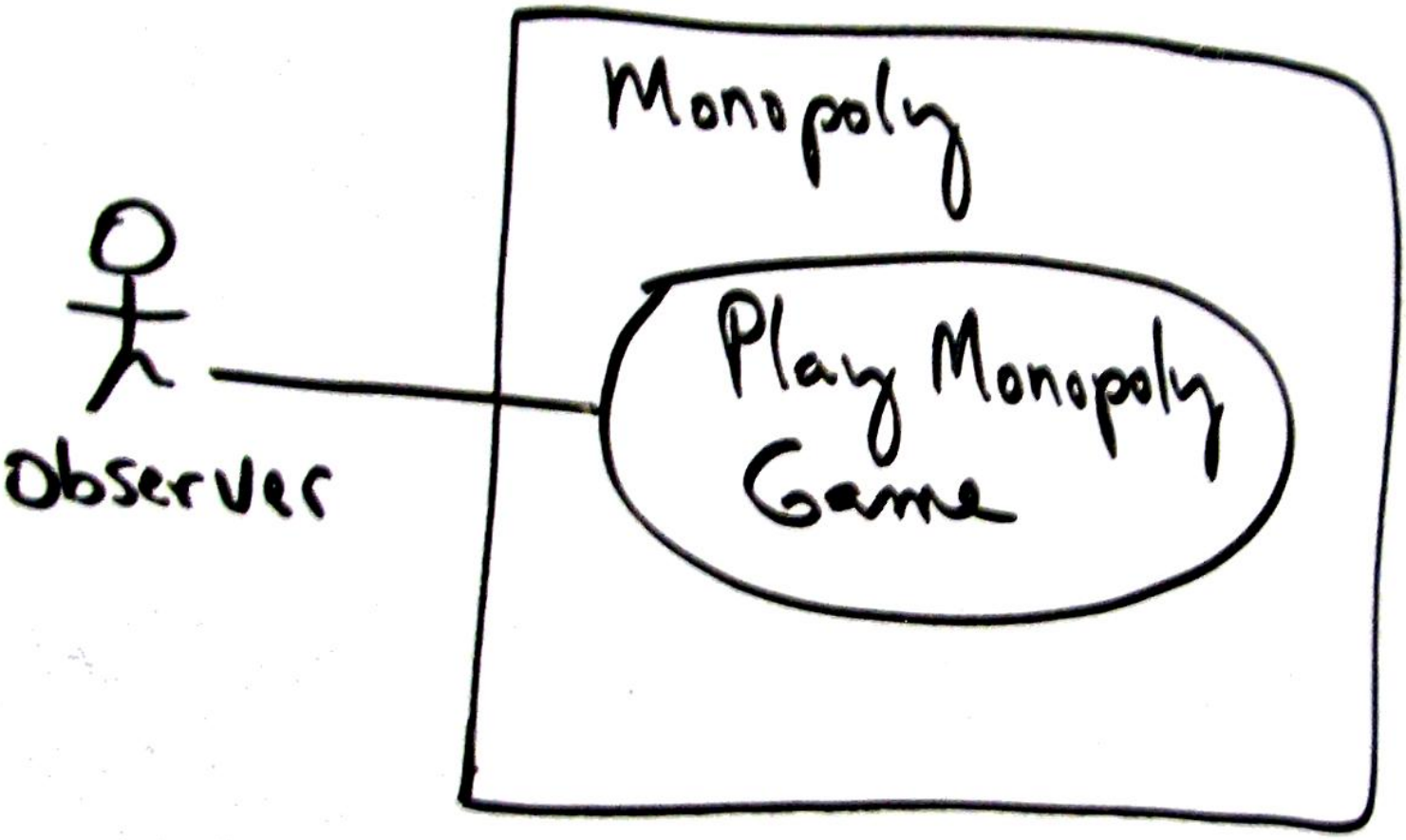
Case 2: Monopoly Game System



A Software Simulation of Monopoly

User starts off game and watches the activities of the simulated players.

Use Case Diagram: Monopoly



Monopoly Requirements (I2)

Same basic requirements for simulated game based on number of users, but with some special square rules:

1. Each player has $1500 at game start. Game has unlimited money.
2. When a player lands on Go, they receive $200.
3. When a player lands on Go-To-Jail, they move to the Jail then continue normally next turn.
4. When a player lands on Income-Tax, they pay minimum of $200 or 10% of their worth.

Case Study: Monopoly (I2)

Use Cases:

- Skipped, as rules for game known.

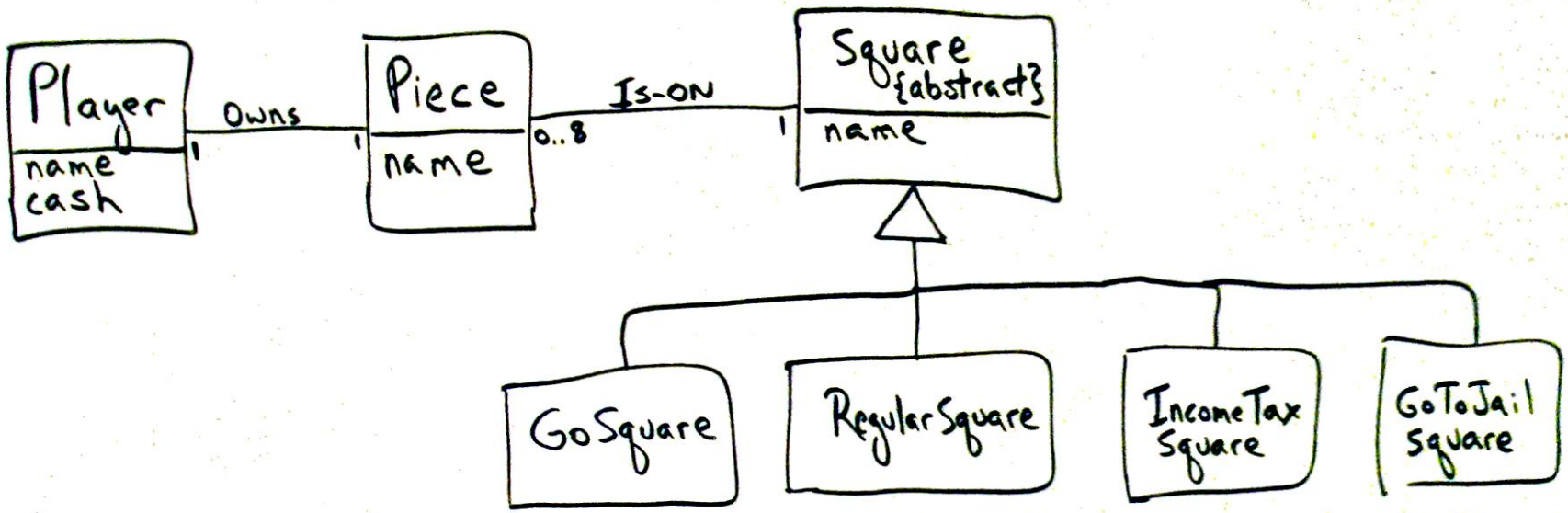
SSDs:

- No update required

Domain Model:

- Concepts: *Square, GoSquare, IncomeTaxSquare, GoToJailSquare*
- Suggests a **class hierarchy**

Monopoly Domain Changes



Generalization-Specialization Class Hierarchy (1)

Generalization:

- Identify commonality among concepts
- Define *superclass* (general concept)
- Define relationships with *subclasses* (specialized concepts)

Why?

- Economy of expression
- Reduction in repeated information
- Improved comprehension

Class Hierarchy (2)

Guideline for creating subclasses:

1. Subclass has additional attributes of interest
2. Subclass has additional associations of interest
3. Subclass is operated on, handled, reacted to, or manipulated differently to the other classes in noteworthy ways

Modelling Guidelines:

a) Declare superclasses abstract

b) Append the superclass name to the subclass

Monopoly Domain Changes