Object – Oriented Design with UML and Java

Part IX - Use Cases / User Stories

Version 2023

Requirements Specification

Your way of echoing to the customers what you heard them say they need.

Traditional, textual specification:

- System functionality is often well documented.
- The user interface is often under-specified.
- Depends heavily on skill of the writers.

Use Case modeling:

- Analysis tool that helps clarify the requirements.
- Models the users' view of the system.
- What the system should do to add value, not how to do it.

Genesis of Use Cases

- Developed by Ivar Jacobson, who wrote widely-read and highly-regarded books about their use.
- Jacobson and colleagues developed a software engineering tool, ***Objectory***, based on use case modeling.
- Folded directly into ***UML***.
- Commonly now referred to as ***User Stories***, and these can be scaled with “story points” as a rough estimate of difficulty.
- Widely considered the best unit of functionality for project planning because crucially, it has a focus on the customer.
- At a higher level, we might use ***Epics***, and ***Features***.

Example: Video Store

You are to design a system to track the activities of a video store.

- The store's inventory consists of new releases, old releases, bargain movies, and videos for sale. Each type has its own pricing and rental terms (length of rental and overdue policy).
- Customers must register with the store before they can rent videos. Registration information includes standard demographic information such as address, phone number, driver's license, etc.
- A customer under the age of 17 cannot rent videos with ratings of R, X, or NC-17, unless the store has an authorization slip on file, signed by a parent or guardian.
- The store tracks the videos that each customer currently has rented, which of them are overdue, and what outstanding overdue charges the customer is liable for.

Video Store Management Application

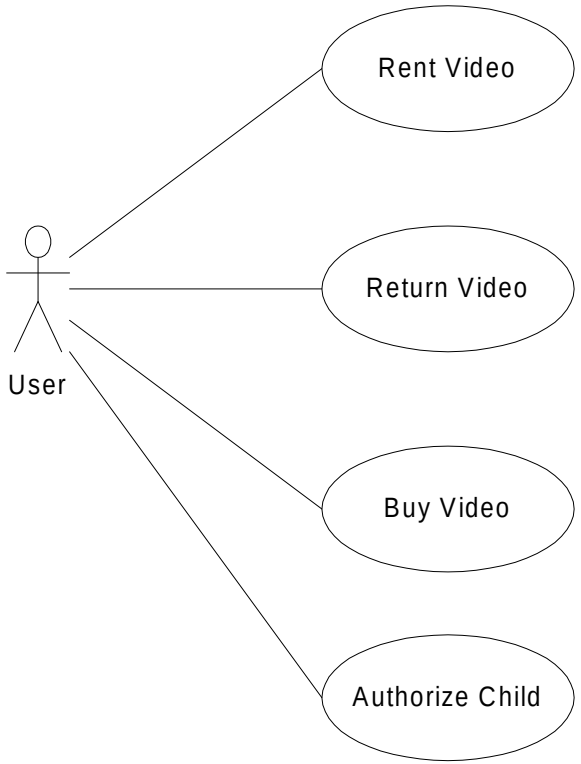
- When a video is rented, it is moved from in-store inventory to the customer's possession. When it is returned, it goes back into in-store inventory.
- When a customer rents one or more videos, a rental agreement is generated, and the customer pays the nominal rental fees for them. When the videos are returned, overdue charges are determined, where appropriate. The customer either pays the charges at that time, or they are accrued to his or her account.
- The store keeps track of how many times each specific video has been rented. This enables the store to track popularity, and retire tapes when they have been viewed many times.

Make a Use Case Model.

Write text describing the basic course of each use case along with any interesting variations (scenarios).

- *What types of requirements have not been addressed?*

Video Store Use Cases



- We will refactor this simple diagram as we iterate...

Use Case Driven Design

- Pick a use case / user story.
- Invent candidate system operations that will accomplish the use case.
- Identify important *scenarios* for each system operation.
- Create an interaction diagram for each interesting scenario.
- Base on this analysis, add the following to the class diagram:
  - New classes
  - Operations
  - Associations
  - Dependencies

Requirements: FURPS

- **Functionality**

  User Goals & Feature Set (Use Cases).

- **Usability**

  Human Factors, Aesthetics, Consistency, Documentation.

- **Reliability**

  Frequency/Severity of Failure, Recoverability, Predictability, Accuracy, Mean Time to Failure, Scalability.

- **Performance**

  Speed, Resource Consumption, Throughput, Response Time.

- **Supportability**

  Testability, Extensibility, Adaptability, Maintainability ($$$), Compatibility, Configurability, Serviceability, Installability, Localizability, Robustness, Flexibility.

Requirements Specification

- Possibly make screen mockups with a tool.
- Don’t be afraid to declare non-quantitative requirements:
  - Ease of use
  - Interface metaphor
    - » “The employee directory will be like a phone book, organized alphabetically by name.”
    - » “The employee phone book will be like an organization chart, organized by groups.”

An alternative to Use Case Modeling is to write a requirements document that *declares* what the system will have to do.

- Use Cases and FURPS complement one another.

Definitions and References

- Some definitions:
  - **Requirement**: “A condition or capability needed by a user to solve a problem or achieve an objective.”
  - **Testability**: “The degree to which a requirement is stated in terms that permit establishment of test criteria, and performance of tests to determine whether those criteria have been met.”
  - **System Context**: “Everything that lies outside of the system, including the users (actors), and sometimes the operating system, etc…”

Features are not the only kind of work that teams must prioritize. There is also **Technical Debt** reduction, **Risk Mitigation** (security concerns), and **Fixing Defects**.

Use Case Model

Use Cases organize ***Functional Requirements*** in ways meaningful to users, describing their interactions with the system in support of their goal(s).

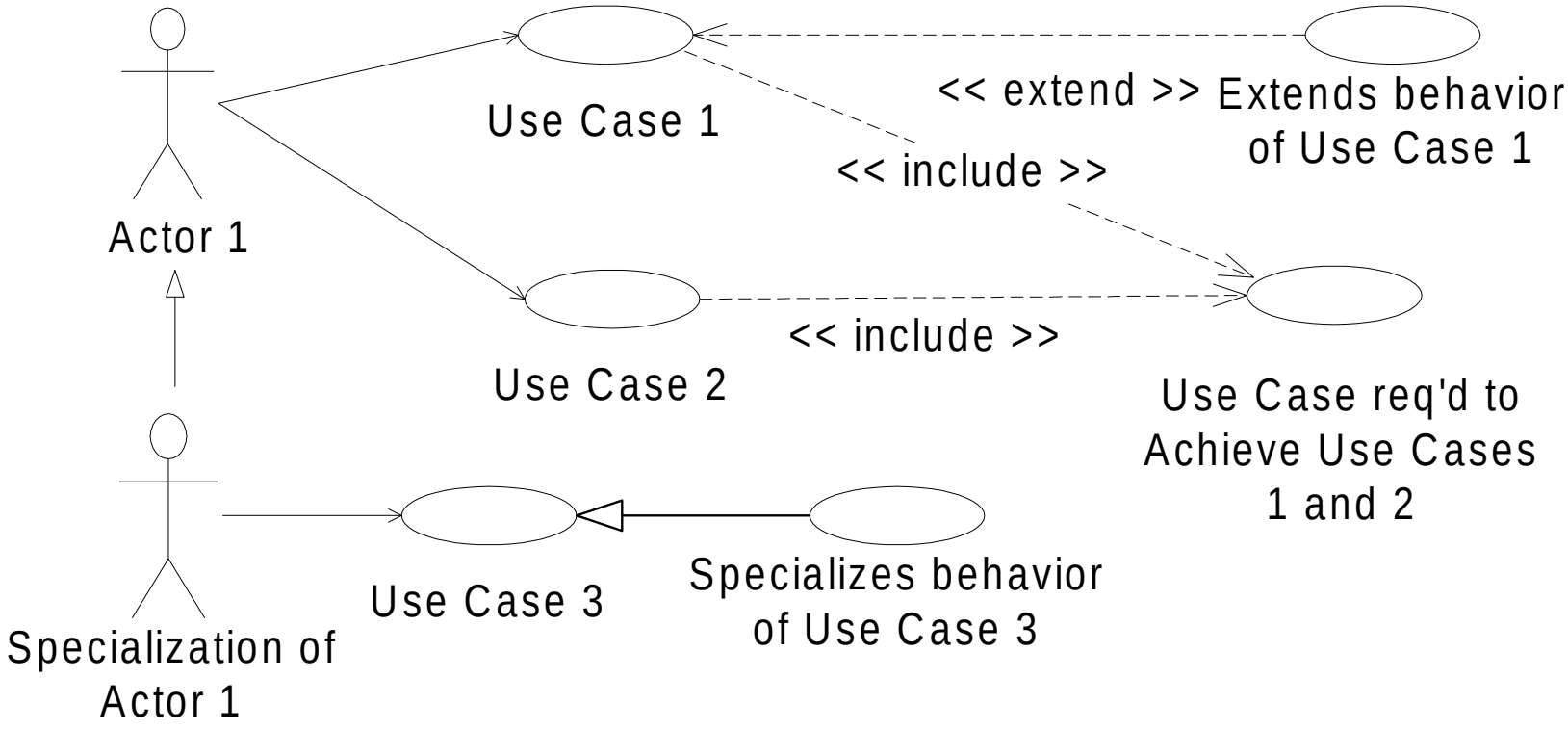
- ***Use Cases*** are high-level descriptions of system processes, workflows, or scenarios, designed to help ensure that the system behavior is what the users require. A Use Case specifies the smallest unit of functionality that should ever be delivered.
- ***Actors*** model the different user ***roles*** in a system, both people & machines. Examples: general user, system administrator, the payroll subsystem.
- The model describes ***what*** the system should do without specifying ***how*** it should be done. Additional text and UML models provide more details.
- A Use Case Model is NOT a complete requirements specification.

Use Cases and Scenarios

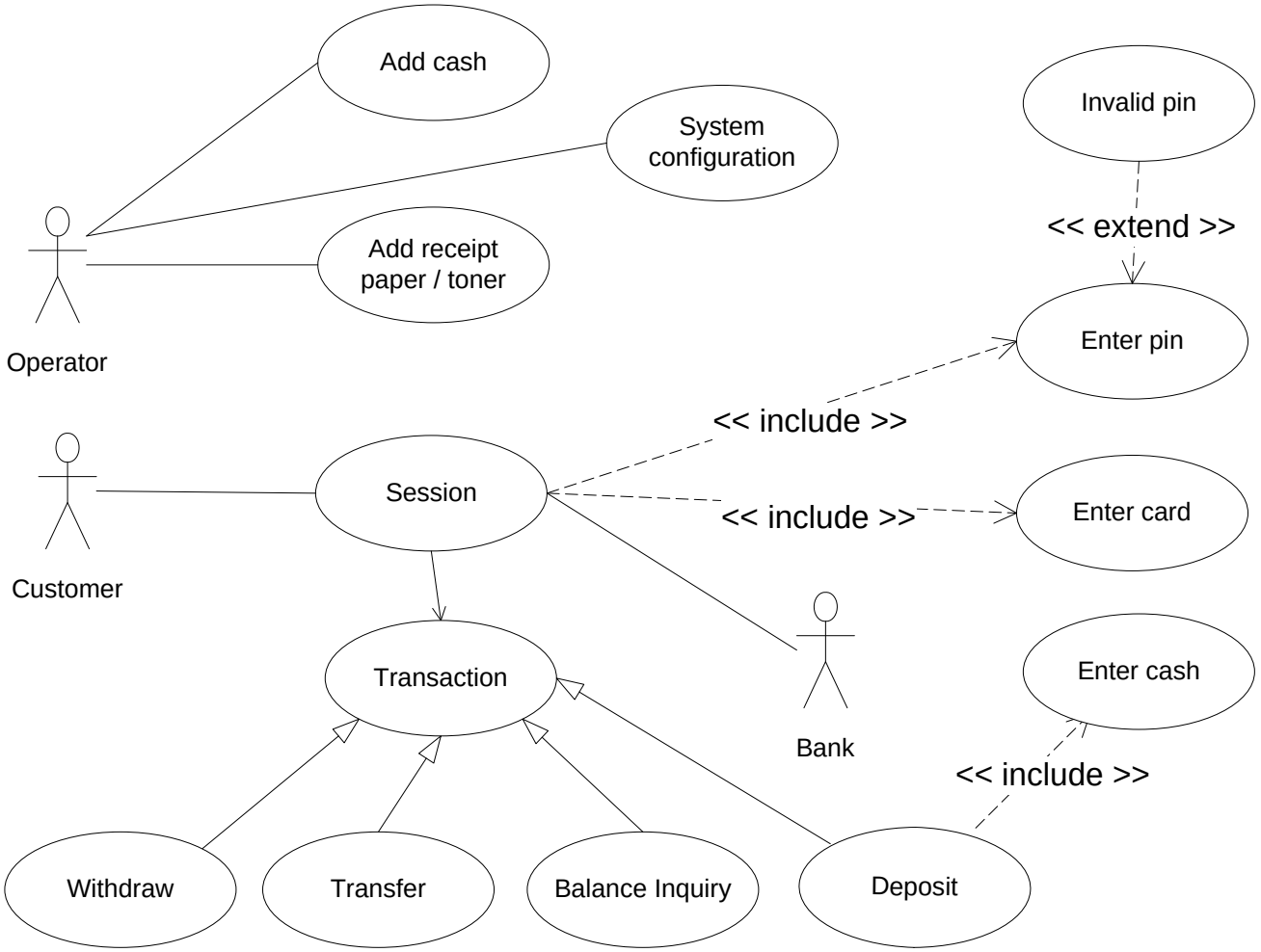
- A ***Scenario*** is an instance of a Use Case
  - Use Case: A customer rents a video.
  - Scenario 1: Rodney Randle rents Rambo.
  - Scenario 2: Rita Randle rents Roman Holiday using Rodney's account.
  - Scenario 3: Rodney Randle fails to rent Rambo because of too many overdue videos.
- Start with the "success" scenario, then add in the failure modes.
- Sequence Diagrams are useful to illustrate the dynamics for one scenario.
- Scenarios are used to validate the design, and serve as a basis for developing test cases. The models must support the functionality represented by the entire suite of scenarios.

As a <type of user>, I want <goal> so that <reason>….

UML Use Case Symbols



Use Case Diagram for ATM Machine



Example Use Case Documentation

Use Case Name: Generate Report XYZ

Author: David Leberknight

Last Revision: January 1st, 2000

Actors: XYZ Analyst.

User Goal: Create XYZ report.

Pre-conditions: XYZ data has been loaded.

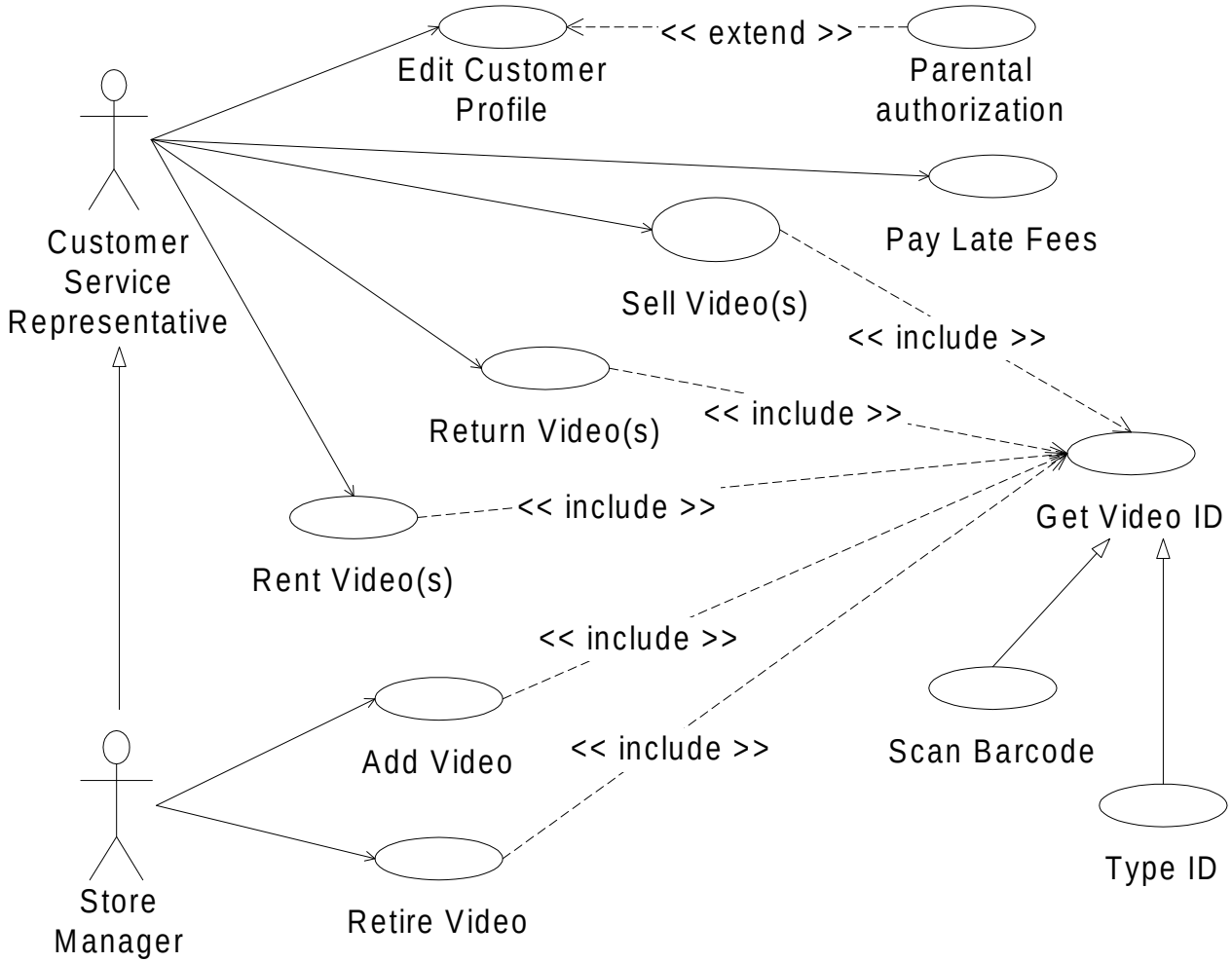
Basic Course:

- Allow the user to specify XYZ report parameters.
- Invoke the XYZ report generation process.

Post-conditions: An XYZ Report is created.

Error Conditions: ...

Video Store Requirements at a glance



Video Store Use Cases

1. Edit Customer Profile

Input new *or* updated demographic data.

Get customer ID *or* generate new one.

Write new *or* updated customer record to Database.

Maybe get parental authorization for a minor?

2. Get Parental Authorization for Minors to Rent R or X movies

< Extends: Edit Customer Profile >

Get parent and child customer IDs.

Update child's Database record showing the transaction.

Video Store Use Cases (cont.)

4. Rent Video(s)

Get customer ID.
Generate rental agreement with customer's ID and "today".
< Uses: Scan Video Barcode >
Verify that customer can rent video, *if* rated X or R.
Note video ID on rental agreement, along with fee paid.
Inform Customer of length of rental and overdue policy.
Get next video being rented, *if* any. *Iterate*.
Mark each video as rented and save new agreement.

Video Store Use Cases (cont.)

5. Pay Late Fees

Get customer ID.

Find all open rental agreements for customer in Database, *if* any.

Remind customer of all rented movies still outstanding, *if* any.

For each movie with a late fee, remind customer of movie and fee.

Tell customer grand total late fee still unpaid.

Collect money owed. / Mark as paid on rental agreement.

If no late fees due & no outstanding videos, mark agreement closed.

Write updated rental agreement(*s*) to Database.

Style Choices

A Use Case can be described:

- textually, as a high level description of business scenario;
- as user/system ***conversations*** (visualize w/ *UML Sequence Diagrams*)

For example, *Rent Videos* could be elaborated using Rebecca Wirfs-Brock's conversational style:

- **Actor**: Clerk enters customer ID into system.
- **System**: Show customer info, including outstanding late fees.
- **Actor**: For each video to rent, scan the barcode.
- **System**: Display accumulated rental fees, video titles, etc...
- **Actor**: Record customer payment.
- **System**: Generate receipt.

The Requirements Gathering Process

- Start with important business processes, user goals, and an architectural vision.
- Who are the actors (user roles)?
- What are their responsibilities and goals?
- For every user goal, identify one use case.
- Do the simple “success” use cases first: the goal succeeds. Then model failure scenarios as either << extends >> or as alternate scenarios, whichever is simpler.
- Look for commonality to identify << include >> relationships.
- Annotate other useful information in a textual document: such as the steps performed by the system, UI screen detail, business rules that must be invoked/enforced, etc.
- Focus on the value being delivered to users.

Requirements Gathering (cont.)

- Model the domain (conceptual, analysis) classes in conjunction with use case elaboration, but keep them separate.
- Only describe the internal behavior of the system to a level of detail that the actor would know about. Remember, we're doing requirements analysis, not design; use other kinds of UML models.
- Beware of using use cases as "functional decomposition" in disguise! Resist the urge to describe the detailed implementation of a scenario; save that for other UML diagrams.
- Good Use Cases make it easy to create good test plans.
- The "Product Owner" along with the Project Manager should prioritize the Use Cases (User Stories) when planning development iterations.
- Users stories should be independent from each other, small, and testable.