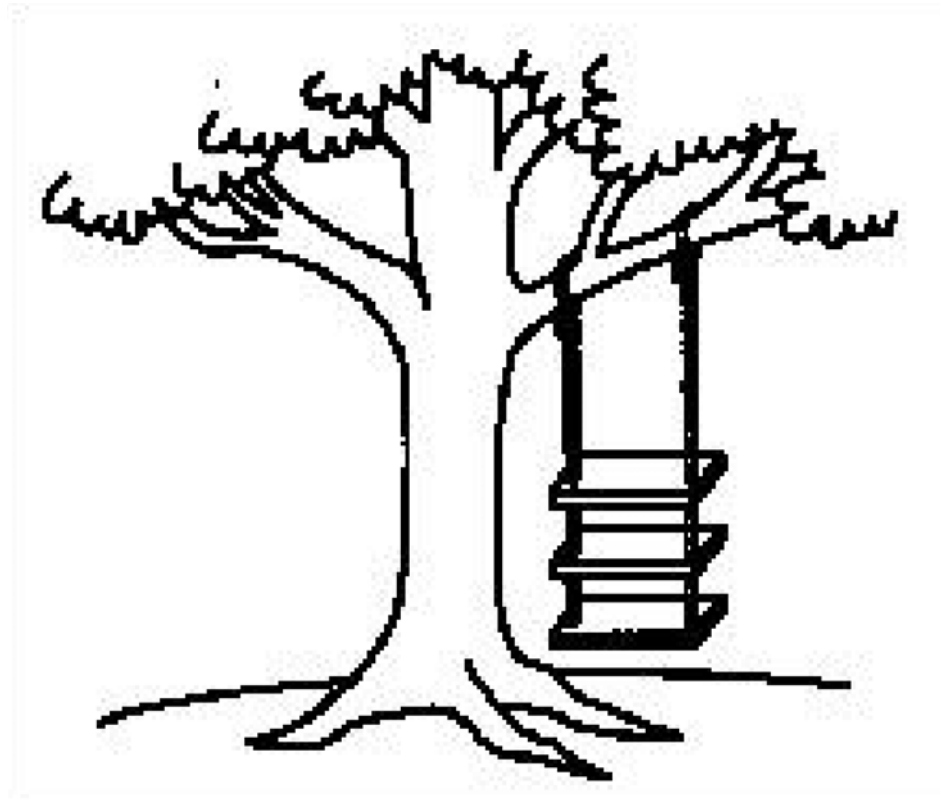


Software Requirement Specification

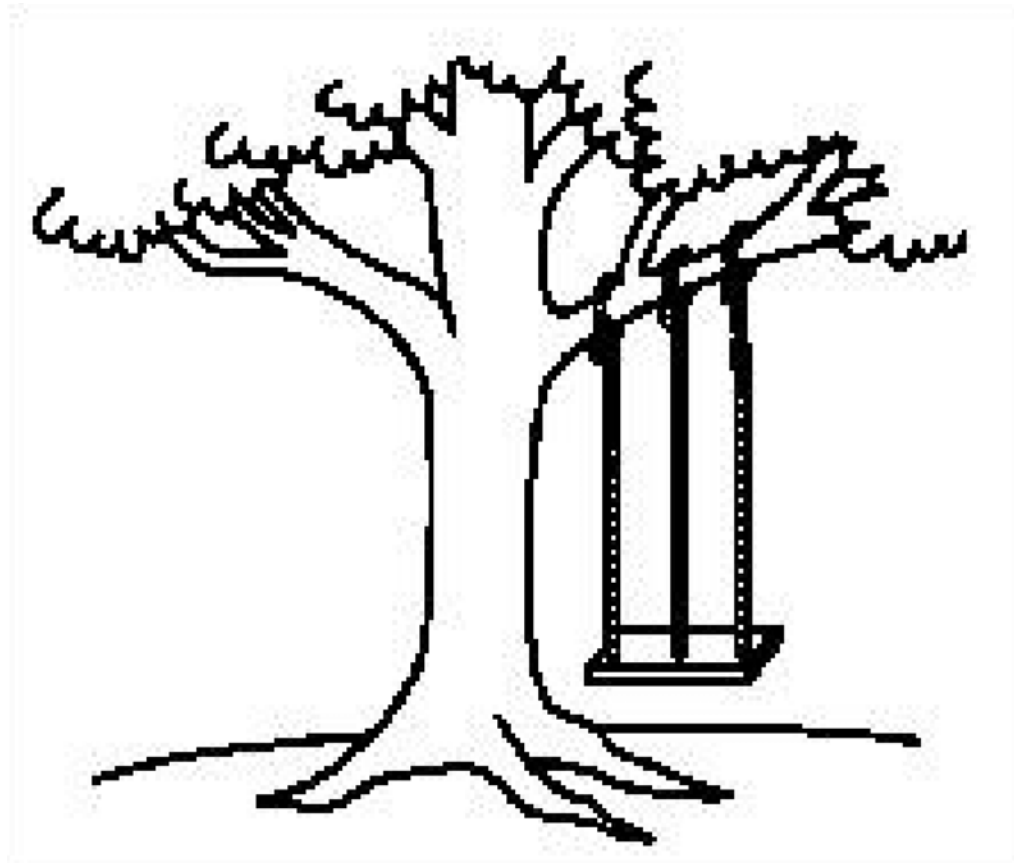
Requirements Analysis ₁

As Marketing requested it.



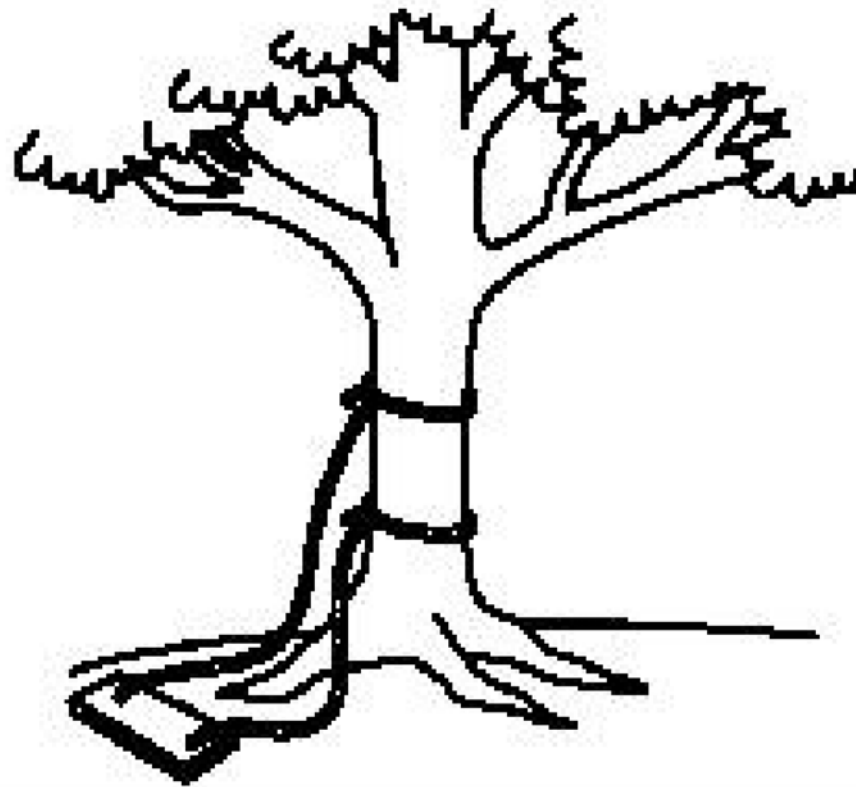
Requirements Analysis ₂

As Sales ordered it.



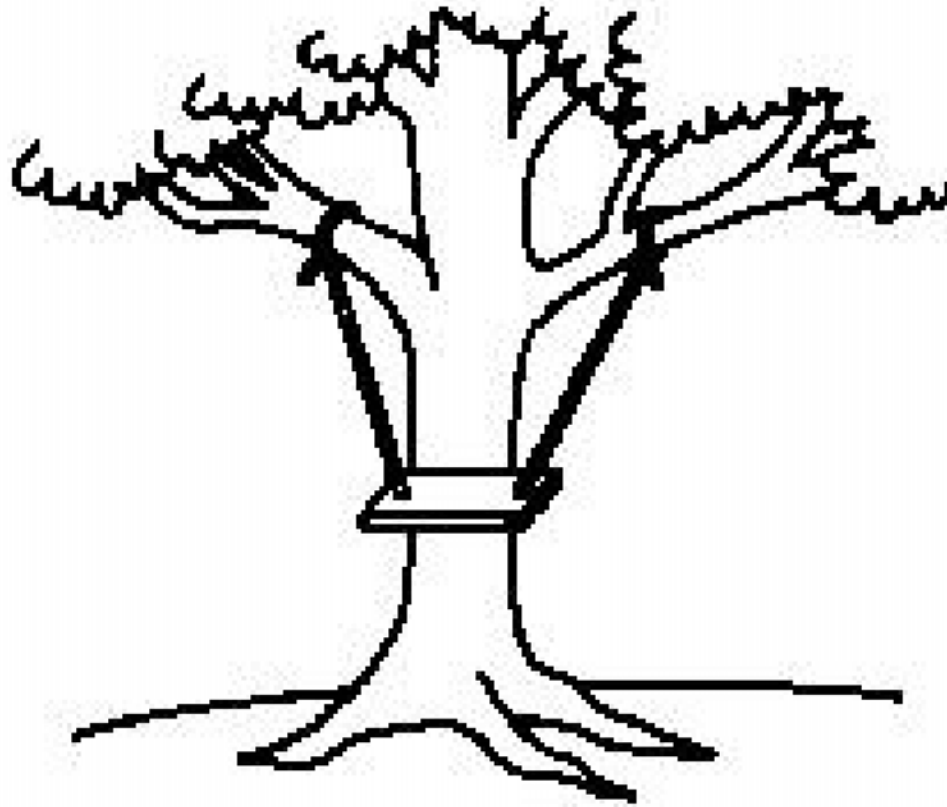
Requirements Analysis ₃

As Engineering designed it.



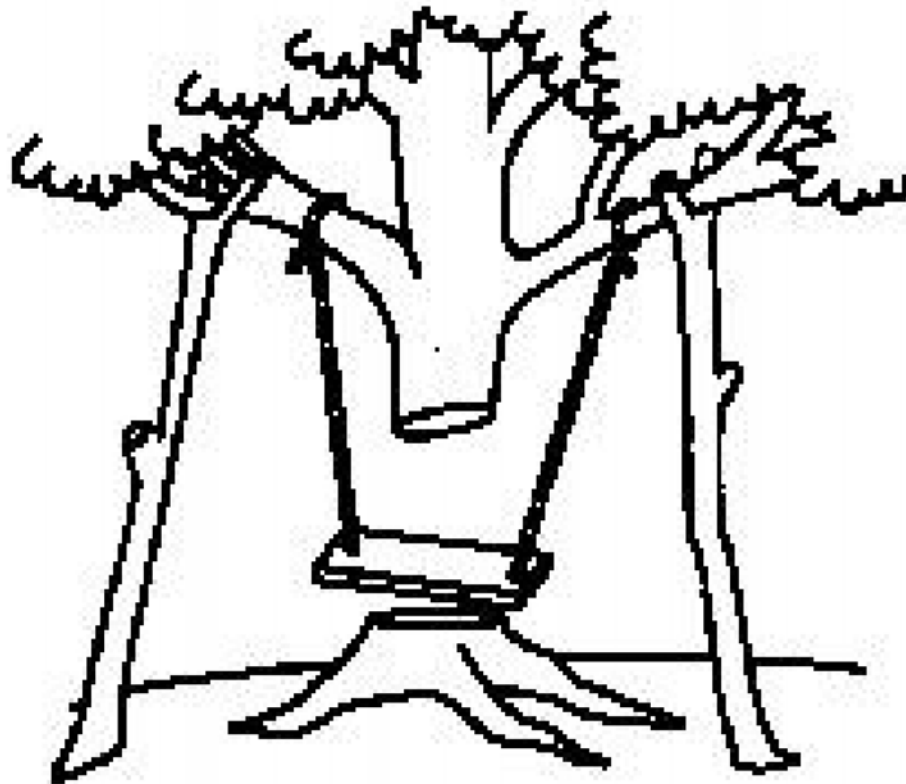
Requirements Analysis ₄

As Production manufactured it.



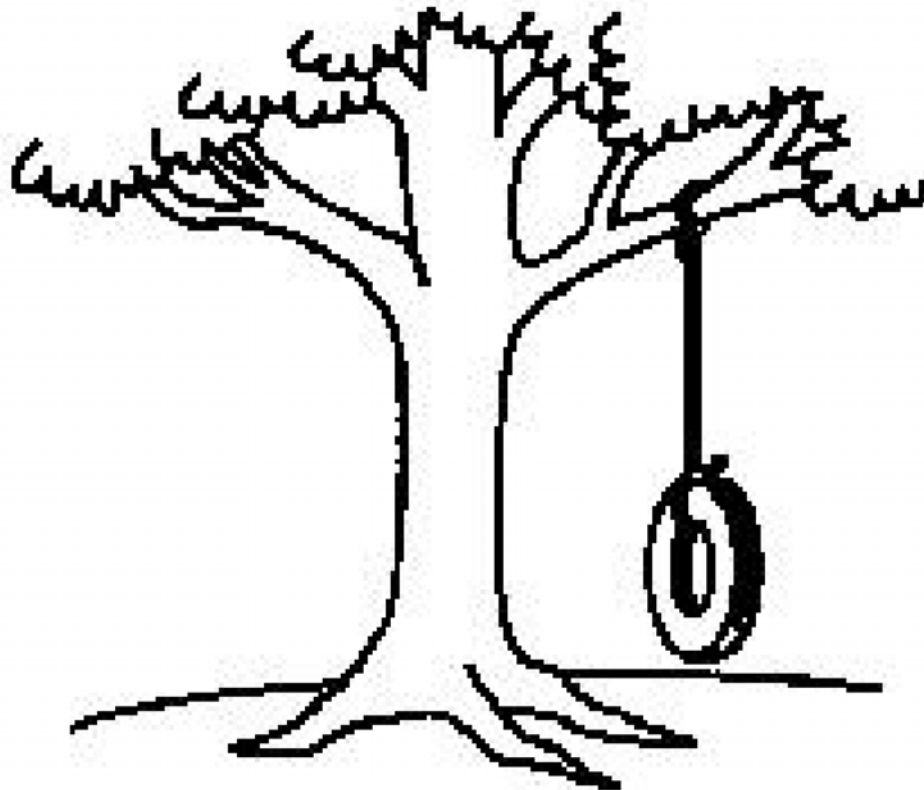
Requirements Analysis ₅

As Maintenance installed it.



Requirements Analysis ₆

What the customer wanted.

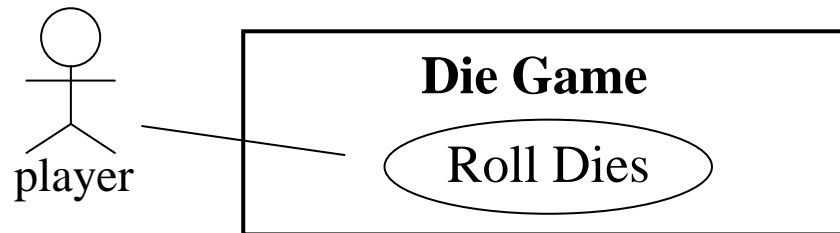


A Short Example ₁

□ *Define Use Cases*

○ *Play a Dice Game* use case:

- ◆ Player requests to roll the dice. System presents results: If the dice face value totals seven, player wins; otherwise, player loses.



Define use cases

Define domain
model

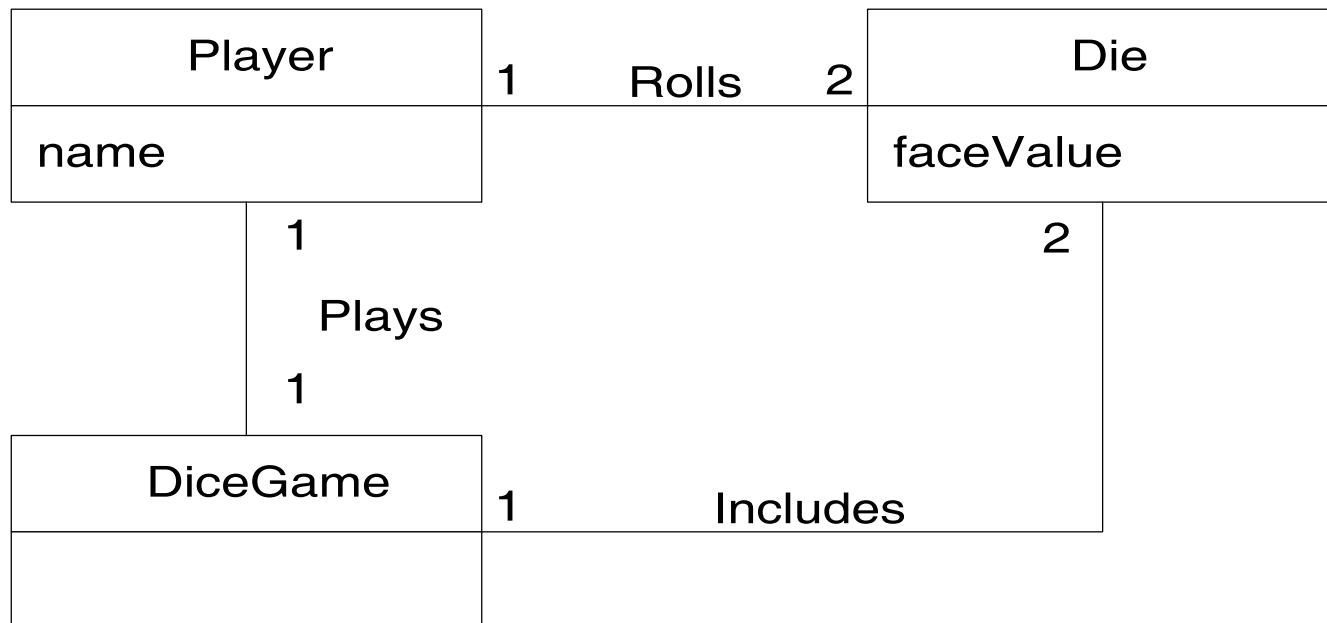
Define interaction
diagrams

Define design
class diagrams

A Short Example ₂

□ *Define a Domain Model*

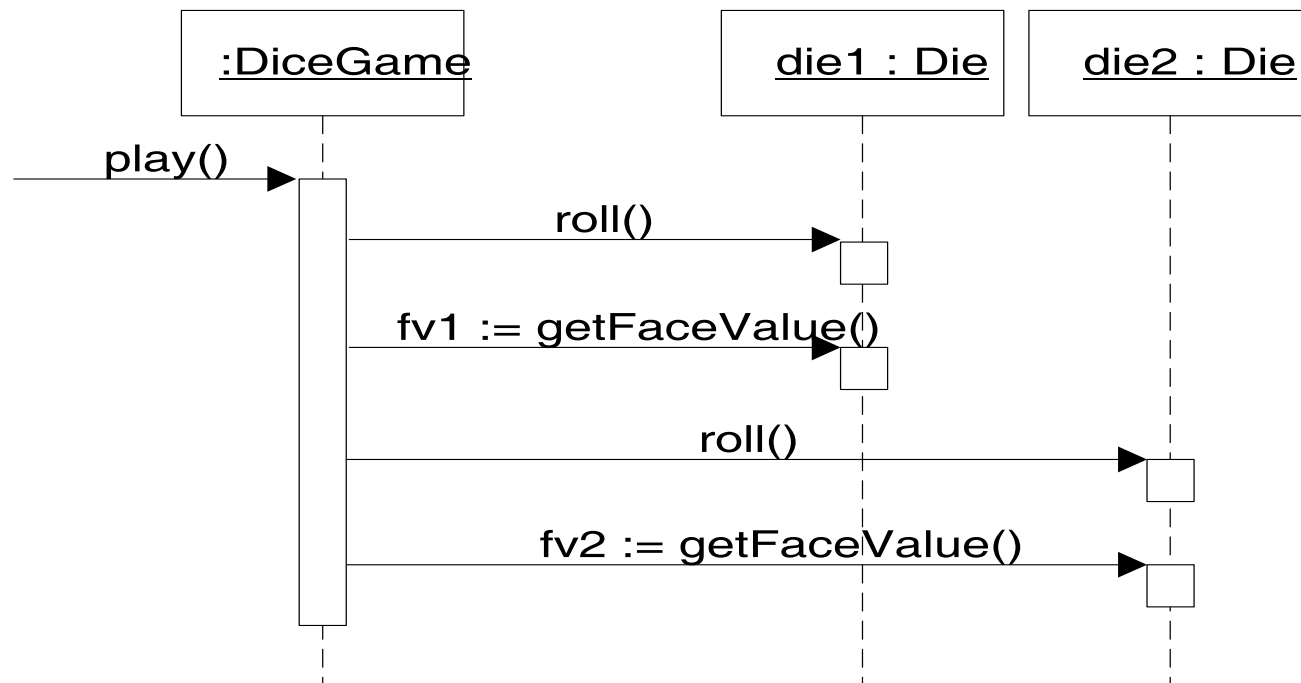
- creating a description of the domain from the perspective of objects. There is an identification of the concepts, attributes, and associations that are considered noteworthy.



A Short Example ₃

□ *Assign Object Responsibilities and Draw Interaction Diagrams*

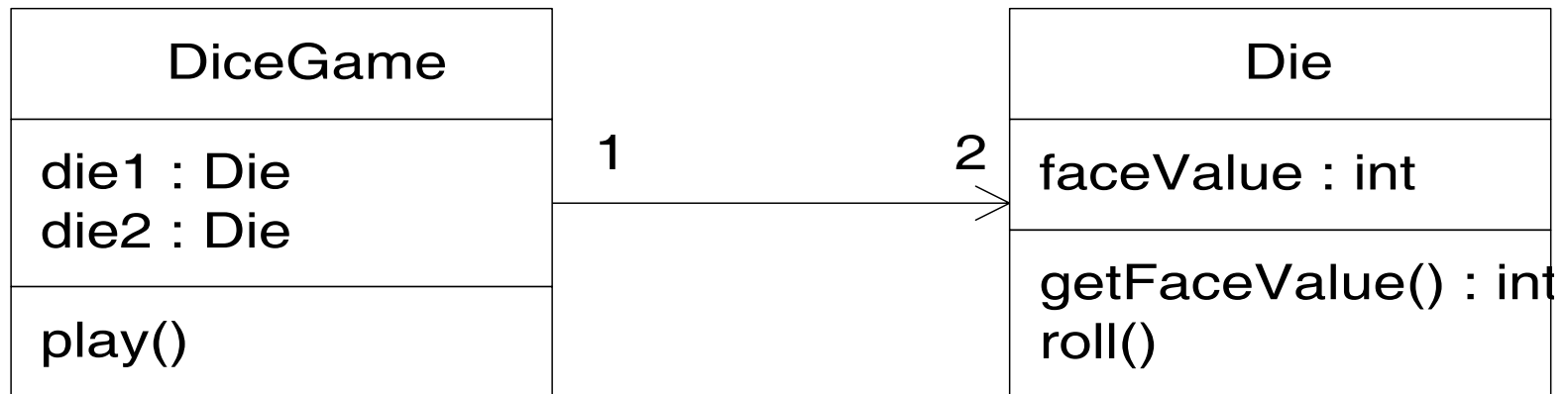
- to illustrate these collaborations is the **sequence diagram**. It shows the flow of messages between software objects, and the invocation of methods.



A Short Example 4

❑ *Define Design Class Diagrams*

- a *static* view of the class definitions is usefully shown with a **design class diagram**. This illustrates the attributes and methods of the classes.



Case Study

Problem Description 1

- ❑ The Point-of-Sale terminal is a computerized system used to record sales and handle payments; it is typically used in a retail store. It includes hardware components such as a computer and bar code scanner, and software to run the system.
- ❑ It interfaces to various service applications, such as a third-party tax calculator and inventory control. These systems must be relatively fault-tolerant; that is, even if remote services are temporarily unavailable (such as the inventory system), they must still be capable of capturing sales and handling at least cash payments (so that the business is not crippled).

Problem Description 2

- ❑ A POS system increasingly must support multiple and varied client-side terminals and interfaces. These include a thin-client Web browser terminal, a regular personal computer with something like a Java Swing graphical user interface, touch screen input, wireless PDAs, and so forth.
- ❑ Furthermore, we are creating a commercial POS system that we will sell to different clients with disparate needs in terms of business rule processing. Each client will desire a unique set of logic to execute at certain predictable points in scenarios of using the system, such as when a new sale is initiated or when a new line item is added.

Problem Description 3

- ❑ Therefore, we will need a mechanism to provide this flexibility and customization. Using an iterative development strategy, we are going to proceed through requirements, object-oriented analysis, design, and implementation.

Requirement 1

□ Requirements artifacts

- Use case Model
- Supplementary Specification (Non-functional requirements) -Reference
- Glossary (Data dictionary)
- System Sequence Diagram

Requirement 2

- ❑ Requirements are categorized according to the FURPS+ model.
 - Functional: features, capability, security
 - Usability: human factors, help, documentation
 - Reliability: frequency of failure, recoverability, predictability
 - Performance: response times, throughput, accuracy, availability, resource usage.
 - Supportability: adaptability, maintainability, internationalization, configurability

Requirement 3

- ❑ The “+” in FURPS+ indicates sub-factors
 - Implementation: resource limitation, languages and tools, hardware, ..
 - Interface: constraint imposed by interfacing with external systems
 - Operations: system management in its operational setting
 - Packaging: a physical box

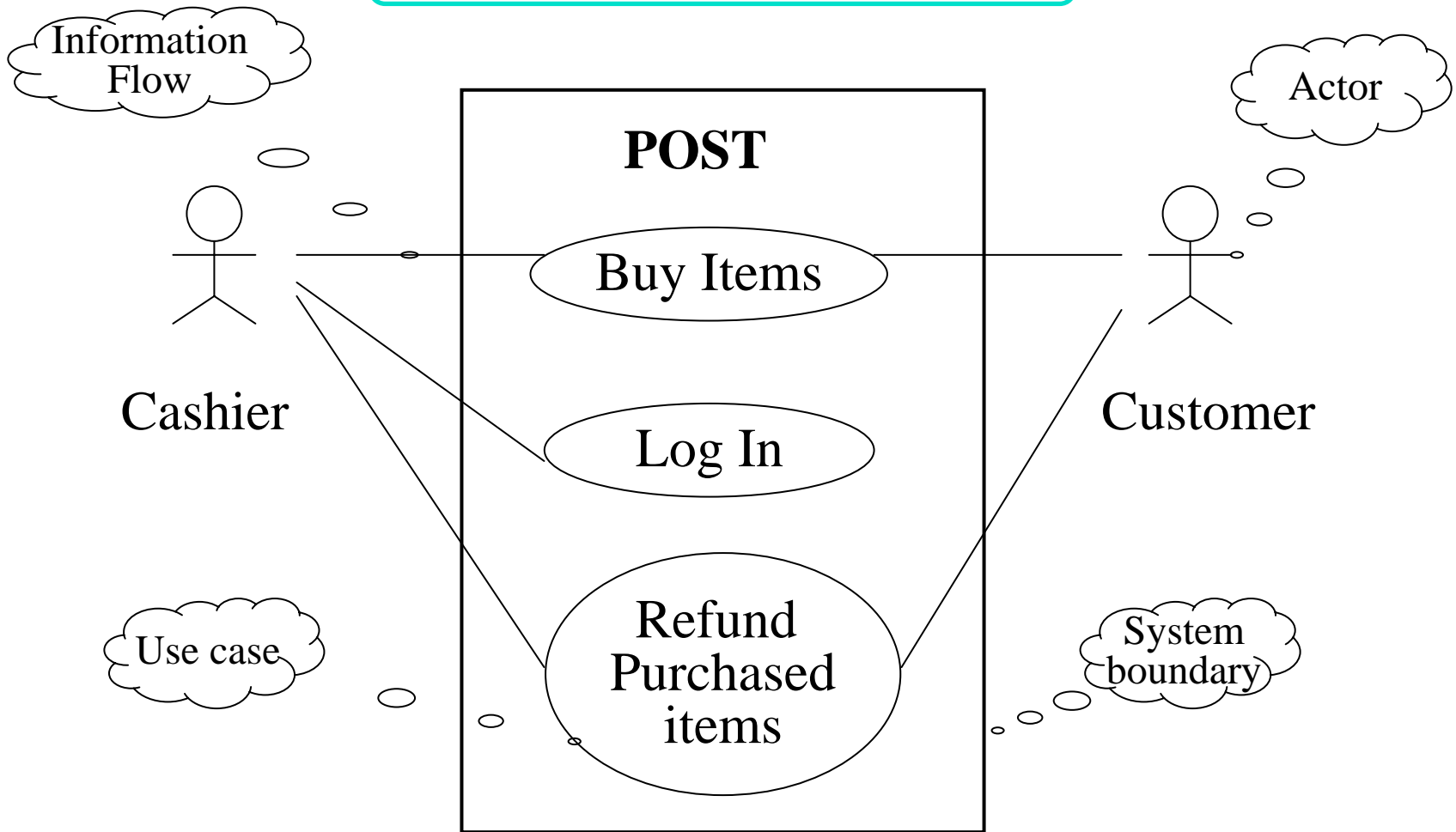
Use Case Model

Use Case Model

□ Use case model

- Be the set of all written use cases; it is a model of the system's functionality and environment.
- be not the only requirement artifact in the UP. There are also the Supplementary Specification, Glossary, Vision, and Business Rules.
- may optionally include a UML use case diagram to show the names of use cases and actors, and their relationships. This gives a nice context diagram of a system and its environment.

Use case Diagrams



Actors

- ❑ Actor: external entity interacts (behavior) with system, such as a person (identified by role), computer system, or organization; for example, a cashier.
- ❑ Three kind of Actors
 - Primary actor has user goals fulfilled through using services. (e.g., the cashier). Find user goals to drive the use cases.
 - Supporting actor provides a service (e.g., the automated payment authorization service is an example). Often a computer system, but could be an organization or person. The purpose is to 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., a government tax agency).

Use Case ₁

□ Use case

- is a collection of related success and failure scenarios that describe an actor using a system to support a goal.
- be text documents, not diagrams
- Use case modeling is primarily an act of writing text, not drawing diagrams.
- There is nothing object-oriented about use cases;
- Use cases are a key requirements input to classic OOA/D.
- be functional or behavioral requirements that indicate what the system will do. In terms of the FURPS+ requirements types, they emphasize the "F", but can also be used for other types.

Use Case 2

❑ The usage of use case

- Decide and describe the functional requirements of the system
- Bring agreement between the customer and software developer
- Give a clear and consistent description of what the system should do.
- Provide a basis for performing tests that verify the system delivers the functionality stated.
- Trace the functional requirements into actual classes and operations in the system.

Scenarios ₁

□ Scenario

- be a specific sequence of actions and interactions between actors and the system; it is also called a use case instance.
- It is one particular story of using a system, or one path through the use case.
- for example, the scenario of successfully purchasing items with cash, or the scenario of failing to purchase items because of a credit payment denial.

I would like a book of stamps, please.

OK. Will that be all?

Yes.

That will be \$7.80.

Here is \$10.

Thanks. Here are your stamps and your change.

Scenarios ₂

❑ Scenario

- A use case represents a collection of scenarios: primary, plus zero or more alternates.
- The primary scenario corresponds to the main system interactions, usually the ‘success’ scenario.
- Alternate scenarios correspond to less frequent interactions and exceptions.

Three Common Use Case Formats

- ❑ Brief (high level)
 - Terse one-paragraph summary, usually of the main success scenario.
 - During early requirements analysis, to get a quick sense of subject and scope. May take only a few minutes to create.
- ❑ Fully dressed
 - All steps and variations are written in detail, and there are supporting sections, such as preconditions and success guarantees.
 - After many use cases have been identified and written in a brief format, then during the first requirements workshop a few (such as 10%) of the architecturally significant and high-value use cases are written in detail.

Brief Use Case Example 1

- ❑ Use case: Buy Items
 - Actors: Customer, Cashier
 - Type: Primary
 - Description: A customer arrives at checkout with items to purchase. The Cashier records the purchase items and collects payment, On completion, the Customer leaves with the items.

Use Case Scenario: Buy Items ₁

1. When a a Customer arrives at the POS Terminal checkout with items to purchase.
2. The Cashier records each items. If there is more than one of an item, the Cashier can enter the quantity as well.
3. The system determines the item price and adds the item information to running sales transaction. The description and price of the current item are presented.
4. On completion of item entry, the Cashier indicates to the POS Terminal that item entry is complete.
5. The system calculates and presents the sale total.
6. The Cashier tells the Customer the total.

Use Case Scenario: Buy Items 2

7. Customer choose payment type: If cash payment, see section *Pay by Cash*. If credit payment, see section *Pay by Credit*.
8. The system logs the complete sale.
9. The system updates inventory.
10. The system generates a receipt.
11. The Cashier gives the receipt to the Customer.
12. The Customer leaves with the items purchases.

Variation

- 2.1. If invalid item identifier entered, indicate error.
- 7.1. If Customer could not pay, cancel sales transaction.

Glossary

Glossary ₁

- ❑ Guideline: Start the Glossary early. It will become a repository of detailed information.
- ❑ Glossary, a document that records data/metadata. During inception the glossary should be a simple document of terms and descriptions. During elaboration, it may expand into a data dictionary.
 - Term attributes could include:
 - aliases
 - description
 - format (type, length, unit)
 - relationships to other elements
 - range of values
 - validation rules

POS Glossary ₁

❑ Revision History

Version	Date	Description	Author
Inception draft	Jan 10, 2031	First draft. To be refined primarily during elaboration.	Craig Larman

❑ Definition

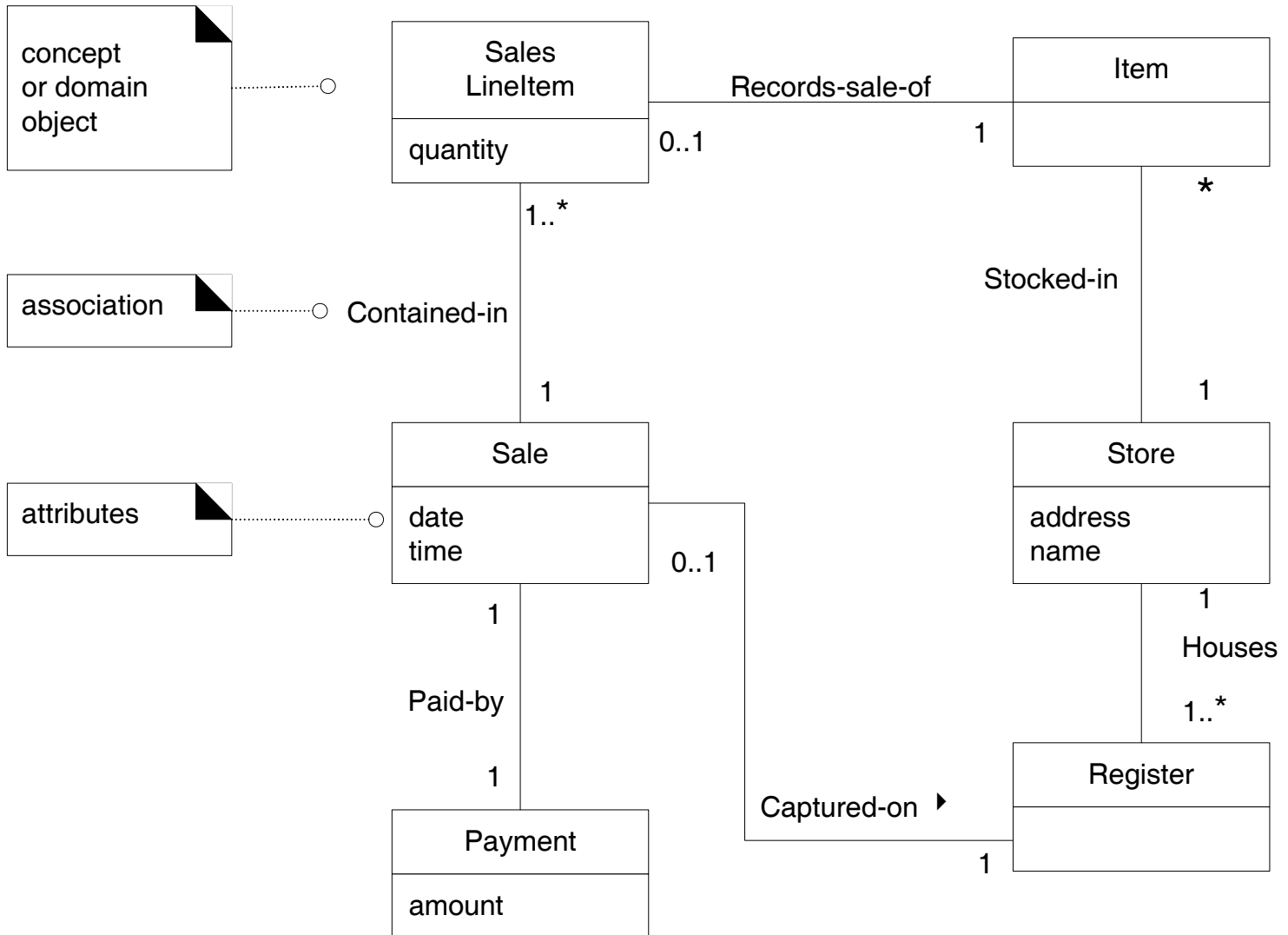
Term	Definition and Information	Format	Validation Rules	Aliases
item	A product or service for sale			
payment authorization	Validation by an external payment authorization service that they will make or guarantee the payment to the seller.			

POS Glossary ₂

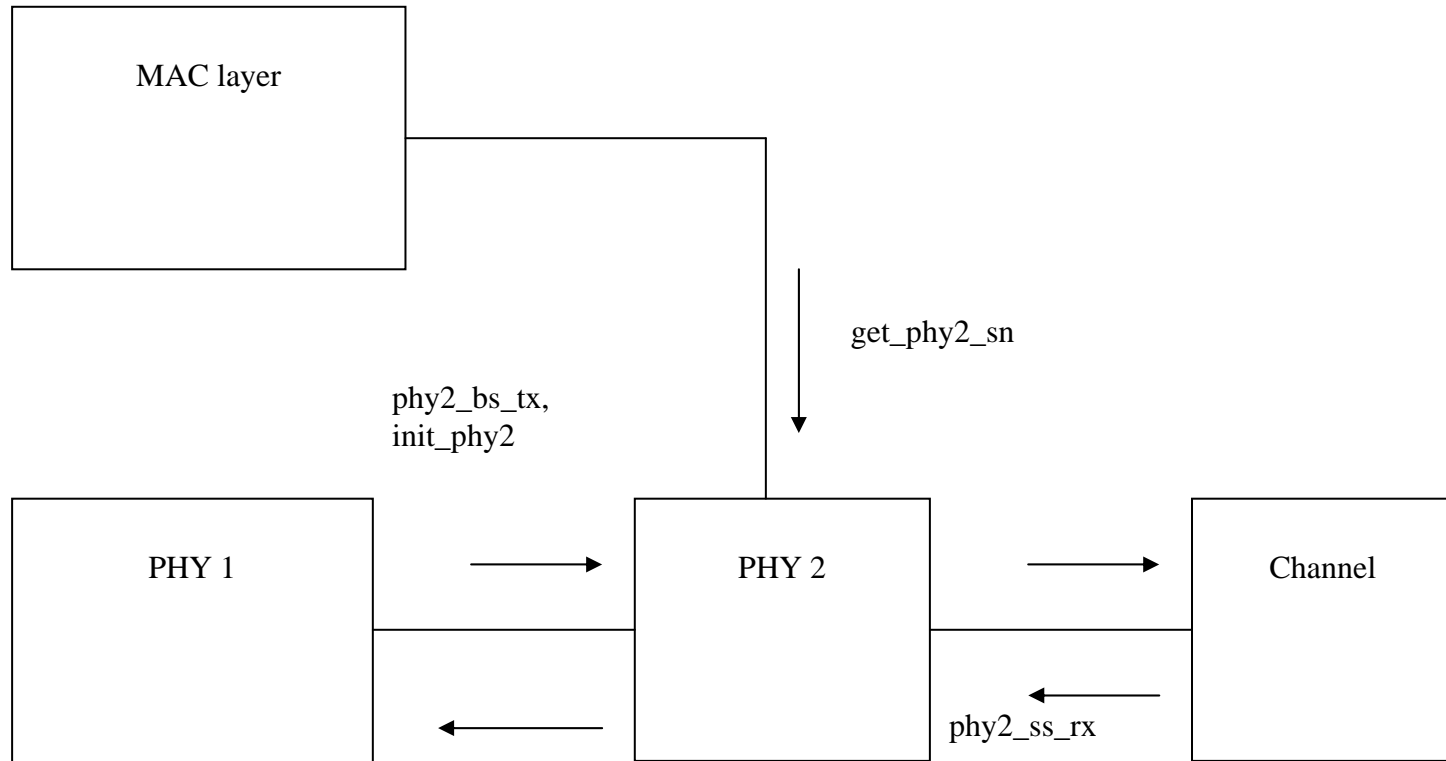
Term	Definition and Information	Format	Validation Rules	Aliases
payment authorization request	A composite of elements electronically sent to an authorization service, usually as a char array. Elements include: store ID, customer account number, amount, and timestamp.			
UPC	Numeric code that identifies a product. Usually symbolized with a bar code placed on products. See www.uc-council.org for details of format and validation.	12-digit code of several subparts.	Digit 12 is a check digit.	Universal Product Code
...	...			

Domain Model or Context Diagram

POS Domain Model

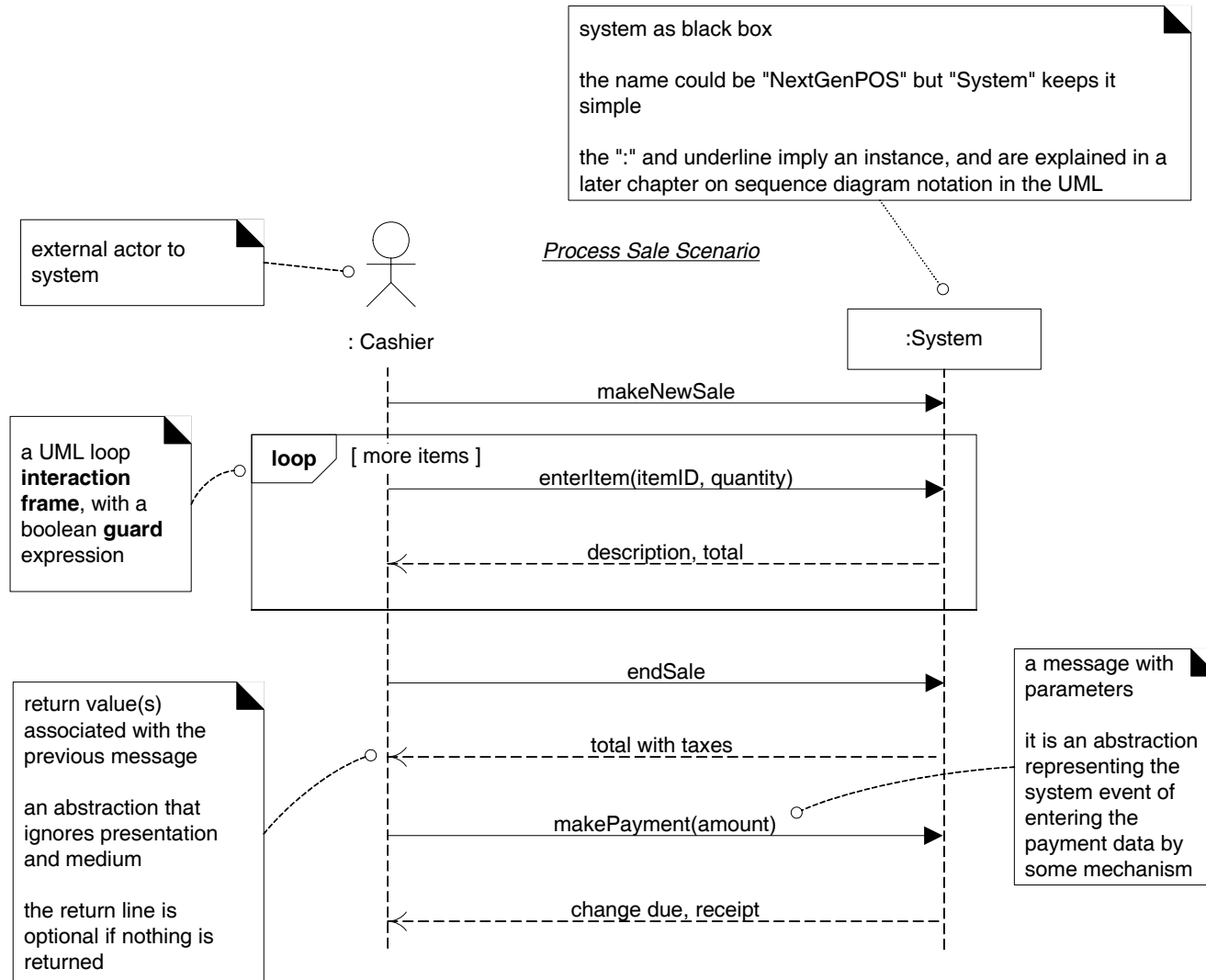


Context Diagram



System Sequence Diagrams

POS SSD: a Process Sale Scenario

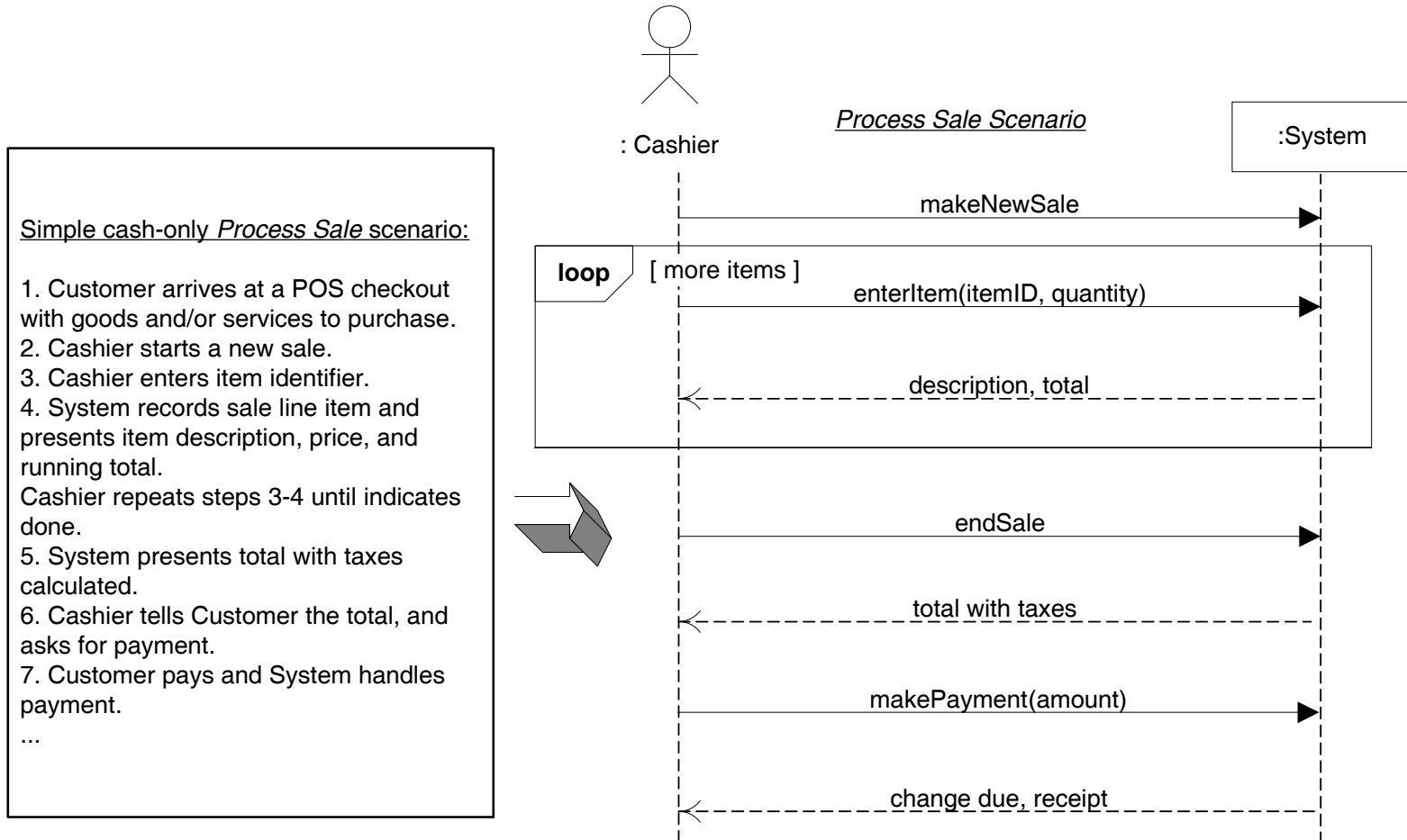


System Sequence Diagram ₁

- ❑ System sequence diagram
 - a picture that shows, for one particular scenario of a use case, the events that external actors generate, their order, and inter-system events.
 - All systems are treated as a black box; the emphasis of the diagram is events that cross the system boundary from actors to systems.
- ❑ Guideline: Draw an SSD for a main success scenario of each use case, and frequent or complex alternative scenarios.

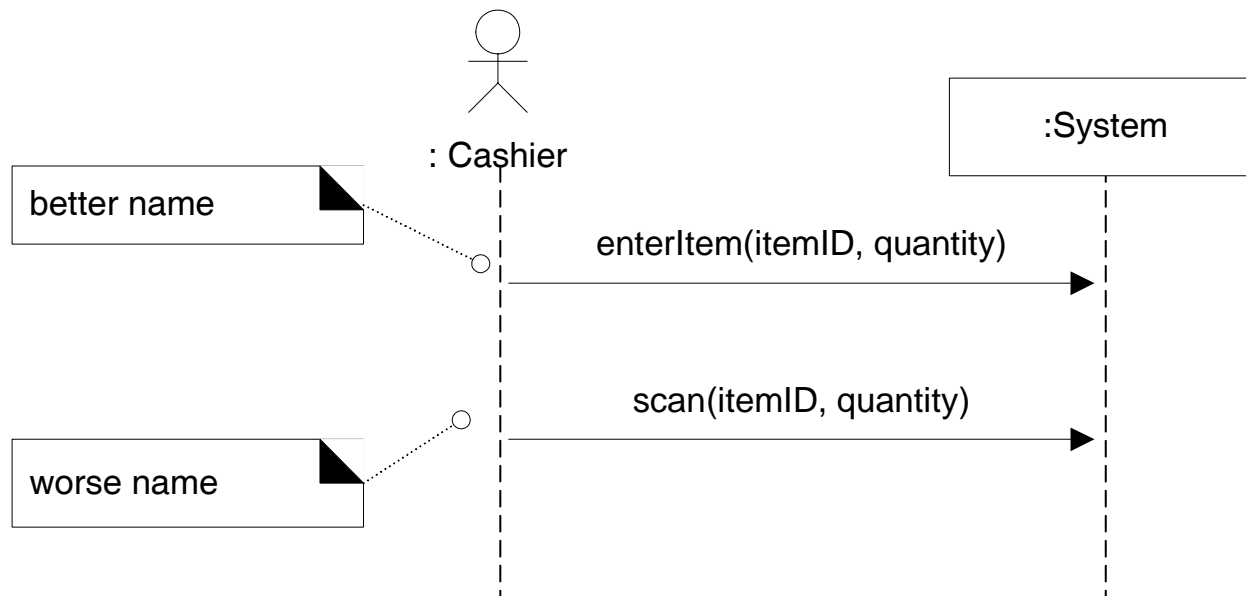
System Sequence Diagram 2

- ❑ SSDs are derived from use cases; they show one scenario.

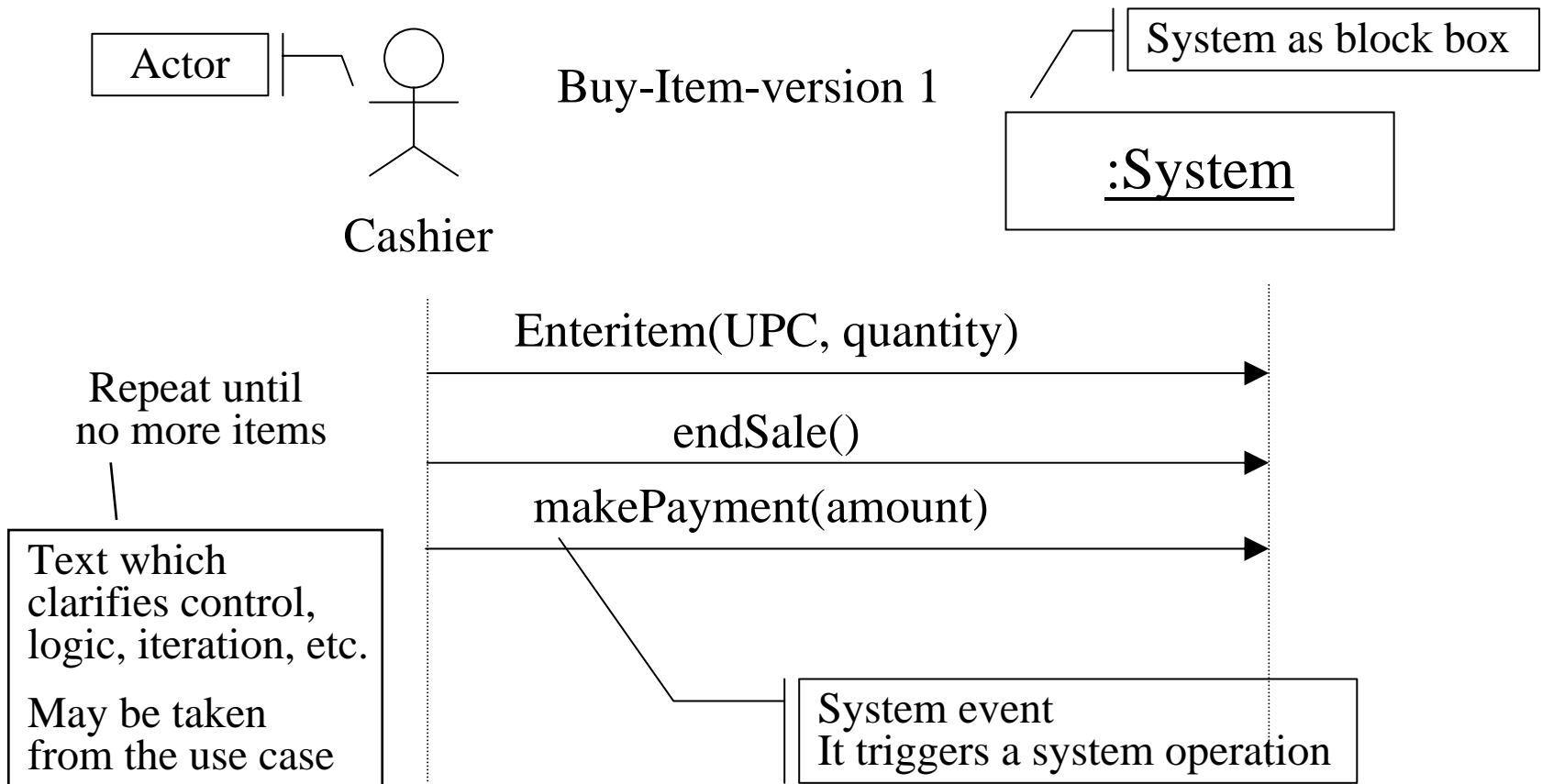


System Sequence Diagram ₃

- ❑ System events should be expressed at the abstract level of intention rather than in terms of the physical input device.
 - "enterItem" is better than "scan" (laser scan) because it captures the abstract intent of the operation.
 - design choices about what interface is used to capture the system event (laser scanner, keyboard, voice input ..)



System Sequence Diagram - *Buy-Item*



Operation Contracts

POS Operation Contract

❑ Contract CO2: enterItem

- Operation: enterItem(itemID: ItemID, quantity: integer)
- Cross References: Use Cases: Process Sale
- Preconditions: There is a sale underway.
- Postconditions:
 - ◆ A SalesLineItem instance sli was created (instance creation).
 - ◆ sli was associated with the current Sale (association formed).
 - ◆ sli.quantity became quantity (attribute modification).
 - ◆ sli was associated with a ProductDescription, based on itemID match (association formed).

Sections of a Contract

- ❑ Operation: Name of operation, and parameters
- ❑ Cross References: Use cases this operation can occur within
- ❑ Preconditions: Noteworthy assumptions about the state of the system or objects in the Domain Model before execution of the operation.
- ❑ Postconditions: The state of objects in the Domain Model after completion of the operation.

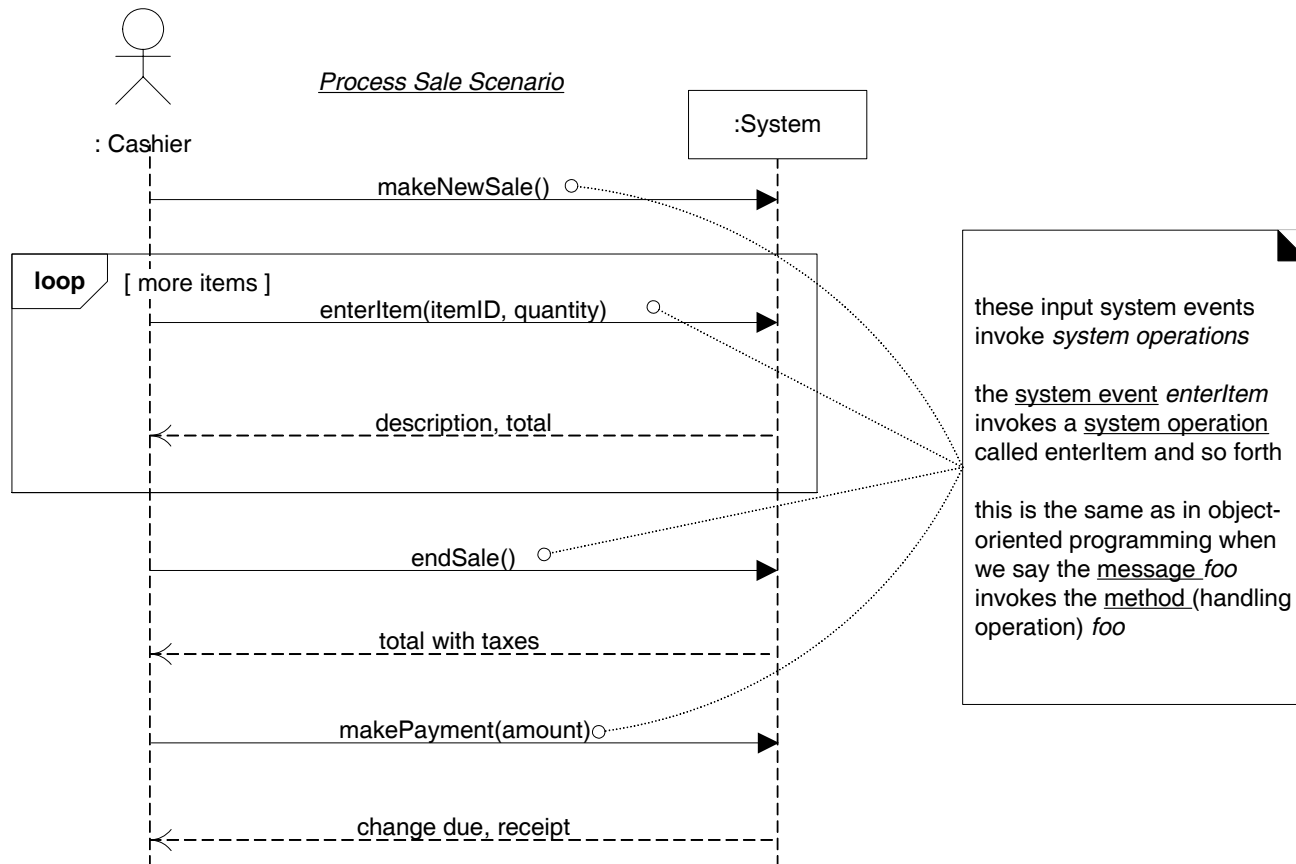
System Operations ₁

□ System operations

- the system as a black box component offers in its public interface.
- can be identified while sketching SSDs
- SSDs show system events or I/O messages relative to the system.

System Operations 2

- SSD. System operations handle input system events



Postconditions ₁

❑ Postconditions

- describe changes in the state of objects in the domain model.
- Domain model state changes include instances created, associations formed or broken, and attributes changed.
- Postconditions are not actions to be performed.

❑ Postconditions fall into these categories:

- Instance creation and deletion.
- Attribute change of value.
- Associations (UML links) formed and broken.

❑ Postconditions Related to the Domain Model

- What instances can be created; What associations can be formed in the Domain Model.

❑ Motivation: Why Postconditions?

- Postconditions support fine-grained detail and precision in declaring what the outcome of the operation must be.

Postconditions ₂

❑ Guideline: To Write a Postcondition

- to emphasize state changes that arose from an operation, not an action to happen.
- (better) A SalesLineItem was created.
- (worse) Create a SalesLineItem, or, A SalesLineItem is created.

Example: enterItem Postconditions

- ❑ Postconditions of the enterItem system operation.
 - Instance Creation and Deletion
 - ◆ After the itemID and quantity of an item have been entered, what new object should have been created? A SalesLineItem. Thus:
 - ◆ A SalesLineItem instance sli was created (instance creation).
 - Attribute Modification
 - ◆ After the itemID and quantity of an item have been entered by the cashier, what attributes of new or existing objects should have been modified? The quantity of the SalesLineItem should have become equal to the quantity parameter. Thus:
 - ◆ sli.quantity became quantity (attribute modification).
 - Associations Formed and Broken
 - ◆ After the itemID and quantity of an item have been entered by the cashier, what associations between new or existing objects should have been formed or broken?
 - ◆ The new SalesLineItem should have been related to its Sale, and related to its ProductDescription. Thus:
 - ◆ sli was associated with the current Sale (association formed).
 - ◆ sli was associated with a ProductDescription, based on itemID match (association formed).

Example: NextGen POS Contracts ₁

- ❑ **System Operations of the Process Sale Use Case**
 - **Contract CO1: makeNewSale**
 - Operation: makeNewSale()
 - Cross References: Use Cases: Process Sale
 - Preconditions: none
 - Postconditions:
 - ◆ A Sale instance s was created (instance creation).
 - ◆ s was associated with a Register (association formed).
 - ◆ Attributes of s were initialized.
- ❑ Keep it as light as possible, and avoid all artifacts unless they really add value.
- ❑ **Contract CO2: enterItem**
 - Operation: enterItem(itemID: ItemID, quantity: integer)
 - Cross References: Use Cases: Process Sale
 - Preconditions: There is a sale underway.
 - Postconditions:
 - ◆ A SalesLineItem instance sli was created (instance creation).
 - ◆ sli was associated with the current Sale (association formed).
 - ◆ sli.quantity became quantity (attribute modification).
 - ◆ sli was associated with a ProductDescription, based on itemID match (association formed).

Example: NextGen POS Contracts ₂

- ❑ Contract CO3: endSale
 - Operation: endSale()
 - Cross References: Use Cases: Process Sale
 - Preconditions: There is a sale underway.
 - Postconditions:
 - ◆ Sale.isComplete became true (attribute modification).
- ❑ Contract CO4: makePayment
 - Operation: makePayment(amount: Money)
 - Cross References: Use Cases: Process Sale
 - Preconditions: There is a sale underway.
 - Postconditions:
 - ◆ A Payment instance p was created (instance creation).
 - ◆ p.amountTendered became amount (attribute modification).
 - ◆ p was associated with the current Sale (association formed).
 - ◆ The current Sale was associated with the Store (association formed); (to add it to the historical log of completed sales)

Reference Supplementary Specification

Supplementary Specification ₁

- ❑ Supplementary Specification
 - captures other requirements, information, and constraints not easily captured in the use cases or Glossary
 - including system-wide "URPS+" (usability, reliability, performance, supportability, and more) quality attributes or non-functional requirements.
- ❑ non-functional requirements specific to a use case can be first briefly written within in the Special Requirements section

Supplementary Specification 2

□ Elements of the Supplementary Specification

○FURPS+

- ◆Some functions or features don't fit in a use case format. we did think of the functionality in terms of features, such as "add EJB Entity Bean 1.0 support."

○reports

○Hardware/software constraints (operating and networking systems, ...)

○development constraints (for example, process or development tools)

○other design and implementation constraints

○internationalization concerns (units, languages)

Supplementary Specification ₃

□ Elements of the Supplementary Specification

- documentation (user, installation, administration) and help
- licensing and other legal concerns
- packaging
- packaging
- standards (technical, safety, quality)
- physical environment concerns (for example, heat or vibration)
- operational concerns (for example, how do errors get handled, or how often should backups be done?)
- application-specific domain rules
- information in domains of interest (for example, what is the entire cycle of credit payment handling?)

POS Supplementary Specification ₁

❑ Revision History

Version	Date	Description	Author
Inception draft	Jan 10, 2031	First draft. To be refined primarily during elaboration.	Craig Larman

- ❑ Introduction: This document is the repository of all NextGen POS requirements not captured in the use cases.
- ❑ Functionality (Functionality common across many use cases)
- ❑ Logging and Error Handling
 - Log all errors to persistent storage.

POS Supplementary Specification ₂

- ❑ Pluggable Rules: At various scenario points of several use cases (to be defined) support the ability to customize the functionality of the system with a set of arbitrary rules that execute at that point or event.
- ❑ Security: All usage requires user authentication.
- ❑ Usability
- ❑ Human Factors: The customer will be able to see a large-monitor display of the POS. Therefore:
 - Text should be easily visible from 1 meter.
 - Avoid colors associated with common forms of color blindness.
 - Speed, ease, and error-free processing are paramount in sales processing, as the buyer wishes to leave quickly, or they perceive the purchasing experience (and seller) as less positive.
 - The cashier is often looking at the customer or items, not the computer display. Therefore, signals and warnings should be conveyed with sound rather than only via graphics.

POS Supplementary Specification ₃

- ❑ Reliability
- ❑ Recoverability: If there is failure to use external services (payment authorizer, accounting system, ...) try to solve with a local solution (e.g., store and forward) in order to still complete a sale. Much more analysis is needed here...
- ❑ Performance: As mentioned under human factors, buyers want to complete sales processing very quickly. One bottleneck is external payment authorization. Our goal: authorization in less than 1 minute, 90% of the time.
- ❑ Supportability
- ❑ Adaptability: Different customers of the NextGen POS have unique business rule and processing needs while processing a sale. Therefore, at several defined points in the scenario (for example, when a new sale is initiated, when a new line item is added) pluggable business rule will be enabled.

POS Supplementary Specification 4

❑ Configurability

- Different customers desire varying network configurations for their POS systems, such as thick versus thin clients, two-tier versus N-tier physical layers, and so forth. In addition, they desire the ability to modify these configurations, to reflect their changing business and performance needs. Therefore, the system will be somewhat configurable to reflect these needs. Much more analysis is needed in this area to discover the areas and degree of flexibility, and the effort to achieve it.

❑ Implementation Constraints

- NextGen leadership insists on a Java technologies solution, predicting this will improve long-term porting and supportability, in addition to ease of development.

❑ Purchased Components

- Tax calculator. Must support pluggable calculators for different countries.

POS Supplementary Specification 5

- ❑ Free Open Source Components
 - In general, we recommend maximizing the use of free Java technology open source components on this project.
 - Although it is premature to definitively design and choose components, we suggest the following as likely candidates:
 - ◆ JLog logging framework
 - ◆ ...
- ❑ Interfaces
- ❑ Noteworthy Hardware and Interfaces
 - Touch screen monitor (this is perceived by operating systems as a regular monitor, and the touch gestures as mouse events)
 - Barcode laser scanner (these normally attach to a special keyboard, and the scanned input is perceived in software as keystrokes)
 - Receipt printer
 - Credit/debit card reader
 - Signature reader (but not in release 1)

POS Supplementary Specification 6

❑ Software Interfaces

- For most external collaborating systems (tax calculator, accounting, inventory, ...) we need to be able to plug in varying systems and thus varying interfaces.

❑ Application-Specific Domain (Business) Rules

- (See the separate Business Rules document for general rules.)

ID	Rule	Changeability	Source
RULE1	Purchaser discount rules. Examples: Employee 20% off. Preferred Customer 10% off. Senior 15% off.	High. Each retailer uses different rules.	Retailer policy.

POS Supplementary Specification 7

ID	Rule	Changeability	Source
RULE2	<p>Sale (transaction-level) discount rules. Applies to pre-tax total. Examples: 10% off if total greater than \$100 USD. 5% off each Monday. 10% off all sales from 10am to 3pm today. Tofu 50% off from 9am-10am today.</p>	<p>High. Each retailer uses different rules, and they may change daily or hourly.</p>	<p>Retailer policy.</p>

POS Supplementary Specification ⁸

ID	Rule	Changeability	Source
RULE 3	Product (line item level) discount rules. Examples: 10% off tractors this week. Buy 2 veggieburgers, get 1 free.	High. Each retailer uses different rules, and they may change daily or hourly.	Retailer policy.

Reference
Fully Use Case

Fully Use Case Template

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.

Fully Use Case Example 1

- ❑ 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: Wants purchase and fast service with minimal effort. Wants easily visible display of entered items and prices. Wants proof of purchase to support returns.
 - Company: Wants to accurately record transactions and satisfy customer interests. Wants to ensure that Payment Authorization Service payment receivables are recorded. Wants some fault tolerance to allow sales capture even if server components (e.g., remote credit validation) are unavailable. Wants automatic and fast update of accounting and inventory.

Fully Use Case Example 2

- Manager: Wants to be able to quickly perform override operations, and easily debug Cashier problems.
- Government Tax Agencies: Want to collect tax from every sale. May be multiple agencies, such as national, state, and county.
- Payment Authorization Service: Wants to receive digital authorization requests in the correct format and protocol. Wants to accurately account for their payables to the store.
- 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.

Fully Use Case Example 3

❑ 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.
- 5. Cashier repeats steps 3-4 until indicates done.
- 6. System presents total with taxes calculated.
- 7. Cashier tells Customer the total, and asks for payment.
- 8. Customer pays and System handles payment.
- 9. 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).
- 10. System presents receipt.
- 11. Customer leaves with receipt and goods (if any).

Fully Use Case Example 4

□ 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.

Fully Use Case Example 5

□ Extensions (or Alternative Flows)

- 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)

Fully Use Case Example 6

❑ Extensions (or Alternative Flows)

○ 3a. Invalid item ID (not found in system):

- ◆ 1. System signals error and rejects entry.
- ◆ 2. Cashier responds to the error:
 - 2a. There is a human-readable item ID (e.g., a numeric UPC):
 - ❖ 1. Cashier manually enters the item ID.
 - ❖ 2. System displays description and price.
 - ❖ 2a. Invalid item ID: System signals error. Cashier tries alternate method
 - 2b. There is no item ID, but there is a price on the tag:
 - ❖ 1. Cashier asks Manager to perform an override operation.
 - ❖ 2. Managers performs override.
 - ❖ 3. Cashier indicates manual price entry, enters price, and requests standard taxation for this amount (because there is no product information, the tax engine can't otherwise deduce how to tax it)
 - 2c. Cashier performs Find Product Help to obtain true item ID and price.
 - 2d. Otherwise, Cashier asks an employee for the true item ID or price, and does either manual ID or manual price entry (see above).

Fully Use Case Example 7

❑ Extensions (or Alternative Flows)

- 3b. There are multiple of same item category and tracking unique item identity not important (e.g., 5 packages of veggie-burgers):
 - ◆ 1. Cashier can enter item category identifier and the quantity.
- 3c. Item requires manual category and price entry (such as flowers or cards with a price on them):
 - ◆ 1. Cashier enters special manual category code, plus the price.
- 3-6a: Customer asks Cashier to remove (i.e., void) an item from the purchase: This is only legal if the item value is less than the void limit for Cashiers, otherwise a Manager override is needed.
 - ◆ 1. Cashier enters item identifier for removal from sale.
 - ◆ 2. System removes item and displays updated running total.
 - 2a. Item price exceeds void limit for Cashiers:
 - ❖ 1. System signals error, and suggests Manager override.
 - ❖ 2. Cashier requests Manager override, gets it, and repeats operation.

Fully Use Case Example 8

❑ Extensions (or Alternative Flows)

- 3-6b. Customer tells Cashier to cancel sale:
 - ◆ 1. Cashier cancels sale on System.
- 3-6c. Cashier suspends the sale:
 - ◆ 1. System records sale so that it is available for retrieval on any POS register.
 - ◆ 2. System presents a "suspend receipt" that includes the line items, and a sale ID used to retrieve and resume the sale.
- 4a. The system supplied item price is not wanted (e.g., Customer complained about something and is offered a lower price):
 - ◆ 1. Cashier requests approval from Manager.
 - ◆ 2. Manager performs override operation.
 - ◆ 3. Cashier enters manual override price.
 - ◆ 4. System presents new price.

Fully Use Case Example 9

□ Extensions (or Alternative Flows)

- 5a. System detects failure to communicate with external tax calculation system service:
 - ◆ 1. System restarts the service on the POS node, and continues.
 - 1a. System detects that the service does not restart.
 - ❖ 1. System signals error.
 - ❖ 2. Cashier may manually calculate and enter the tax, or cancel the sale.
- 5b. Customer says they are eligible for a discount (e.g., employee, preferred customer):
 - ◆ 1. Cashier signals discount request.
 - ◆ 2. Cashier enters Customer identification.
 - ◆ 3. System presents discount total, based on discount rules.

Fully Use Case Example ₁₀

□ Extensions (or Alternative Flows)

- 5c. Customer says they have credit in their account, to apply to the sale:
 - ◆ 1. Cashier signals credit request.
 - ◆ 2. Cashier enters Customer identification.
 - ◆ 3. Systems applies credit up to price=0, and reduces remaining credit.
- 6a. Customer says they intended to pay by cash but don't have enough cash:
 - ◆ 1. Cashier asks for alternate payment method.
 - 1a. Customer tells Cashier to cancel sale. Cashier cancels sale on System.
- 7a. Paying by cash:
 - ◆ 1. Cashier enters the cash amount tendered.
 - ◆ 2. System presents the balance due, and releases the cash drawer.
 - ◆ 3. Cashier deposits cash tendered and returns balance in cash to Customer.
 - ◆ 4. System records the cash payment.

Fully Use Case Example ₁₁

□ Extensions (or Alternative Flows)

○ 7b. Paying by credit:

- ◆ 1. Customer enters their credit account information.
- ◆ 2. System displays their payment for verification.
- ◆ 3. Cashier confirms.
 - 3a. Cashier cancels payment step:
 - ❖ 1. System reverts to "item entry" mode.
- ◆ 4. System sends payment authorization request to an external Payment Authorization Service System, and requests payment approval.
 - 4a. System detects failure to collaborate with external system:
 - ❖ 1. System signals error to Cashier.
 - ❖ 2. Cashier asks Customer for alternate payment.

Fully Use Case Example ₁₂

□ Extensions (or Alternative Flows)

○ 7b. Paying by credit:

- ◆ 5. System receives payment approval, signals approval to Cashier, and releases cash drawer (to insert signed credit payment receipt).
 - 5a. System receives payment denial:
 - ❖ 1. System signals denial to Cashier.
 - ❖ 2. Cashier asks Customer for alternate payment.
 - 5b. Timeout waiting for response.
 - ❖ 1. System signals timeout to Cashier.
 - ❖ 2. Cashier may try again, or ask Customer for alternate payment.
- ◆ 6. System records the credit payment, which includes the payment approval.
- ◆ 7. System presents credit payment signature input mechanism.
- ◆ 8. Cashier asks Customer for a credit payment signature. Customer enters signature.
- ◆ 9. If signature on paper receipt, Cashier places receipt in cash drawer and closes it.

Fully Use Case Example ₁₃

□ Extensions (or Alternative Flows)

- 7c. Paying by check...
- 7d. Paying by debit...
- 7e. Cashier cancels payment step:
 - ◆ 1. System reverts to "item entry" mode.
- 7f. Customer presents coupons:
 - ◆ 1. Before handling payment, Cashier records each coupon and System reduces price as appropriate. System records the used coupons for accounting reasons.
 - 1a. Coupon entered is not for any purchased item:
 - ❖ 1. System signals error to Cashier.
- 9a. There are product rebates:
 - ◆ 1. System presents the rebate forms and rebate receipts for each item with a rebate.
- 9b. Customer requests gift receipt (no prices visible):
 - ◆ 1. Cashier requests gift receipt and System presents it.
- 9c. Printer out of paper.
 - ◆ 1. If System can detect the fault, will signal the problem.
 - ◆ 2. Cashier replaces paper.
 - ◆ 3. Cashier requests another receipt.

Fully Use Case Example ₁₄

❑ Special Requirements

- Touch screen UI on a large flat panel monitor. Text must be visible from 1 meter.
- Credit authorization response within 30 seconds 90% of the time.
- Somehow, we want robust recovery when access to remote services such the inventory system is failing.
- Language internationalization on the text displayed.
- Pluggable business rules to be insertable at steps 3 and 7.
- ...

Fully Use Case Example 15

□ Technology and Data Variations List

- *a. Manager override entered by swiping an override card through a card reader, or entering an authorization code via the keyboard.
- 3a. Item identifier entered by bar code laser scanner (if bar code is present) or keyboard.
- 3b. Item identifier may be any UPC, EAN, JAN, or SKU coding scheme.
- 7a. Credit account information entered by card reader or keyboard.
- 7b. Credit payment signature captured on paper receipt. But within two years, we predict many customers will want digital signature capture.
- Frequency of Occurrence: Could be nearly continuous.

Fully Use Case Example ₁₆

❑ Open Issues

- What are the tax law variations?
- Explore the remote service recovery issue.
- What customization is needed for different businesses?
- Must a cashier take their cash drawer when they log out?
- Can the customer directly use the card reader, or does the cashier have to do it?