1 *Unit*

UNIFIED PROCESS AND USE CASE DIAGRAMS

1.1 INTRODUCTION TO OOAD WIT OO BASICS

A software development methodology is a series of processes like

- System Analysis
- Modeling
- Design
- Implementation
- Testing and
- Maintenance

that leads to the development of an application.

There are two orthogonal views of software development.

1) Traditional Technique – focuses on data and functions.
2) Object Oriented methodologies – focuses on objects that combines data and functionality.

Object oriented systems development develop software by building objects that can be easily replaced, modified and reused. Objects has attribute (data) and methods (functions).

Object Oriented systems are

- Easier to adapt to changes
- Easier to maintain
- Promote greater design and code reuse
- Creates modules of functionality

WHAT IS OOAD?

1.1.1 Object Oriented Analysis

Analysis is the process of investigation of the problem and the requirements than finding its solution.

Analysis is classified as

- Requirement Analysis
- Object Oriented Analysis

In object oriented analysis finding and describing the objects or concepts in the problem domain is done.

Example:

In flight information system, during analysis the concepts identified are

- Plane
- Flight
- Pilot

1.1.2 Object Oriented Design

Design is the process of finding a conceptual solution that fulfills the requirements than implementing it.

Design is classified into

- Database design
- Object oriented design

In object oriented design, software objects are defined and collaborated to fulfill the requirements.

Example

In flight information system plane is a software object having

- Attribute – TailNumber
- Method – getFlightHistory

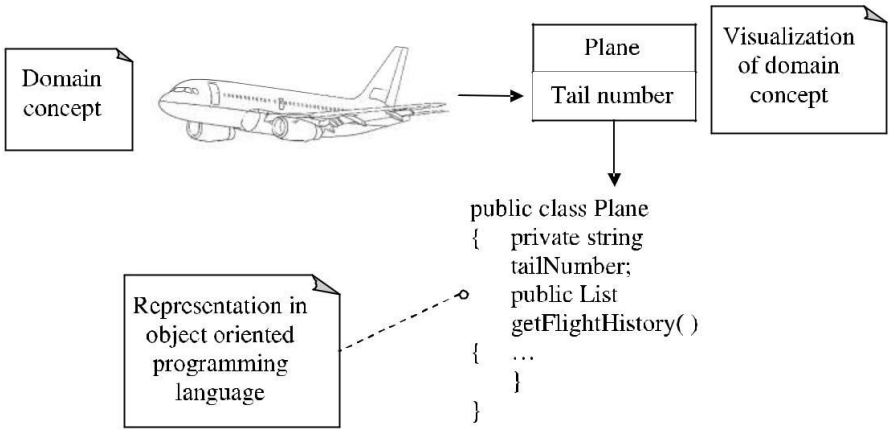


Figure 1.1 Object oriented representation of objects

To summarize

- Analysis means – **DO THE RIGHT THING**

- Design means – **DO THE THING RIGHT**

1.1.3 Example

The key steps in the analysis and design include.

1) Define use cases
2) Define domain model
3) Define interaction diagrams
4) Define design class diagrams

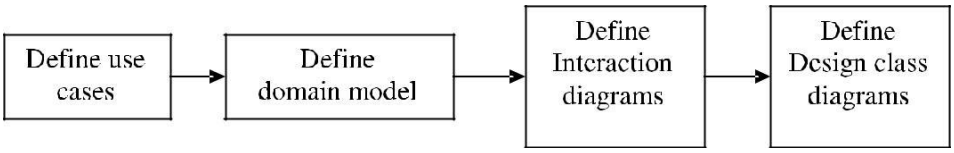


Figure 1.2 Key steps in OOAD

These key ideas are expressed with the "dice game" → A software simulates a player rolling two dice.

If the total is 'seven', they win; otherwise they lose.

Define use cases

Requirement analysis consists of use cases scenarios of how people use the application.

In the dice game, the use cases include

- Play a Dice game

  Player rolls dice

  System gives results

  Player wins if dice value totals seven

  Otherwise loses

Define a Domain Model

Object oriented analysis describes the problem domain by identifying

- Concept
- Attributes
- Associations

All these are represented in domain model.

The concepts, attributes and association of the dice game are represented as

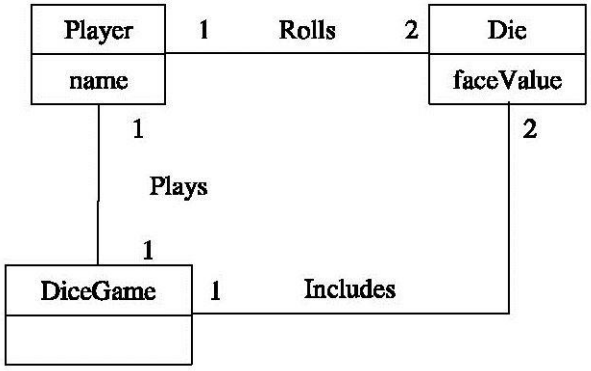


Figure 1.3 Partial Domain Model of dice game

Define Interaction Diagrams

- In Object Oriented Design, software objects are defined with their responsibilities and collaborations.
- Sequence diagram is used to represent the collaborations.
- It is the flow of messages between software objects.
- ***Example:*** In dice game, the player rolls the dice in real world.
- But in software design it is illustrated as die object shown in the following sequence diagram.

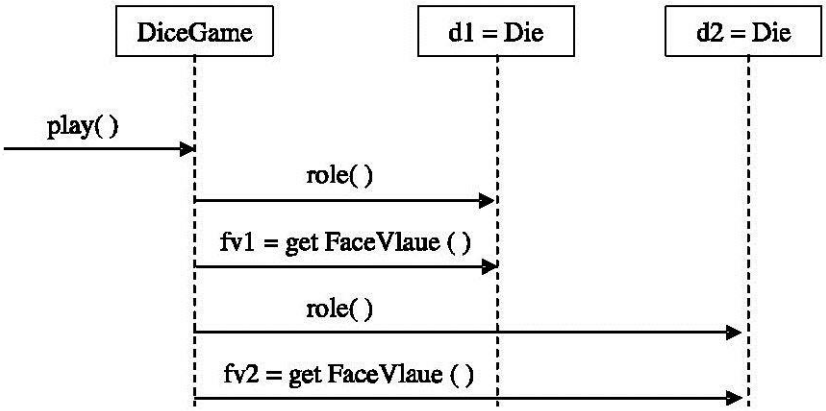


Figure 1.4 Sequence Diagram for dice Game

Define design class diagram

- In the interaction diagram (sequence diagram) the dynamic view of collaborating objects is shown.
- The static view of the classes can be shown with class diagram.
- *Example:* The class diagram for the above sequence diagram of the dice game can be shown, Here Dice game has play method. Dice has roll and getFaceValue methods.

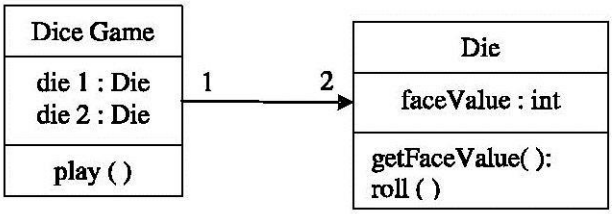


Figure 1.5 Partial design class diagram

1.2 WHAT ARE UP (UNIFIED PROCESS) PHASES

Software development process is an approach to i) Build, ii) Deploy and iii) Maintain the software

- The Unified Process (UP) is a software development process mainly for building object oriented systems.
- The detailed refinement of the unified process is Rational Unified Process or RUP.
- The Unified Process
  - is an iterative process
  - provides an example structure for how to do Object Oriented Analysis and Design
  - is flexible
  - can be applied to lightweight and agile approach.
  - combines iterative lifecycle and risk driven development into a cohesive and well documented process description.

UP Phases

There are four major phases of UP

They are

1) Inception
2) Elaboration
3) Construction
4) Transition

1.2.1 Inception

Inception is a feasibility phase where enough investigation is done to support a decision to continue or stop.

Inception includes

- Approximate vision
- Business scope

- Vision
- Scope
- Vogue Estimates

1.2.2 Elaboration

Elaboration is the phase where the core architecture is iteratively implemented.

Here the high risk issues are mitigated.

Elaboration includes

- Refined vision
- Iterative implementation of the core architecture
- Resolution of high risks
- Identification of most requirements and scope
- More realistic estimates

1.2.3 Construction

Construction involves

- Iterative implementation of the remaining lower rise and easier elements
- Preparation for deployment

1.2.4 Transition

Transition involves

- Beta tests and
- Deployment.

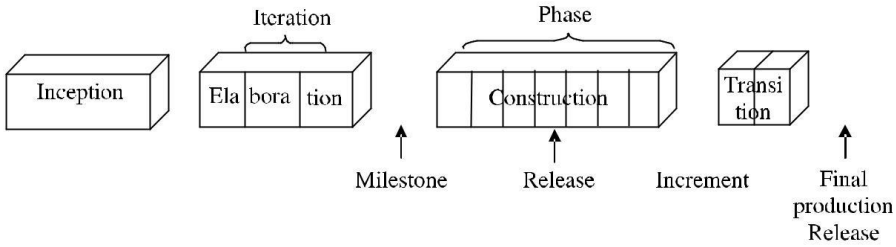


Figure 1.6 Schedule oriented terms in UP

Milestone	–	An iteration endpoint where some significant decision/evaluation occurs
Release	–	A stable executable subset of the final product
Increment	–	the difference between the release of two subsequent iterations
Final production release	–	The system is released for production use

Figure 1.6 Schedule oriented terms in UP

UP Disciplines

There are several disciplines in UP.

Some are

- Business Modeling
  - The domain model artifact to visualize concepts in application domain.
- Requirements
  - Usecase model

Supplementary specification to capture functional and non functional requirements

- Design
  - Design Model artifact
  - Design software objects
- Implementation

Programming and building the system

- Environment
  - Establishing the tools and customizing the process for the project.

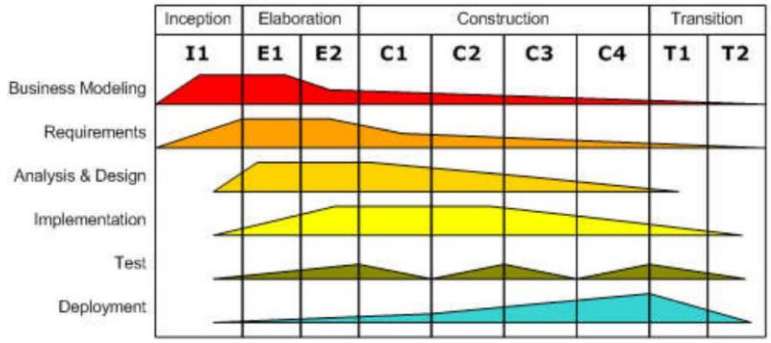


Figure 1.7 Disciplines and phases

1.3 UML DIAGRAMS

UML is expanded as '**UNIFIED MODELING LANGUAGE**'

The UML is a visual language for

- Specifying
- Constructing and
- Documenting artifacts of the systems

The word visual indicates UML as the defacto standard diagramming notation for drawing or presenting pictures related to software – i.e., Object Oriented Software.

1.3.1 Three ways to apply UML

(1) UML as sketch

Informal and incomplete diagrams are created to explore difficult parts of the problem or solutions.

(2) UML as blue print

Detailed design diagrams are used for

- Reverse Engineering or
- Forward engineering

A UML tool reads the source and generates UML class, sequence and package diagrams in reverse engineering.

Detailed diagrams can provide guidance for code generation.

UML as programming language

Complete executable specification of software system in UML.

Executable code is automatically generated.

It is still under development for usability.

1.3.2 Three perspectives to Apply UML

UML describes raw diagrams like

- Class diagrams
- Sequence diagrams

In three different perspectives

1) Conceptual
2) Specification
3) Implementation

(1) Conceptual Perspective

Diagrams interpret things in a situation of the real world or domain of interest.

(2) Specification (Software) perspective

The diagrams describe software abstracted or components with specification and interfaces.

(3) Implementation (Software) perspective

The diagrams describes software implementations in a particular technology like Java.

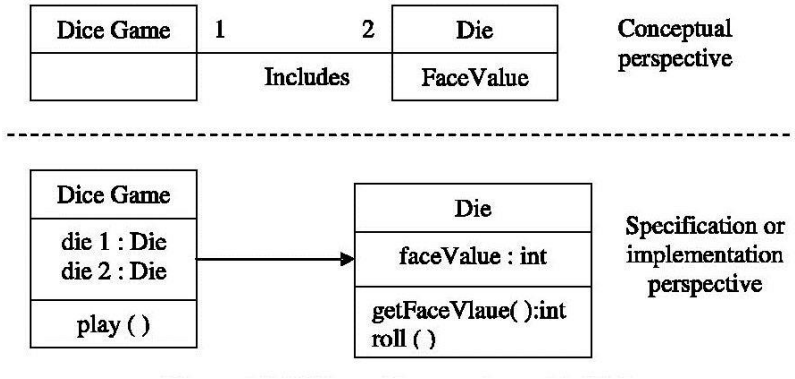


Figure 1.8 Different Perspectives with UML

"Class" Perspectives

Classes are represented by rectangular boxes.

Classes represent

- Physical things
- Abstract concepts
- Software things
- Events etc.

The class related terms used are

- **Conceptual class or Domain concepts**

  Realworld concept or thing

  A conceptual or essential Perspective

- **Software class**

  A class representing a specification or implementation perspective of a software component.

- **Implementation class**

  A class implemented in a specific Object Oriented language such as Java.

1.4 USE CASE DIAGRAM

The use case concept was introduced by Ivar Jacobson in object oriented software engineering (OOSE).

Usecases represent specific flow of events in the system.

Use cases define the outside (actors) and inside (use case) of the system's behavior.

A use case diagram is a graph of actors, a set of use cases enclosed by a system boundary, communication (participation) association between the actors and the use cases and generalization among the use cases.

Example: A USE CASE DIAGRAM FOR A HELP DESK

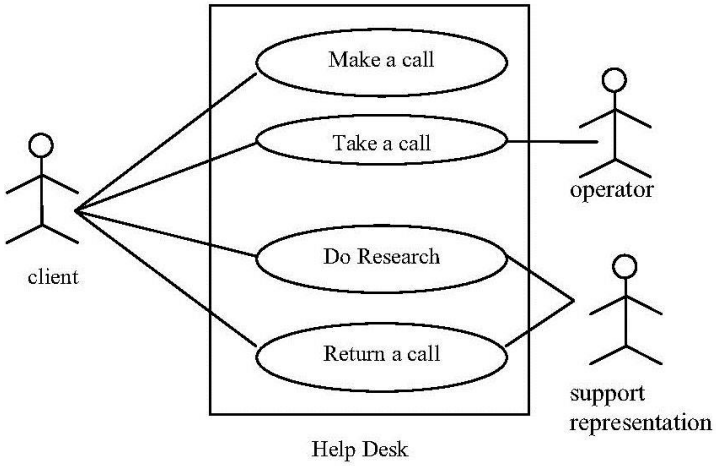


Figure 1.9

A use case is shown as an ellipse containing use case name.

Use case name can be placed below or inside the ellipse.

An actor is shown as class rectangle with the level <<actor>> or the label and a stick figure or just the stick figure with the name of the actor below figure.

Three Representation of Actor Are Equivalent

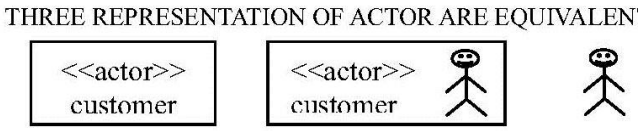


Figure 1.10

The relationship shown in use case diagram are:

1) Communication
2) Uses
3) Extends

The communication relationship of an actor in a use case is shown by connecting the actor symbol to the use case symbol with a solid path. The actor is said to communicate with "use case".

A uses relationship between the use cases is sown by a generalization arrow from the use cases.

The extends relationship is used when we have one use cases similar to another but does a bit mode.

1.4.1 USE CASES

A use case is a set of scenarios tied together by a common use goal.

A scenario is a sequence of steps describing an interaction between a user and a system.

Example Use Case Text

A simple format of use case involves describing the primary scenario as a

- Sequence of numbered steps and
- Alternatives as variations on that sequence

But a product

1) Customer browser through catalog and selects items to buy.
2) Customer goes to check out.
3) Customer fills in shipping information (address; next-day or 3-day delivery)
4) System presents full pricing information including shipping.
5) Customer fills in credit card information
6) System authorizes purchase
7) System confirms sale immediately
8) System sends confirming email to customer

Alternative

Authorization failure

At step 6 system fails to authorize credit purchase

Allow customer to reenter credit card information and reentry

Alternative

Regular customer

3a. System displays current shipping information, pricing information and last four digit of credit card information.

3b. Customer may accept or override these defaults

Return to primary

There is no specific standard for the use case in the UML.

1.4.2 Use Case Diagrams

The diagrams which visualizes the use cases are called use case diagrams. This was introduced by Jacobson in 1994.

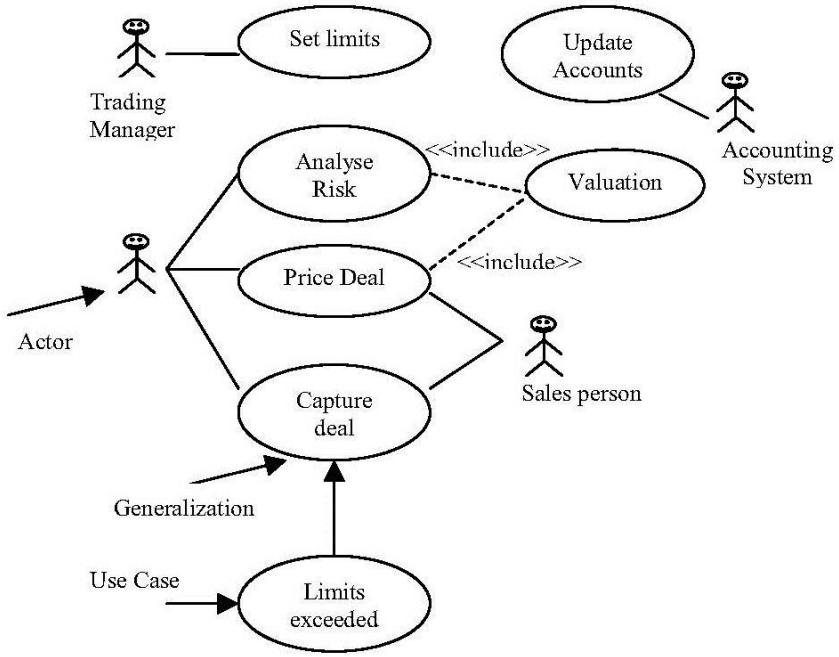
The use case diagram is a part of UML.

One of the projects in which use cases are involved are

- Keeping each one on an index card and
- Sorting the cards into piles to show what needed building in each iteration

Example

We can see the use cases for a financial trading system.



Actors

An actor is a role that a user plays with respect to the system.

There are four actors in the above diagram

- Trading manager
- Trader
- Salesperson and
- Accounting System

This can be also known as role but as far as system is concerned, all play the same role.

A user may play more than one role.

For example,

- A senior trader any play the trading manager role and also the role of a trader.
- A trader may also be a salesperson

Actors carry out use cases. A single actor may perform many use cases and a use case may have several actors performing it Actors are most useful than the use cases. It is easier to arrive at a list of actors first and then try to workout the use cases for each other. Actors need not be human beings though they are represented as stick figures in the use case diagram.

An actor can also be a external system that needs some information from the current system. Some people show every external system or human actor on the use case diagram. Others prefer to show the initiator of the use case. In some situations, we can track the actors later like.

The system may need configuring for various kinds of users. Here each kind of user is an actor and use cases show what each actor needs to do. Tracking who wants use cases can help us to negotiate priorities among various actors. Some use cases don't have clear links to specific actors.

Example:- In a utility company one of the use cases is to send out bill.

Here no ser requests a bill or no customer is objecting if the doesn't get the bill. The best guess of the actor is given by the Billing Department. One way to identify use cases is external events.

1.4.3 Use Case Relationship

Links exist between actors and use cases. In addition to that several kinds of relationship exist between use cases like

1) Include
2) Use case Generalization
3) Extend

Include

When we have a chunk of behavior that is similar across more than one use case and we don't want to copy the description of the behavior.

Example: Both Analyze Risk and Price Deal require to value the deal.

Instead of having a fair chunk of writing and copying it again, we can write a separate value Deal use case and refer it from the original use cases.

Use Case Generalization

When we have one use case that is similar to another use cases but does a bit more then we use, use case generalization.

Example: The basic use case is capture Deal.

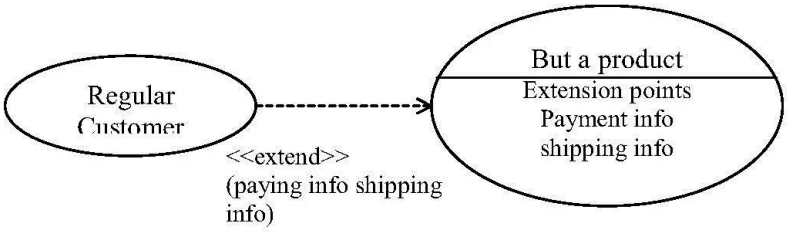
One special case is when the limit is exceeded ie., the maximum amount the trading organization has established for a particular customer.

Extend

The third relationship called as extend is similar to generalization but with more rules to it.

The extending use case may add behavior to the base use case, but it must declare certain "extension points" and the extending use case may add additional behavior only at those extension points.

Extend Relationship



A use case may have many extension points and an extending use case may extend one or more of these extension points.

Apply the following rules

Use include when you are repeating yourself in two or more separate use cases and you want to avoid repetition.

Use generalization when you are describing a variation on normal behavior and you wish to describe it casually.

Use extend when you are describing a variation on normal behavior and you wish to use the more controlled form, declaring your extension points in your base use case.

Business and System Use Cases

Two different use cases are there. They are

- System Use Case
- Business Use Case

A system use case is dealing with the interaction with the software.

Business use case discusses how a business responds to a customer or an event.

Useage of Use Case

Most of the use cases will be generated during the phase of the project but they will be unconverted as we proceed the project.

Every use case is a potential requirement and, we can plan it only after we capture a requirement.

Use cases represent an external view of the system.

During a recent OOPSLA panel discussion, several use case experts said that for a 10 person year project, they would expect around a dozen use cases. These are base use cases, each use case would have many scenario and many variant use cases.

1.5 CASE STUDY - THE NEXTGEN POS SYSTEM INCEPTION

Inception is the initial short step to establish a common vision and basic scope for the project.

Inception deals with "Environment the product scope, vision and business case".

The purpose of inception is

- To collect enough information to establish a common vision.
- To decide if moving forward is feasible.

1.5.1 Use cases

The use cases are text stories used to discover and record requirements.

Use cases influence many aspect of a project.

They are input to artifacts.

The use cases also influence.

1) Analysis
2) Design
3) Implementation
4) Project management
5) Test artifacts

The influence of UP artifacts is shown in the following figure.

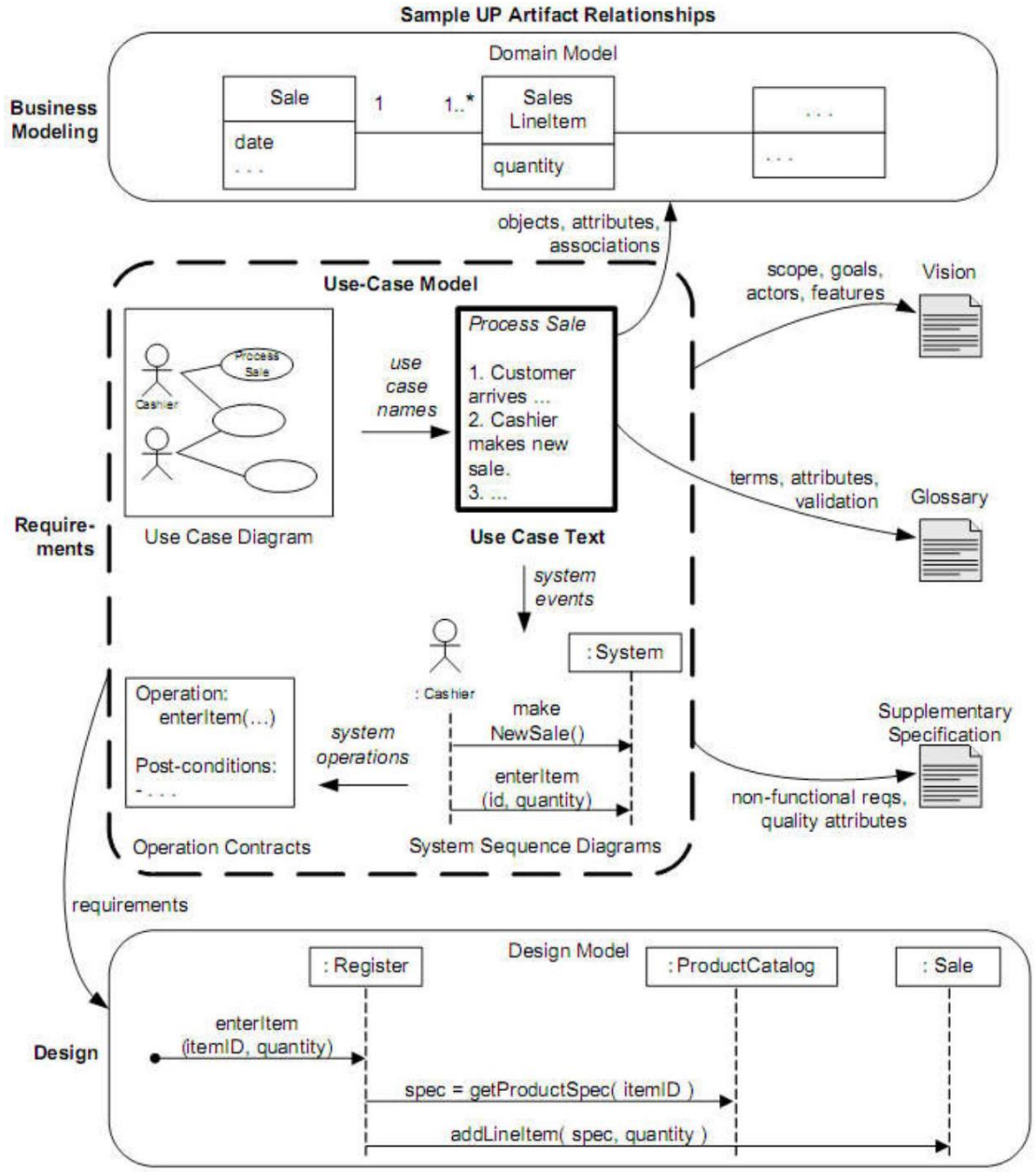


Figure 1.13 Sample UP Artifacts Relationships

1.5.2 Definitions

(1) Actors

Something with behaviour, such as

A person (identified by role)

Computer system

or organization

is called an actor

Example:

Cashier

(2) Scenario

A scenario is a specific sequence of actions and interactions between actors and the system.

It is called use case instance.

Example:

Scenario of successfully purchasing items with cash.

Scenario of failing to purchase items because of credit payment denial.

Use Case

Use case is a collection of related success and failure scenarios that describes an actor using a system to support a goal.

Example:

Main success scenario:

A customer arrives at a shop to return items.

The cashier uses POS system and records returned items.

Alternative scenarios

If customer paid credit and if return to the credit account is rejected, pay the customer with cash.

If the item is not found, notify the cashier to enter manually etc.

Thus RUP (Rational Unified Process) defines use case as

"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".

1.6 USE CASE MODELING

Use cases and Use case Model

Use Cases are text documents and not diagrams.

Use case modeling is primarily an act of writing text, not drawing diagrams.

Use cases are requirements Functional or Behavior

Use cases emphasize the user goals and perspectives.

The strength of use cases is the ability to scale both up and down in terms of

- Sophistications and
- Formality

1.6.1 Three kinds of Actors

An actor is anything with behavior including the System Under Discussion (SUD) when it calls upon services of other systems.

Actors are roles played not only by people but by organization.

Example:

- Software
- Machines

In relation to SUD there are three types of actors

Primary actor

Supporting actor

Offstage actor

(1) Primary actor

It has user goals fulfilled by using services of SUD.

Example: Cashier

Finds user goals that drive use cases

(2) Supporting actor

It provides services to SUD.

Example: Automated payment authorization

(3) Offstage actor

It has an interest in the behavior of use cases but not primary or supporting.

Example: A government tax agency.

Use case formats

There are different formats for use cases like

- Brief
- Casual
- Fully dressed

Brief

- One paragraph summary of main success scenario.
- It is written during early requirements analysis.
- It is written to get a quick sense of subject and scope
- It can be written in a few minutes.

Example:

Process sale (given previously)

Casual

1) Informal paragraph format
2) It involves multiple paragraphs that cover various scenarios.

Example:

Handle Returns

Fully dressed

- Detailed steps are written.
- Supporting sections like preconditions are present.

Example:

Next Gen case study

Processed sale, Fully Dressed style

More detailed and structured style is given in fully dressed format.

During the requirement analysis, 10% of the critical use cases are written in this format.

The design and programming then starts on the most significant use cases or scenarios.

Many templates are available for detailed use cases.

The popular format template is given by Alistair Cockburn.

The template is shown below

Use case section	Comment
Use case Name	Start with a verb
Scope	System under design
Level	“user-goal” or “subfunction”
Primary Actor	Calls on the system to deliver its services
Stakeholders & interest	Who cares about use case? 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 input methods and data formats
Frequency of occurrence	Influences investigation, testing and timing of implementation
Miscellaneous	Such as open issues

This template can be used to the NextGen casestudy and the detailed illustration is shown below.

The detailed branching scenarios are illustrated.

Use Case UC1: Process Sale

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.
- 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.

Main Success Scenario (or Basic Flow):

- Customer arrives at POS checkout with goods and/or services to purchase.
- Cashier starts a new sale.
- Cashier enters item identifier.
- System records sale line item and presents item description, price, and running total. Price calculated from a set of price rules.

Cashier repeats steps 3-4 until indicates done.

- System presents total with taxes calculated.
- Cashier tells Customer the total, and asks for payment.
- Customer pays and System handles payment.

(5) 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).

(6) System presents receipt.

(7) Customer leaves with receipt and goods (if any).

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.

1a. Customer or Manager indicates 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 reenters 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)

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 perform 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).

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.

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.

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.

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.

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.

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.

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.

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.

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.

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?

Meaning of the sections

Preface elements

(1) Scope

- It bounds the system under design

- Since usecase describes use of one software it is known as system use case
- Use cases describe how a business is used by its customers and partners.
- The enterprises level process description is called Business Use Case, which is an example of wide applicability of use cases.

(2) Level

Use cases are classified as

- User goal level or
- Subfunction level

in Cockburn's system

User goal level

The user goal level describes the scenarios to fulfill the goals of a primarily actor

It corresponds to elementary Business Process (EBP) in Business Process Engineering.

Subfunction level

A subfunction level use case describes substeps to support a user goal.

It is created to eliminate duplicate substeps shared by several regular use cases

Example:

Pay by credit is a subfunction use case which is shared by many regular use cases

Primary Actor

It calls upon system services to fulfill a goal

Stakeholders and InterestList

- This list is more important and practical
- It suggests and bounds what the system has to do.

"The system operates a contract between stake holders, with the use cases detailing the behavioral parts of that contract".

The use case, as the contract for behavior, captures all and only the behaviors related to satisfying stakeholder's interest.

The stakeholder interest viewpoint provides a through and methodical procedure for discovering and recording all required behaviors.

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 commission updated
- …….

Preconditions and Success Guarantees (Post conditions)

- Before beginning a scenario, preconditions state what must always be true.
- Preconditions are assumed to be true and are not tested within use case
- ***Example:*** "The system has power"

Post conditions (Success Guarantees)

The post condition must be true on successful completion of use case (ie, Main success scenario or some alternate path)

Pre conditions: Cashier is identified and authenticated.	
Success Guarantee: (post condition)	Sale is saved Tax is correctly calculated Accounting and inventory are updated Commissions recorded Receipt is generated

Main Success Scenario and Steps (or Basic Flow)

It is called " happy path" scenario, Basic flow or typical flow.

Main Success Scenario describe a typical success path that satisfies the interests of stakeholders.

It does not have any conditions or branching.

All conditions and branching statements can be present in Extension section.

The scenario has three kinds of steps

(i) An interaction between actors
(ii) A validation
(iii) A state change by the system (example: recording or modifying something)

The actor's names are always written in capital letters for ease of identification.

Main Success Scenario

1) Customer arrives at a POS checkout with items to purchase
2) Cashier starts a new sale
3) Cashier enters item identifier
4) ……Cashier repeats steps 3- 4 until indicates done
5) ……

Extensions (or Alternate Flows)

- Extensions comprise the majority of text
- They are more complex than Main Success Scenario section because they indicates all scenarios success and failure.
- They are notated with respect to its steps 1…..N
- An extensions has two parts
  - the condition and
  - the handling

Extension handling may be in one step or includes a sequence with a range of steps

3-6a: Customer asks Cashier to remove an item from the purchase:

1) Cashier enters item identifier for removal from the sale.
2) System displays updated running total.

If the extension is not halting, the scenario merges back with the main success scenario at the end of extension handling.

Some complex extension points can be had as a separate use case.

Failures within the extension can also be demonstrated.

7b. Paying by credit

1) Customer enters their credit account information
2) System sends payment authorization request to an external payment Authorization Service system, and requests payment approval.

   2a) System detects failure to collaborate with external system

   1) System signals error to cashier
   2) Cashier asks customer for alternate payment.

The labels *a, *b,… can be used to describe an extension condition.

Sometimes, a usecase branches to perform another use cases scenario.

Example: In the Process Sale use case, we use the Find Product help.

The second use case us shown with underlining.

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

1) System signals error and rejects entry.
2) Cashier responds to the error:

   2a. …..

   1. 2c. Cashier performs Find Product Help to obtain true item ID and price.

Special Requirements

- The non functional requirements
- Quality attributes or
- Constraints

Relates specifically to use case are mentioned as special requirements The quality may be of

- Performance
- Reliability
- Usability
- Design constraints etc.
- Touch screen on a large flat panel monitor
- Text must be visible from 1 meter
- Credit authorization response within 30 seconds 90% of the time.
- Language internationalization on the text displayed
- Pluggable business rules to be insertable at steps 2 and 6.

Technology and Data Variations List

There may be technology variations in how things are done.

Example: One stakeholders says

"The POS system must support credit account input using a card reader and the keyboard".

Another stakeholder may prefer some other input/output

This list is the place to record such variations.

Technology and Data Variations List

3a. Item identifier given by laser scanner 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. After sometime it may be digital signature capture.

Other format

A two column or conversational format where intersections are between actors and the system.

It is given by Wirfs-Brock and promoted by Constantine and Lockword.

The same example is shown in two column format.

Use Case UC 1: 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 3-4 until indicates done.	5. Presents total with taxes
6. Cashier tells customer the total and asks for payment	
7. Customer pays	8. Handles payment
	9. Logs all completed sale & send information to the external accounting and inventory systems. System presents receipt

Some prefer one column style and some prefer two column style.

Some guidelines are followed in the styles.

1) Guidelines – Write in an Essential is to 'log in'

a) New Improved

When the goal of cashier is to 'log in'.

Instead of thinking of GUI, dialog box, user ID and password, use keyboards and mice with biometric readers for fingerprint.

b) Essential style writing

Here use cases are written in essential style

The guidelines is "Keep the user interface out focus on intent"

c) Contrasting Examples

……

1) Administrator identifies self

2) System authenticates identity

3) ……..

(i) Identification

(ii) Authentication

Concrete style

The concrete use cases are used as aid to concrete or detailed GUI design work. The text may

- Show window screen shots
- Discuss window navigation
- GUI widget manipulation etc

Example

…….

1) Administrator enters ID and password in dialog box
2) System authenticates Administrator
3) System displays "edit users" window.

2) Guideline: Write Terse Use Cases.

Terse use cases are written by deleting noise words.

Example: small changes like " System Authenticates" than "The system authenticates".

3) Guideline: Write Black Box Use Cases

Black box use cases are recommended because they do not describe internal workings of the system, its components or design.

Here, in black box uses "what the system must do" (function requirement) is specified rather than "how it will do".

Black Box style	Not
The system records the sale	The system writes the sale to a database Or The system generates a SQL INSERT statement for sale..

4) Guideline: Take an Actor and Actor – Goal perspective

1) Write requirements that focus on users or actors of a system with their goals.
2) Focus on understanding what the actor considers a valuable result.

5) Guideline: How to find Use Cases

The procedure is

1) Choose the system boundary (software or hardware application)
2) Identify primary actors
3) Identify goals for each primary actor
4) Define use cases that satisfy user goals

Step 1: Choose system boundary

The boundary of the system. The external primary actors and supporting actors.

Step 2 and 3: Find Primary Actors and Goals.

- The following are some questions to find the Actors and Goals.
  1) Who starts and stops the system?
  2) Who does system administration?
  3) Who evaluates system activity or performance?
  4) Who evaluates log? Etc
- Organise Actors and Goals

  There are two approaches

  1) After getting results, draw them in use case diagram, naming the goals as use cases.
  2) Write actor goal list, review and refine it then draw use case diagram

Sales Activity System

Actor	Goal	Actor	Goals
Cashier	Process sales	Manager	startup
	Process rentals handle returns cash in cash out		shutdown

- Ask about Actor Goals rather than use cases.

  Finding actors and their goals helps us to identify use cases. Its better to ask.

  "What are your goals whose results have measurable value?" that to ask "What do you do?"
- Event Analysis – Another way to find Actors and Goals Identifying external events helps to find use cases.

Example

External Event	From Actor	Goal/ Use Case
Enter sale line item	Cashier	Process a sale
Enter payment	Cashier or Customer	Process a sale

Step 4: Define use cases

- Define one use case for each user goal
- Start the name of use cases with a verb.

Example:

Goal – Process a sale

Use case – Process a sale

Another way

CRUD – Create, Retrieve, Update delete separate goals into on CRUD use case Manage Users – Edit user

Delete User etc.

(6) Guideline: Tests that help to find useful use cases:

1) BOSS test
2) EBP test
3) The size test

BOSS test

The use cases that fail in BOSS test are at some low goal level.

It is not the desirable level of focus for requirements analysis.

Example:

BOSS asks " what have you been doing all day?" Reply " Logging in!"

If the boss is not happy then the use case fails in BOSS test.

BOSS test failing usecases need not be ignored.

User authentication may fail boss test but it is important and difficult.

EBP Test

Elementary business process.

It is defined as

"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 the data in a consistence state".

Example: Approve credit or price order.

The EBP takes five or ten steps i.e., a task done during a single session.

The size test

A use case is not always a single step.

For example:

'Enter an item ID'

Use case seem to be single line but consists of a series of steps.

Example:

Applying tests

1) Negotiate a supplier contract

   Broader and longer than EBP.

   Can be modeled as business use case than system use case.

2) Handle returns

   Ok with boss test

   Seems like EBP

   Size is good

- Log In

  Boss not happy so fails boss teat.

- Move price on game board

  Single step. So fails size test.

Applying UML: Use case Diagrams

- The names of use cases and actors
- The relationship between them are shown by UML diagrams.

A use case diagram shows an excellent picture of system context.

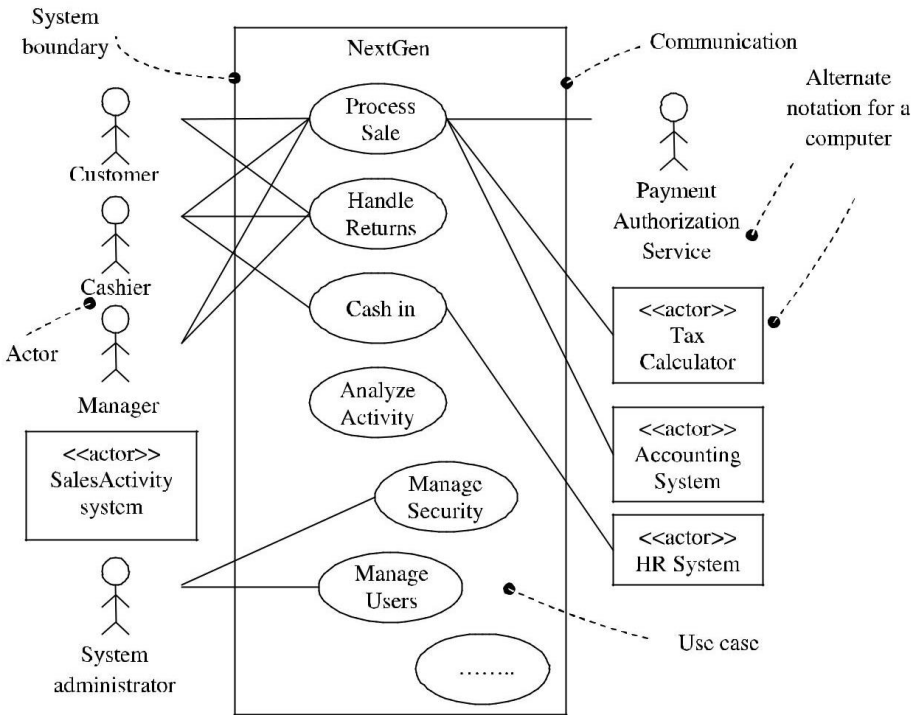


Figure 1.14 Partial Use case context diagram

The context diagram in UML shows the boundary of a system that lies outside it.

The use case diagram is a communication tool that summarizes the behavior of a system and its actors.

Guidelines

(1) The notation for actors

Human actors – stick figures

Computer system actors – actor box with single character bracket.

Example:

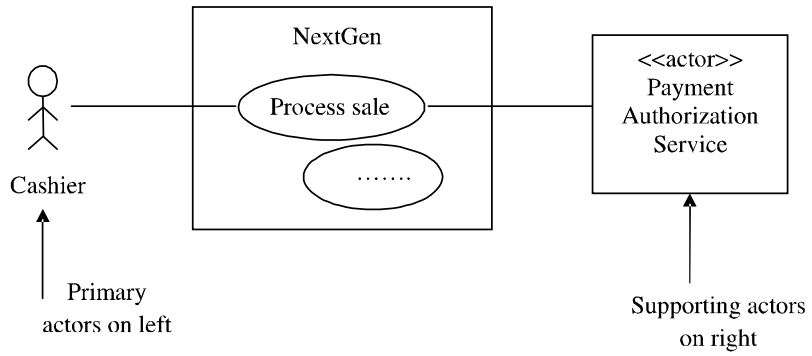


Figure 1.15 a) Notation of actor

Some prefer external computer system actors like.

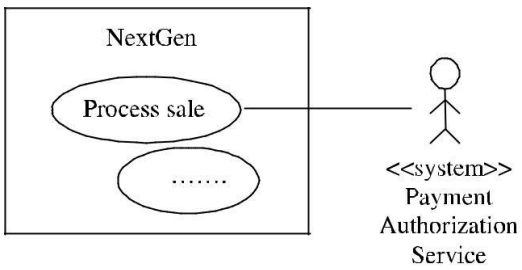


Figure 1.16 b) Actor Notation

(2) Guide line

Keep the diagram short and simple.

An organization cannot spend too much time on writing use case.

Applying UML: Activity Diagrams

The UML has an activity diagram.

It is used to visualize work flows and business processes.

It also describes complex work flows and concurrent actions.

Benefits of Use cases

- Use cases are simple and widely understood.
- Use cases focus on key actors, their goals and common tasks.

Example:

Another example is "Monopoly Game".

The typical use case in monopoly game is "Play Monopoly Game".

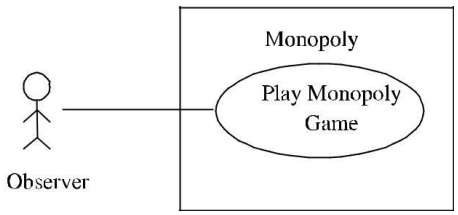


Figure 1.17 Monopoly Game

The Use case Text is given by

Use case UC1 : Play Monopoly Game

Scope : Monopoly application

Level : User goal

Primary actor : Observer

Stakeholders and interest :

Observer : Wants to easily observe the output of the game simulation.

Main success scenario:

1) Observer requests new game initialization, enters number of players.
2) Observer starts play
3) System displays game trace for next player move. Repeat step 3 until a winner or observer cancels.

Extensions:

*a. At any time, system fails:

1) Observer restarts system
2) System detects prior failure, reconstruct state and prompts to continue
3) Observer chooses to continue

Special requirements

Provide both graphical and text trace modes.

Additional Examples for use cases:

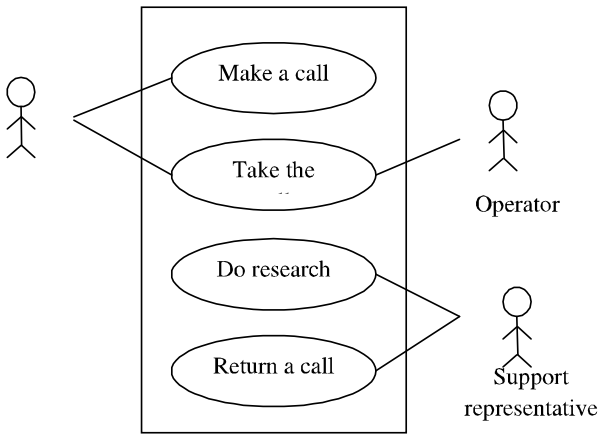


Figure 1.18 Use Case Diagram of a HelpDesk

This use case diagram shows the relationship among actors and use cases within a system.

Process: How to work with use cases in Iterative methods?

The use cases driven development includes

- Functional Requirements
- Use cases are part of iterative planning
- Design is driven by use cases
- Organization manuals influence use cases
- Functional or system testing deals with scenarios of use cases
- UI or shortcuts are created for common tasks.

Use Cases in Inception

During inception all the use cases are not written in fully dressed format.

Some of the use cases are identified according to magnitude, complexities.

Use Cases in Elaboration

In elaboration user goals are refined and 80 - 90% of use cases are written details.

Use case relationship

Links exist between actors and use cases. Several relationships exists between use cases like:

1) Include
2) Use case generalization
3) Extend

Include

When we have a chunk of behavior that is similar across more than one use case and we don't want to copy the description of the behavior.

Example:

Both analyze Risk and Price require to value the deal.

Instead of having a fair chunk of writing and copying it again, we can write a separate value deal use case and refer it from the original use cases.

Use case generalization

When we have one use case that is similar to another use cases but does a bit more then we use, case generalization.

Example:

The basic use case is capture deal.

One special case is when the limit is exceeded i.e., the maximum amount the trading organization has established for a particular customer.

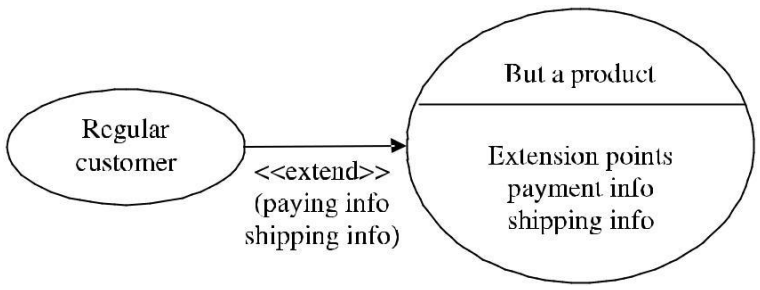
Extend

The third relationship called as extend is similar to generalization but with more rules to it.

The extending use case may add behavior to the use case, but it must declare certain "extension points" and the extending use case may add additional behavior only at those extension points.

Extend relationship

A use case may have many extension points and an extending use case may extend one or more of these extension points.



Apply the following rules

Use *include* when you are repeating yourself in two or more separate use cases and you want to avoid repetition.

Use generalization when you are describing a variation on normal behavior and you wish to describe it casually.

Use extend when you are describing a variation on normal behavior and you wish to use the more controlled form, declaring your extension points in your base use case.

Business and system Use cases

Two different use cases are there. They are

- System Use Case
- Business Use Case

A system use case is dealing with the interaction with the software. Business use case discusses how a business responds to a customer or event.

Useage of Use Case

Most of the use case will be generate during the phase of the project but they will be uncovered as we proceed the project.

Every use case is a potential requirement and, we can plan it only after we capture a requirement.

Use cases represent an external view of the system.

1.7 RELATING USE CASES

Use cases can be related to each other.

Example:

Regular use cases like Process Sale. Process Rental can have subfunctions use case like HandleCredit Payment.

Relationship among use cases

- Improves communication and comprehension of use cases
- Reduces duplication of text
- Improves management of use case document

Writing the use case text is highly essential. This improves comprehension and reduces duplication.

Relationships

The common relationships present are

(i) include relationship

(ii) extend relationship and

(iii) generalize relationship

1.7.1 Include Relationship

- It is the common and important relationship
- The partial behavior is used in common among several use cases

Example

Paying by credit occurs in several use cases like Process Sale, Process Rental, Contribute to Lay-away Plan etc.

Instead of duplicating the text, separate it into its own subfunction use case and indicate its inclusion.

Guideline to use include relationship – Fowler

"Use include when you are repeating yourself in two or more separate use cases and you want to avoid repetition"

Longer use cases can be decomposed into subunits to improve comprehension.

They include relationship is used when

- They are duplicated in other use cases
- A use case is very complex and long, and separating it into subunits aid comprehension.

Example:

UC1 : Process Sale

…..
Main Success Scenario
Customer arrives at a POS Checkout with goods and/or services to purchase
……
Customer pays and system handles payment
……..
Extensions:
7b. Paying by credit : Include Handle Credit Payment
7c. Paying by check : Include Handle Check Payment

UC7: Process Rental

……
6b. Paying by Credit : Include handle Credit Payment …….
The underlined functions are the include functions for highlighting The subfunction "Handle Credit Payment" can be expanded as follows.

UC12 : Handle Credit Payment

…….

Level : Subfunction

Main Success Scenario

1) Customer enters their credit account information
2) System send payment authorization request to an external Payment Authorization Service System and request payment approval.
3) System receives payment approval and signals approval to Cashier.
4) ……

2a. System detects failure to collaborate with external system.

1) System signals error to cashier
2) Cashier asks customer for alternate payment

Another use of the include relationship is to describe the handling of asynchronous event like.

- When user is able to select/branch to a particular function, window or webpage within a range of steps.

Notations of include

- Basic notations use a*, b*,… style labels in the Extensions section.
- A range label such as 3 – 9 can be used for asynchronous event within relatively large range if use case steps.

UC1: Process FooBars

Main Success Scenarios:

1……

Extensions:

a* Customer can edit personal information, at any time Edit Personal Information.

b* Customer selects printing help, at any time, Present Printing Help.

2 – 11, Customer cancels : Cancel Transaction Confirmation ……..

Technologies

1) Concrete use case
   - It is initiated by an actor
   - It performs the entire behavior desired by actor

   Example:

   Process sale

2) Abstract use case

It is never instantiated by itself

It is a subfunction use case that is part of another use case

Example:

Handle credit payment

It does not stand on its own.

Part of process sale

3) Base use case

A use case that includes another use case.

It is extended or specialized by another use case.

Base use case are usually concrete.

Example:

Process sale is a base use case with represent to handle credit payment subfunction.

4) Additional use case

The use case that is an

Inclusion

Extension

Specialization

Is called additional use case

Example:

Handle credit payment is addition use case in include relationship to process sale.

Additional use cases are abstract.

1.7.2 The Extend Relationship

The extends relationship is used when an use case is similar to another use case but does a bit more.

Here for an use case – create an extending or addition use case and within it, describe where and where and under what condition it extends the behavior of some basic use case.

Example:

UC 1: Process sale (Base use case)

…
Extension points: VIP customer, step1.Payment, step 7
Main Success Scenario:

1. Customer arrives at POS checkout with goods and/or services to purchase

….

7. Customer pays and system handles payment

The base use case can be extended as

UC 15: Handle Gift certificate payment (the extending use case)

…

Trigger : customer wants to pay with gift certificate

Extension points: Payment in Process Sale

Leve l : Subfunction Main Success Scenario:

2. customer gives gift certificate to cashier
3. cashier enters gift certificate ID.

The use of extension point is that extending use case is triggered by some condition.

The extension points are labels in base use case.

The extending use case references as point of extension.

Some extension point may be

"At any point in use case X".

Example:

In asynchronous events like word processor.

- Do a spell check now.
- Do a the saurus now.

The 'include' relationship can also have been used but since 'The process sale' use case should not be modified, it is better to go with 'extends' relationship.

Instead of creating complex use case relationships, updating "extensions" can be done.

1.7.3 The Generailze Relationship

Generalization is the relationship between a more general class and a more specific class.

Example:

The vehicle class behaves as a generalized class that has sub class like Bus, Truck can.

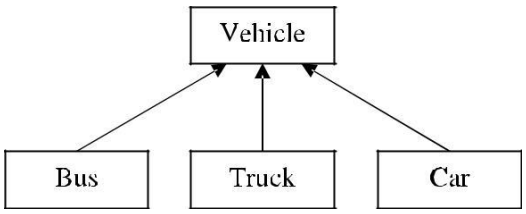


Figure 1.20 Generalize relationship

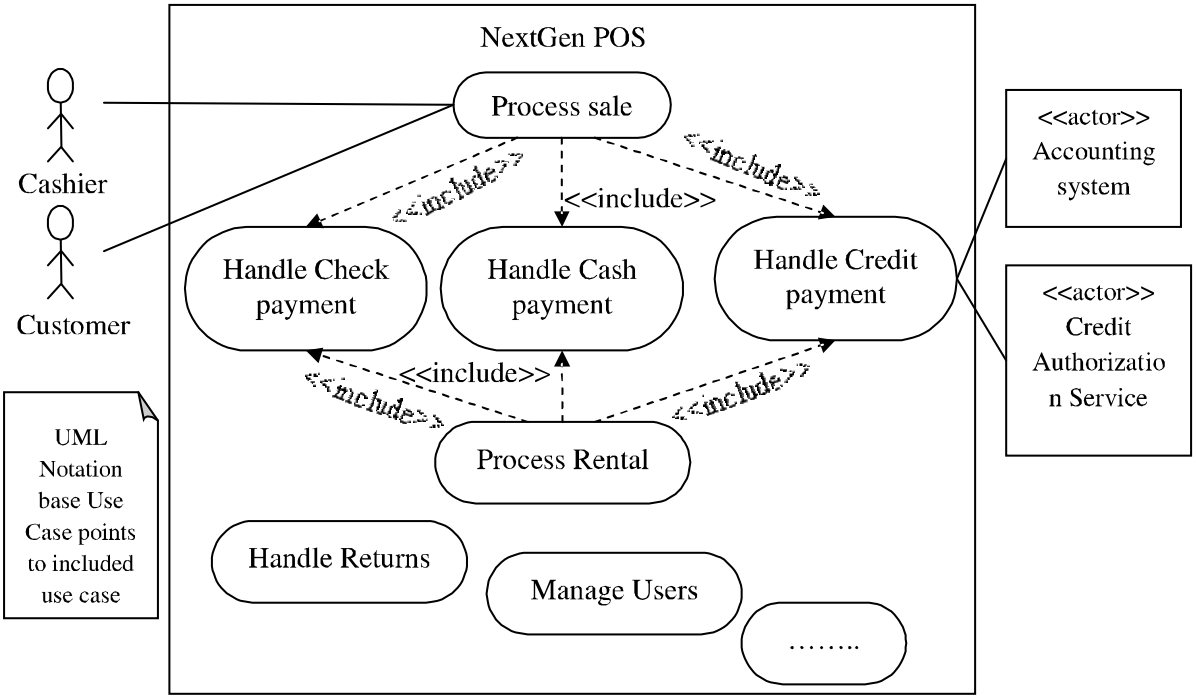


Figure 1.21 Use case diagram for Next Gen POS

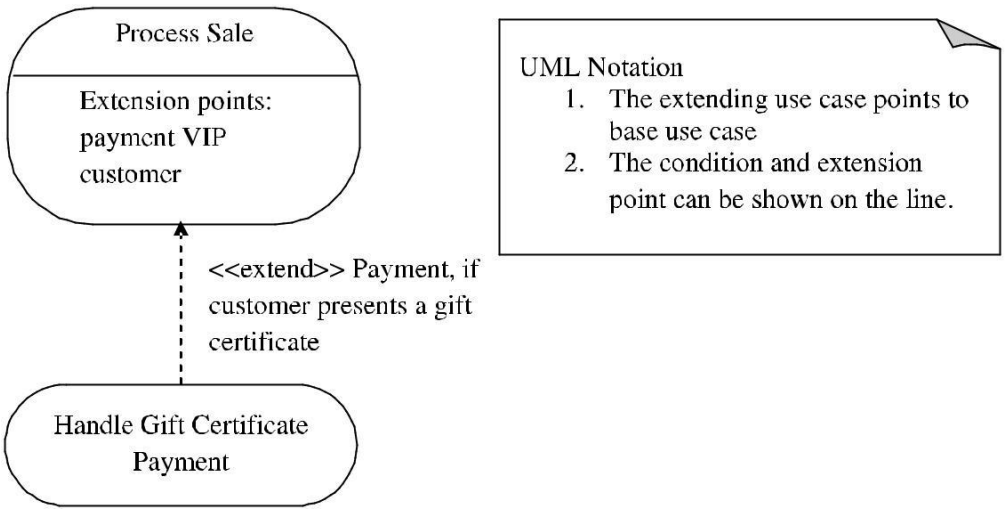


Figure 1.22 The extend Relationship notation

Additional example

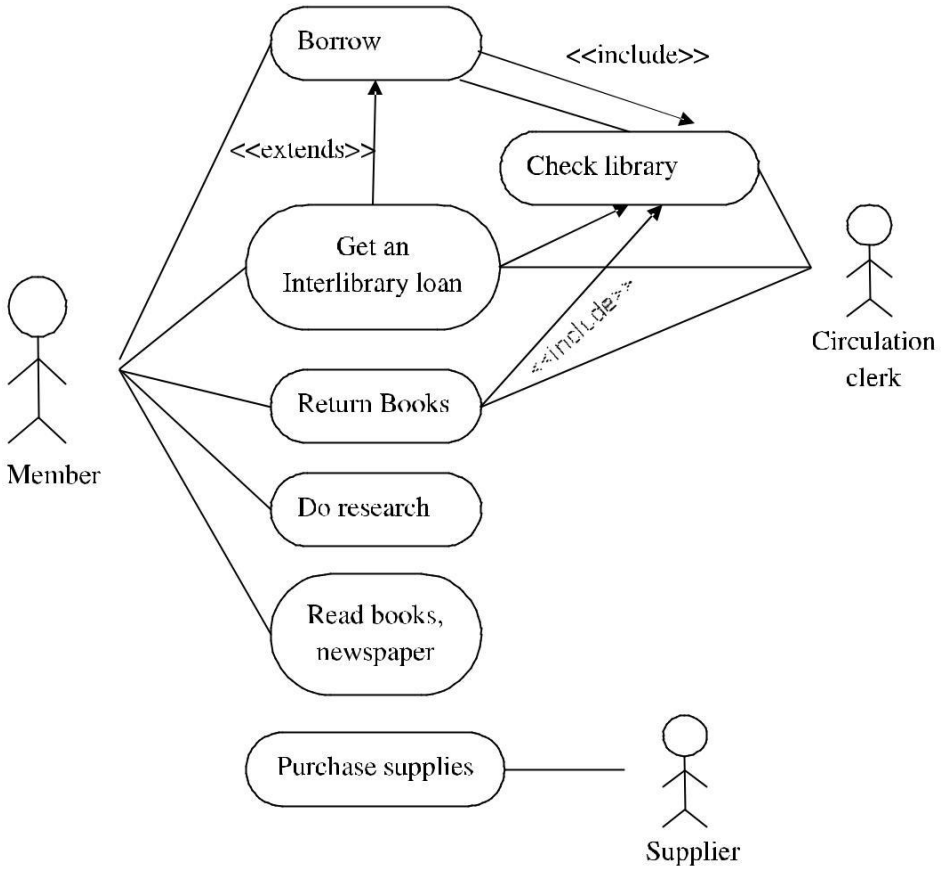


Figure 1.23 Use case diagram for interlibrary loan

The use case diagram above depicts the

- Extends
- includes

Relationships

The interlibrary loan is a special case of checking out books.

Entering into the systems is common for

- getting into the system is common for So all are included

So all are included

1.8 WHEN TO USE: USE CASES DIAGRAMS

- Use cases are used in almost every project.
- They are helpful in exposing requirements and planning the project.
- During the initial stage of a project most use cases should be defined.

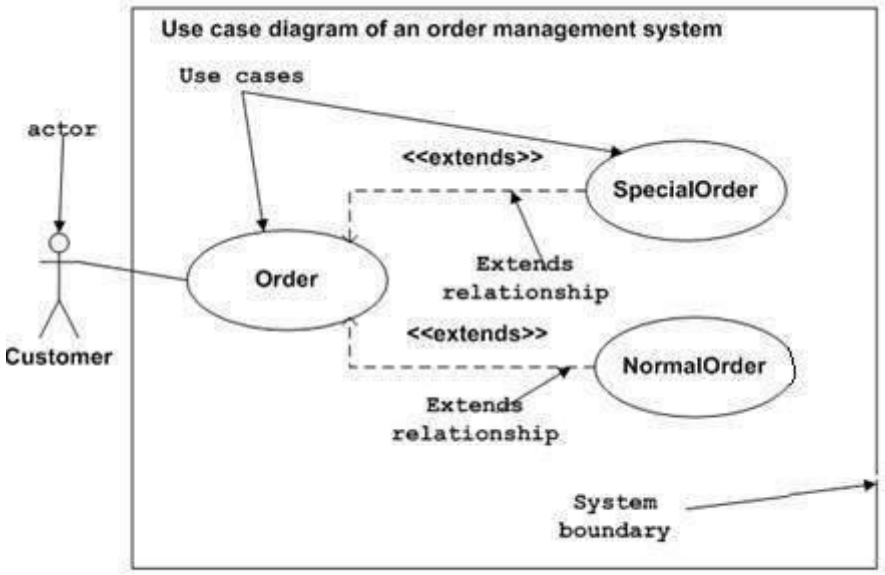


Figure: Sample Use Case diagram