USE CASES & USECASE MODELLING

Use cases are text stories, widely used to discover and record requirements. use cases are text stories of some actor using a system to meet goals. There are 3 formats to represent the use case

I) Brief Format
II) Casual Format
III) Fully Dressed Format

Definition: What are Actors, Scenarios, and Use Cases

An **actor** is something with behavior, such as a person (identified by role), computer system, or organization; for example, a cashier.

A **scenario** is 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.

A **use case** is a collection of related success and failure scenarios that describe an actor using a system to support a goal.

Use Cases and the Use-Case Model

- Use cases are text documents, not diagrams, and use-case modeling is primarily an act of writing text, not drawing diagrams.
- There are also the Supplementary Specification, Glossary, Vision, and Business Rules. These are all useful for requirements analysis.
- The Use-Case Model 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. It also provides a quick way to list the use cases by name.

Motivation: Why Use Cases?

- Lack of user involvement in software projects is near the top of the list of reasons for project failure. Use cases are a good way to help keep it simple, and make it possible for domain experts or requirement donors to themselves write use cases.
- Another value of use cases is that they emphasize the user goals and perspective; we ask the question "Who is using the system, what are their typical scenarios of use, and what are their goals?" This is a more user-centric emphasis compared to simply asking for a list of system features.

Definition: Are Use Cases Functional Requirements

Use cases are requirements, primarily functional or behavioral requirements that indicate what the system will do. A related viewpoint is that a use case defines a contract of how a system will behave

What are Three Kinds of Actors?

An actor is anything with behavior, including the system under discussion (SuD) itself when it calls upon the services of other systems. **Primary and supporting** actors will appear in the action steps of the use case text. **Actors are roles played not only by people, but by organizations, software, and machines.** There are three kinds of external actors in relation to the SuD:

1. **Primary actor** has user goals fulfilled through using services of the SuD. For example, the cashier.
    - Why identify? To find user goals, which drive the use cases.
2. **Supporting actor** provides a service (for example, information) to the SuD. The automated payment authorization service is an example. Often a computer system, but could be an organization or person.
    - Why identify? To clarify external interfaces and protocols.
3. **Offstage actor** has an interest in the behavior of the use case, but is not primary or supporting; for example, a government tax agency.
    - Why identify? To ensure that all necessary interests are identified and satisfied. Offstage actor interests are sometimes subtle or easy to miss unless these actors are explicitly named.

Three Common Use Case Formats

- **Brief** - Terse one-paragraph summary, usually of the main success scenario. The prior Process Sale example was brief. It was created during early requirements analysis, to get a quick sense of subject and scope. May take only a few minutes to create.

Example : Process Sale: A customer arrives at a checkout with items to purchase. The cashier uses the POS system to record each purchased item. The system presents a running total and line-item details. The customer enters payment information, which the system validates and records. The system updates inventory. The customer receives a receipt from the system and then leaves with the items. Notice that use cases are not diagrams, they are text.

- **Casual** - Informal paragraph format. Multiple paragraphs that cover various scenarios. It was created during early requirements analysis, to get a quick sense of subject and scope. May take only a few minutes to create.

  Example : Handle Returns Usecase

  Main Success Scenario: A customer arrives at a checkout with items to return. The cashier uses the POS system to record each returned item …

  Alternate Scenarios: If the customer paid by credit, and the reimbursement transaction to their credit account is rejected, inform the customer and pay them with cash.

  If the item identifier is not found in the system, notify the Cashier and suggest manual entry of the identifier code . If the system detects failure to communicate with the external accounting system,

- **Fully Dressed** - All steps and variations are written in detail, and there are supporting sections, such as preconditions and success guarantees. It was created , 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.

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.

Use Case Section	Comment
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.

Use Case UC1: Process Sale : Fully Dressed Format Example

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.
- **Manage**r: 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.

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.

…

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?

Format Description:

Scope : It can be either system use case or Business Usecase .

- system use case : A use case describes use of one software system
- business use case: At a broader scope, use cases can also describe how a business is used by its customers and partners. Such an enterprise-level process description is called a business use case.

Level : Use cases are classified as at the user-goal level or the sub function level, among others.

A user-goal level use case is the common kind that describe the scenarios to fulfill the goals of a primary actor to get work done.

A subfunction-level use case describes substeps required to support a user goal, and is usually created to factor out duplicate substeps shared by several regular use cases an example is the subfunction use case Pay by Credit, which could be shared by many regular use cases.

Primary Actor : The principal actor that calls upon system services to fulfill a goal.

Stakeholders and Interests List It satisfies all the stakeholders' interests. by starting with the stakeholders and their interests before writing the remainder of the use case, we have a method to remind us what the more detailed responsibilities of the system should be.

Stakeholders and Interests:

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.

- …

Preconditions and Success Guarantees (Postconditions)

Preconditions state what must always be true before a scenario is begun in the use case. Preconditions are not tested within the use case; rather, they are conditions that are assumed to be true. Typically, a precondition implies a scenario of another use case, such as logging in, that has successfully completed.

Main Success Scenario and Steps (or Basic Flow) This has also been called the "happy path" scenario, or the more prosaic "Basic Flow" or "Typical Flow." It describes a typical success path that satisfies the interests of the stakeholders.

The scenario records the steps, of which there are three kinds:

1. An interaction between actors.
2. A validation (usually by the system).
3. A state change by the system (for example, recording or modifying something).

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)

Extension scenarios are branches (both success and failure) from the main success scenario, and so can be notated with respect to its steps 1…N. For example, at Step 3 of the main success scenario there may be an invalid item identifier, either because it was incorrectly entered or unknown to the system. An extension is labeled "3a"; it first identifies the condition and then the response. Alternate extensions at Step 3 are labeled "3b" and so forth.

Extensions:

3a. Invalid identifier:

1. System signals error and rejects entry.

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.

An extension has two parts: the condition and the handling.

Guideline: When possible, write the condition as something that can be detected by the system or an actor.

This extension example also demonstrates the notation to express failures within extensions.

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.

Special Requirements

If a non-functional requirement, quality attribute, or constraint relates specifically to a use case, record it with the use case. These include qualities such as performance, reliability, and usability, and design constraints (often in I/O devices) that have been mandated or considered likely.

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.
- Language internationalization on the text displayed.
- Pluggable business rules to be insertable at steps 2 and 6.

Technology and Data Variations List

A common example is a technical constraint imposed by a stakeholder regarding input or output technologies. For example, a stakeholder might say, "The POS system must

support credit account input using a card reader and the keyboard." It is also necessary to understand variations in data schemes, such as using UPCs or EANs for item identifiers, encoded in bar code symbology.

Technology and Data Variations List:

3a. Item identifier entered 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. But within two years, we predict many customers will want digital signature capture.

Guidelines For Use Case Modeling:

Guideline 1. Write in an Essential UI-Free Style

Guideline : Write use cases in an essential style; keep the user interface out and focus on actor intent**.**

Essential Style

Assume that the Manage Users use case requires identification and authentication:

1. Administrator identifies self.
2. System authenticates identity.
3. …

The design solution to these intentions and responsibilities is wide open: biometric readers, graphical user interfaces (GUIs), and so forth.

Concrete Style Avoid During Early Requirements Work

In contrast, there is a concrete use case style. In this style, user interface decisions are embedded in the use case text. The text may even show window screen shots, discuss window navigation, GUI widget manipulation and so forth. For example:

1. Administrator enters ID and password in dialog box
2. System authenticates Administrator.
3. System displays the "edit users" window
4. …

Guideline 2. Write Terse Use Cases : Delete "noise" words. Even small changes add up, such as "System authenticates…" rather than "The System authenticates…"

Guideline 3 : Write Black-Box Use Cases : Black-box use cases are the most common and recommended kind; they do not describe the internal workings of the system, its components, or design. Rather, the system is described as having responsibilities, which is a

common unifying metaphorical theme in object-oriented thinking software elements have responsibilities and collaborate with other elements that have responsibilities.

Black-box style	Not Recommended
The system records the sale.	The system writes the sale to a database. …or (even worse): The system generates a SQL INSERT statement for the sale…

Guideline 4 : Take an Actor and Actor-Goal Perspective : Here's the RUP use case definition, from the use case founder Ivar Jacobson:

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. It stresses two attitudes during requirements analysis:

- Write requirements focusing on the users or actors of a system, asking about their goals and typical situations.
- Focus on understanding what the actor considers a valuable result.

Guideline 5: To Find Use Cases

Use cases are defined to satisfy the goals of the primary actors. Hence, the basic procedure is:

1.	Choose the system boundary. Is it just a software application, the hardware and application as a unit, that plus a person using it, or an entire organization?
2.	Identify the primary actors those that have goals fulfilled through using services of the system.
3.	Identify the goals for each primary actor.
4.	Define use cases that satisfy user goals; name them according to their goal. Usually, user-goal level use cases will be one-to-one with user goals, but there is at least one exception, as will be examined.

Step 1: Choose the System Boundary

The POS system itself is the system under design; everything outside of it is outside the system boundary, including the cashier, payment authorization service, and so on. For example, is the complete responsibility for payment authorization within the system boundary? No, there is an external payment authorization service actor.

Steps 2 and 3: Find Primary Actors and Goals

Guideline: Identify the primary actors first, as this sets up the framework for further investigation. The following questions helps to identify other actors .

Who starts and stops the system?	Who does system administration?
Who does user and security management?	Is "time" an actor because the system does something in response to a time event?
Is there a monitoring process that restarts the system if it fails?	Who evaluates system activity or performance?
How are software updates handled? Push or pull update?	Who evaluates logs? Are they remotely retrieved?
In addition to human primary actors, are there any external software or robotic systems that call upon services of the system?	Who gets notified when there are errors or failures?

Representing Goals of an Actor :

There are at least two approaches:

1. Draw them in a use case diagram, naming the goals as use cases.
2. Write an actor-goal list first, review and refine it, and then draw the use case diagram.

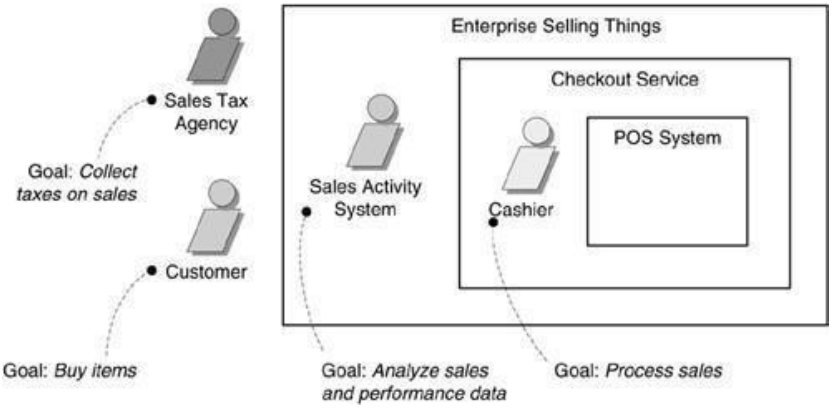
For example:

Actor	Goal		Actor	Goal
Cashier	process sales process rentals handle returns cash in cash out…		System Administrator	add users modify users delete users manage security manage system tables…
Manager	start up shut down…		Sales Activity System	analyze sales and performance data
…	…		…	…

Is the Cashier or Customer the Primary Actor?

The answer depends on the system boundary of the system under design, and who we are primarily designing the system for the viewpoint of the POS system the system services the goal of a trained cashier (and the store) to process the customer's sale.

Primary actors and goals at different system boundaries.



The customer is an actor, but in the context of the NextGen POS, not a primary actor; rather, the cashier is the primary actor because the system is being designed to primarily serve the trained cashier's "power user" goals. The system does not have a UI and functionality that could equally be used by the customer or cashier. Rather, it is optimized to meet the needs and training of a cashier.

Second Method to Find Actors and Goals - Event Analysis

Another approach to aid in finding actors, goals, and use cases is to identify external events. What are they, where from, and why? Often, a group of events belong to the same use case. For 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

In general, define one use case for each user goal. Name the use case similar to the user goal for example,

Goal: process a sale; Use Case: Process Sale.

Start the name of use cases with a verb. A common exception to one use case per goal is to collapse CRUD (create, retrieve, update, delete) separate goals into one CRUD use case, idiomatically called Manage <X>. For example, the goals "edit user," "delete user," and so forth are all satisfied by the Manage Users use case.

Guideline 6: Tests To Find Useful Use Cases

There are several rules of thumb, including:

- The Boss Test

- The EBP Test
- The Size Test

The Boss Test : To check for achieving results of measurable value

Your boss asks, "What have you been doing all day?" You reply: "Logging in!" Is your boss happy?. If not, the use case fails the Boss Test, which implies it is not strongly related to achieving results of measurable value. It may be a use case at some low goal level, but not the desirable level of focus for requirements analysis.

The EBP Test

- An Elementary Business Process (EBP) is a term from the business process engineering field, defined as:

EBP is similar to the term user task in usability engineering, although the meaning is less strict in that domain. Focus on use cases that reflect EBPs.

- 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 consistent state, e.g., Approve Credit or Price Order
- The EBP Test is similar to the Boss Test, especially in terms of the measurable business value qualification.

The Size Test

A use case typically contains many steps, and in the fully dressed format will often require 3- 10 pages of text. A common mistake in use case modeling is to define just a single step within a series of related steps as a use case by itself, such as defining a use case called Enter an Item ID. You can see a hint of the error by its small size the use case name will wrongly suggest just one step within a larger series of steps, and if you imagine the length of its fully dressed text, it would be extremely large.

Example: Applying the Tests

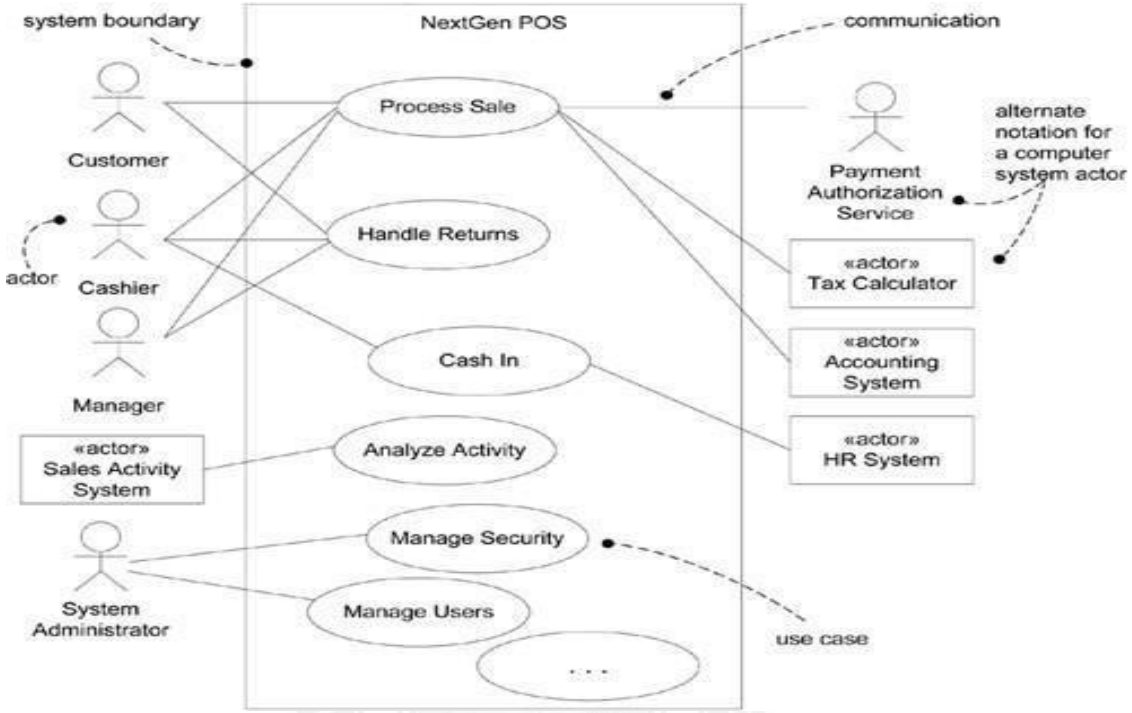
- Negotiate a Supplier Contract
  - Much broader and longer than an EBP. Could be modeled as a business use case, rather than a system use case.
- Handle Returns
  - OK with the boss. Seems like an EBP. Size is good.
- Log In
  - Boss not happy if this is all you do all day!
- Move Piece on Game Board
  - Single step fails the size test.

Applying UML: Use Case Diagrams

- The UML provides use case diagram notation to illustrate the names of use cases and actors, and the relationships between them

- Use case diagrams and use case relationships are secondary in use case work. Use cases are text documents. Doing use case work means to write text.

Partial use case Diagram - context diagram.



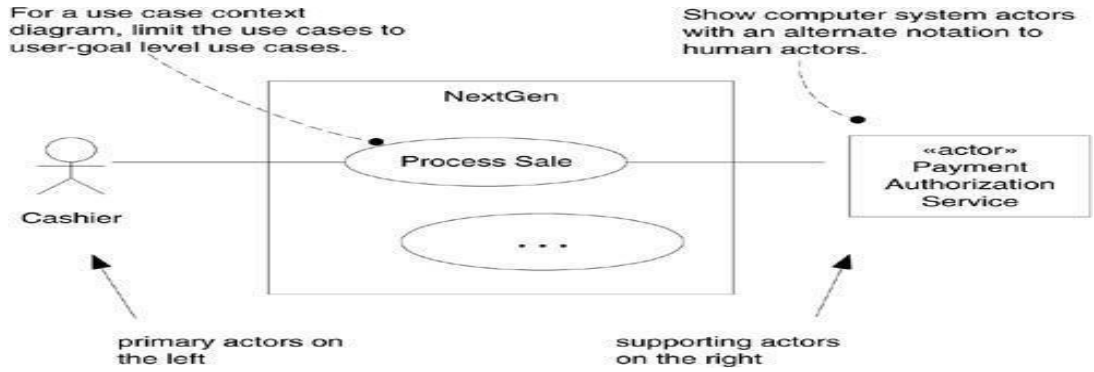
Guideline

- Use case diagram is an excellent picture of the system context
- It makes a good context diagram that is, showing the boundary of a system, what lies outside of it, and how it gets used.
- It serves as a communication tool that summarizes the behavior of a system and its actors.

Guideline: Diagramming

Notice the actor box with the symbol «actor». This style is used for UML keywords and stereotypes, and includes guillemet symbols special single-character brackets («actor», not <<actor>>)

Notation suggestions.



Alternate actor notation.