Object-Oriented Software Analysis and Design

School of Computer Science
University of Windsor

Use Cases

- Use cases are **text stories**, widely used to discover and record requirements.

  Here is an example of a brief format use case:

  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.

Use Cases (contd.)

- The essence of use cases is **discovering and recording functional requirements** by writing stories of using a system to fulfill user goals; that is, cases of use.
- **Use cases are not diagrams, they are text.**
- UML use case diagram is only secondary.
- Use cases are text stories of some actor using a system to meet goals.

Sample UP artifact influence

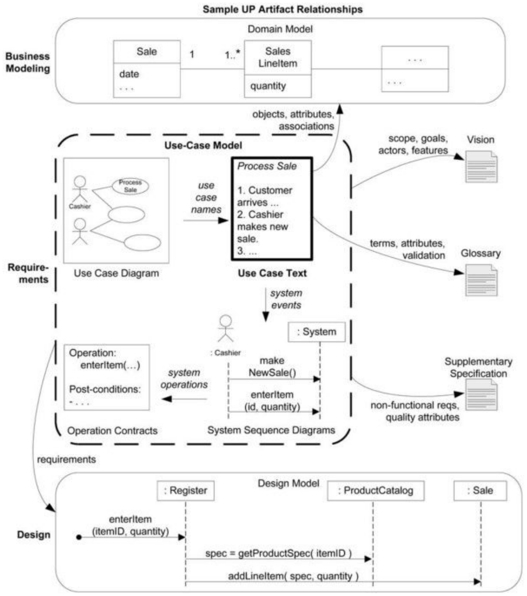


Figure: *Sample UP artifact influence.*

Use Cases and the Use-Case Model

- The **Use-Case Model** is the set of all written use cases; it is a model of the system's functionality and environment.
- Use cases are text documents, not diagrams, and use-case modeling is primarily an act of writing text, not drawing diagrams.

Use Cases and the Use-Case Model (contd.)

- Besides use cases, there are other artifacts in requirements, such as, 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.
- There is nothing object-oriented about use cases; we're not doing OO analysis when writing them. That's not a problem use cases are broadly applicable, which increases their usefulness. That said, use cases are a key requirements input to classic OOA/D.

Why Use Cases?

- Easy for customers to contribute to the project
- Lower the risk of project failure.
- Emphasize the user goals and perspective.
- Ability to scale both up and down in terms of sophistication and formality.

Use Cases vs. Functional Requirements

- Use cases are mostly functional or behavioural requirements which indicates what the system will do.
- Another view to look at use cases is that a use case defined a contract of how a system will behave.
- Use cases are indeed requirements (although not all requirements).
- A **key idea** of use cases is to (usually) reduce the importance or use of detailed old-style feature lists and rather write use cases for the functional requirements.

Use Cases: Definitions and Notations

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.

What are Actors, Scenarios, and Use Cases? (contd.)

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

  For example, here is a casual format use case with alternate scenarios:

  - **Handle Returns**
  - **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 (perhaps it is corrupted).
    - If the system detects failure to communicate with the external accounting system

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.
- 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
  2. Supporting actor
  3. Offstage actor
- Primary and supporting actors will appear in the action steps of the use case text.

Three Kinds of Actors: 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.

Three Kinds of Actors: 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.

Three Kinds of Actors: 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

- Use cases can be written in different formats and levels of formality.

Three common use case formats: Brief

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

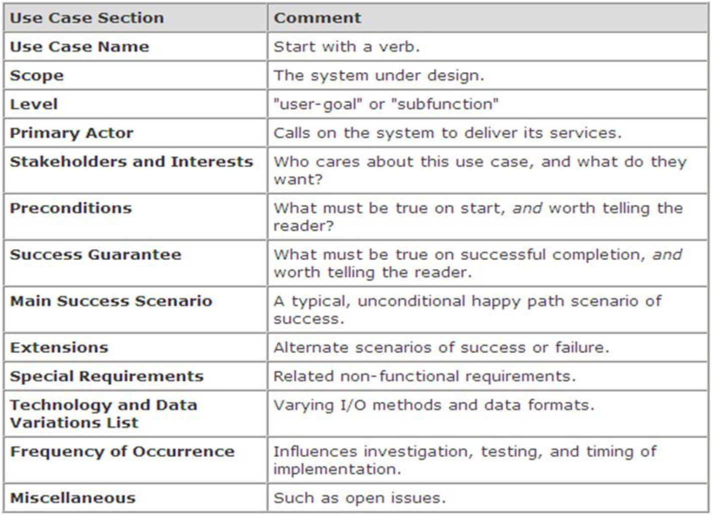
Three common use case formats: Causal

- Informal paragraph format. Multiple paragraphs that cover various scenarios. The prior Handle Returns example was casual.
- When?
  - During early requirements analysis, to get a quick sense of subject and scope. May take only a few minutes to create.

Three common use case formats: Fully Dressed

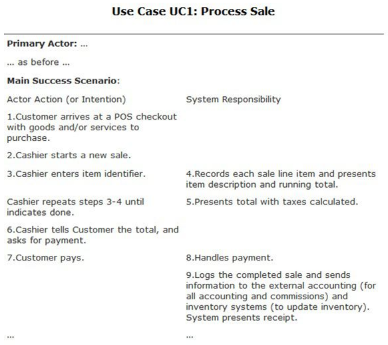
- All steps and variations are written in detail, and there are supporting sections, such as preconditions and success guarantees.
- When?
  - After many use cases have been identified and written in a brief format, then during the first requirements workshop a few (such as 10%) of the architecturally significant and high-value use cases are written in detail.
- Example uploaded separately in the BB

The Template of Fully Dressed Format



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.

Notation: Are There Other Formats? A Two-Column Variation



Use Case UC1: 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 steps 3-4 until indicates done.	5.Presents total with taxes calculated.
6.Cashier tells Customer the total, and asks for payment.	
7.Customer pays.	8.Handles payment.
	9.Logs the completed sale and sends information to the external accounting (for all accounting and commissions) and inventory systems (to update inventory). System presents receipt.
...	...

- One column or two column? Which one is better?
- It is up to you!
- How would you write your own use cases?
- Follow the guidelines. . .

Use Cases: Guidelines

Guideline: Write in an essential UI-free style

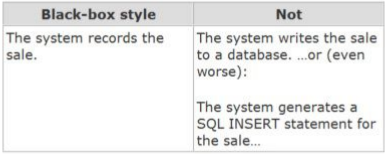
- Write use cases in an essential style; keep the user interface out and focus on actor intent.

Guideline: Write Terse Use Cases

- Do you like to read lots of requirements? I didn't think so. So, write terse use cases. Delete "noise" words. Even small changes add up, such as "System authenticates. . . " rather than "The System authenticates. . . "

Guideline: 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.



Black-box style	Not
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: Take an Actor and Actor-Goal Perspective

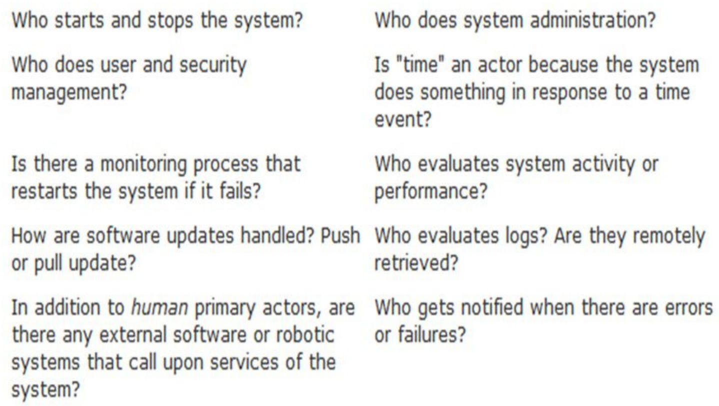
- Write requirements focusing on the users or actors of a system, asking about their goals and typical situations.
- Focus on understanding what the actor considers a valuable result.

Guideline: How to Find Use Cases

1. Choose the system boundary.
2. Identify the primary actors
3. Identify the goals for each primary actor.
4. Define use cases that satisfy user goals; name them according to their goal.
   - Start the name of use cases with a verb.

Are There Questions to Help Find Actors and Goals?

- In addition to obvious primary actors and goals, the following questions help identify others that may be missed:

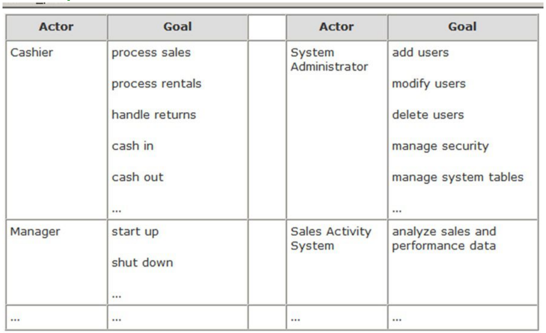


Who starts and stops the system?	Who does system administration?
Who does user and security management?	Is "time" an actor because the system does something in response to a time event?
Is there a monitoring process that restarts the system if it fails?	Who evaluates system activity or performance?
How 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?

How to Organize the Actors and Goals?

- Two approaches:
  - As you discover the results, draw them in a use case diagram, naming the goals as use cases.
  - Write an actor-goal list first, review and refine it, and then draw the use case diagram.

For example,

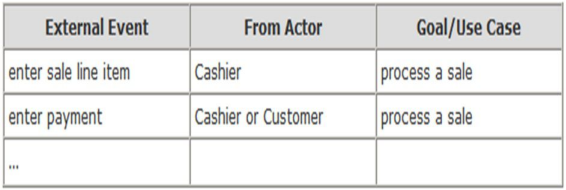


Actor	Goal		Actor	Goal
Cashier	process sales process rentals handle returns cash in cash out ...		System Administrator	add users modify users delete users manage security manage system tables ...
Manager	start up shut down ...		Sales Activity System	analyze sales and performance data
...	...		...	...

Other Ways 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
...		

Guideline: What Tests Can Help Find Useful Use Cases?

- In general, “What is a valid use case?”, a more practical question is: “What is a useful level to express use cases for application requirements analysis?” There are several rules of thumb, including:
  - The Boss Test
  - The EBP Test
  - The Size Test

The Boss Test

- 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.
- That doesn’t mean to always ignore boss-test-failing use cases. User authentication may fail the boss test, but may be important and difficult.

The EBP Test

- An Elementary Business Process (EBP) is a term from the business process engineering field, 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 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.
- Focus on use cases that reflect EBPs.

The Size Test

- A use case is very seldom a single action or step; rather, a use case typically contains many steps, and in the fully dressed format will often require 3-10 pages of text. (7 pages of the sample use case we saw)
- 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 short.

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.

Reasonable Violations of the Tests

- Although the majority of use cases identified and analyzed for an application should satisfy the tests, exceptions are common.
- It is sometimes useful to write separate subfunction-level use cases representing subtasks or steps within a regular EBP-level use case.

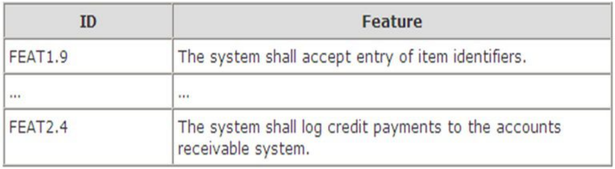
Example: Applying the Tests

  - a subtask or extension such as “paying by credit” may be repeated in several base use cases.

- If so, it is desirable to separate this into its own use case, even though it does not really satisfy the EBP and size tests, and link it to several base use cases, to avoid duplication of the text.

The benefits of Use Cases

- A motivation for use cases is focusing on who the key actors are, their goals, and common tasks. Plus, in essence, use cases are a simple, widely-understood form (a story or scenario form).
- Another motivation is to replace detailed, low-level function lists (as shown below, which were common in 1970s traditional requirements methods) with use cases.



ID	Feature
FEAT1.9	The system shall accept entry of item identifiers.
...	...
FEAT2.4	The system shall log credit payments to the accounts receivable system.

In contrast, use cases organize a set of requirements in the context of the typical scenarios of using a system.

The benefits of Use Case (contd.)

- Is the feature list really bad? Abandon it?
- High-Level System Feature Lists Are Acceptable. ✓
- Detailed function lists are undesirable, a terse, high-level feature list, called **system features**, added to a Vision document can usefully summarize system functionality.

The benefits of Use Case (contd.)

- In contrast to 50 pages of low-level features, a system features list includes only a few dozen items. It provides a succinct summary of functionality, independent of the use case view.

  For example,

  - sales capture
  - payment authorization (credit, debit, check)
  - system administration for users, security, code and constants tables, and so on
  - ...

When Are Detailed Feature Lists Appropriate Rather than Use Cases?

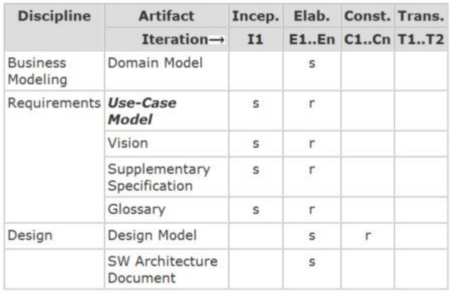
- Sometimes use cases do not really fit; some applications cry out for a feature-driven viewpoint.

  For example,

  application servers, database products, and other middleware or back-end systems need to be primarily considered and evolved in terms of features (“We need Web Services support in the next release”).
- Use cases are not a natural fit for these applications or the way they need to evolve in terms of market forces.

When Should Various UP Artifact (Including Use Cases) be Created?

Table 6.2. Sample UP artifacts and timing. s - start; r - refine



Discipline	Artifact	Incep.	Elab.	Const.	Trans.
	Iteration→	I1	E1..En	C1..Cn	T1..T2
Business Modeling	Domain Model		s		
Requirements	***Use-Case Model***	s	r		
	Vision	s	r		
	Supplementary Specification	s	r		
	Glossary	s	r		
Design	Design Model		s	r	
	SW Architecture Document		s		

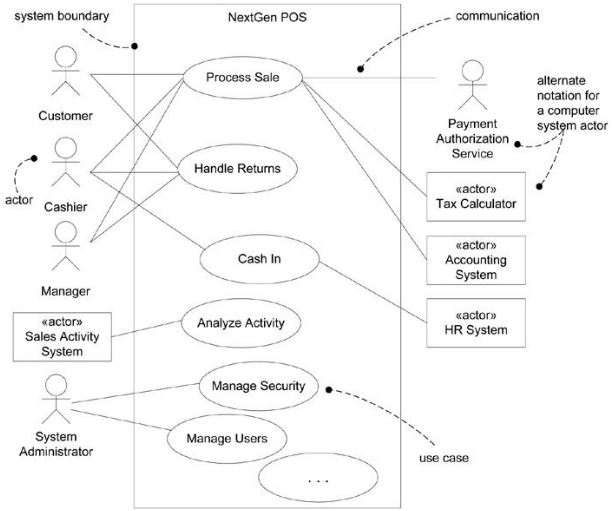
How to Work With Use Cases in Iterative Methods?

- The Use-Case Model is started in inception, with perhaps only 10% of the architecturally significant use cases written in any detail.
- The majority are incrementally written over the iterations of the elaboration phase.
- By the end of elaboration, a large body of detailed use cases and other requirements (in the Supplementary Specification) are written, providing a realistic basis for estimation through to the end of the project.

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.
- A simple use case diagram provides a succinct visual context diagram for the system, illustrating the external actors and how they use the system.
- **But remember that, Use cases are text documents. Doing use case work means to write text.**
- Use case diagrams and use case relationships are **secondary** in use case work.

Use Case Diagrams (contd.)



Use Case Diagrams (contd.)

- A 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.

Guidelines: Use Case Diagrams

- Guideline 1:
  - Draw a simple use case diagram in conjunction with an actor-goal list.
- Guideline 2:
  - See the diagram for notation suggestions
- Guideline 3:
  - Downplay Diagramming, Keep it Short and Simple.

To reiterate, the important use case work is to write text, not diagram or focus on use case relationships.

Use Case Diagrams: Notations

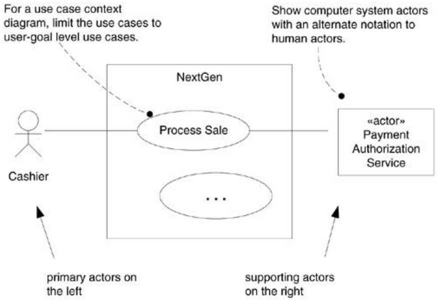

For a use case context diagram, limit the use cases to user-goal level use cases.
Show computer system actors with an alternate notation to human actors.
NextGen
Process Sale
...
Cashier
«actor» Payment Authorization Service
primary actors on the left
supporting actors on the right

Figure: *Notation suggestions.*

Use Case Diagrams: Notations

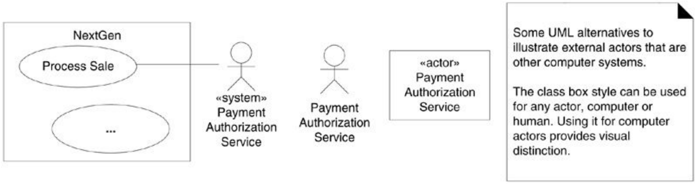
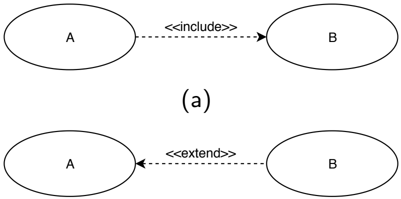

NextGen
Process Sale
...
«system»
Payment
Authorization
Service
Payment
Authorization
Service
«actor»
Payment
Authorization
Service
Some UML alternatives to illustrate external actors that are other computer systems.
The class box style can be used for any actor, computer or human. Using it for computer actors provides visual distinction.

Figure: *Alternate actor notation.*

Use Case Diagrams: Relationships

- An association is a relation between an actor and a use case.
- Actors may be related by generalization relations.
- A use case may depend on another use case to achieve its goal. An “include” dependency occurs when a use case always uses another one.



(a)

(b)

Figure: (a)Use cases and include dependencies (b)Use cases and extend dependencies

- A use case may also call another use case but only under special circumstances, for instance, as alternatives. In that

Use Case Diagrams: An Example

Consider that you are analyzing requirements for a housing system run by the campus housing service.

The campus housing service helps students find apartments. Apartment owners complete information forms about the available rental units (e.g., location, number of bedrooms, monthly rent), which are then entered into a database. Students can search this database via the Web to find apartments that meet their needs (e.g., a two-bedroom apartment for $400 or less per month within a half mile of campus) and contact the apartment owners directly to see the apartment and possibly rent it. Apartment owners call the service to delete their listing when they have rented their apartment(s).

Use Case Diagrams: An Example (contd.)

1. Identify the actors and major use cases for the following high level business processes in a housing system run by the campus housing service.
2. Draw a use case diagram for the system, considering all of the actors and use cases you identified in Step 1.

It's Quiz Time

1. A use case is a collection of related success and failure scenarios that describe an actor using a system to support a goal. (True or False)
2. Start the name of use cases with a noun. (True or False)
3. Primary and . . . . actors will appear in the action steps of the use case text.