



www.primaryview.org

from the notebooks of John Artim

Use Cases and Software Requirements Specification

The Basics of Use Case Authoring

John M. Artim, ©2003, 2004. Permission to copy, modify, or distribute is granted.

Acknowledgements

- Initial content in this notebook was developed in collaboration with Chuck Walrad of Davenport Consulting, Inc (www.daven.com). Any errors or misrepresentations are solely the responsibility of John Artim.
- Feel free to use this material, in whole or part, for your own training needs but please indicate that you based your work, with permission, on a copyrighted work by John M. Artim that is available from www.primaryview.org.
- A Microsoft® PowerPoint file of this notebook is available by request. Send an email to John Artim (jartim@uistyle.com) requesting the Use Case Basics Notebook PowerPoint file in the Use Cases and Software Requirements Specification series. Please include your name, affiliation, and your intended use of the material.

A Note About Diagrams and UML

- Some approaches to use cases are strongly text-based, others mix text and diagrams
- This approach introduces both text and diagrams representing a use case model

- This is done to help you learn use case methods
- It is not an assertion that either approach is better

Note: It is always more important that participants in a project agree on approach and notation rather than worry the specifics of either

- Notation in this notebook complies with UML
 - UML is the current standard of the software industry
 - Any variation from standard UML is explicitly noted

Goals of this Notebook

- Teach the basics of use case authoring for functional specification. The concepts of use case, actor, and scenario are defined. The structure of a use case scenario is outlined and an approach to authoring scenarios based on this structure is presented.
- Introduce the notion of non-functional requirements.

Browsing time for this notebook is about 3 hours.



Background for this Notebook

- You do not need any specific background material to benefit from the material in this notebook.
- Any prior experience you have in the process of software specification—either as a producer or consumer of specifications—will help you understand this material.

The Problem to Solve

- Understanding and documenting a proposed system is difficult
 - Slicing up behavior into pieces is often arbitrary
 - If you build using arbitrary slices, then the end-user must recombine meaningless little bits into something useful
 - This kind of usability problem is costly to the business
- For the business buying a proposed system it may be difficult to understand how new system function relates to their business
 - The tie between business processes and system functionality is often poorly understood or poorly communicated
 - This leads to “solutions” that do not fit in with the business’ processes and which provide little obvious value

Searching for the Solution

- **Structured Programming** (Late '60's through the '80s)
 - Provided some guidance for specifying a new system
 - But functional decomposition was too arbitrary
 - Some individual practitioners did a good job and this approach was generally better than the chaos that preceded it
 - But this was not enough
- **Object Technology** (Late '80's to the present)
 - Offers a good tie between customer terminology and software specification and design
 - But not between customer business processes and specification and design

Current Practice

- Use Cases (early '90's to the present)
 - Guides the analyst in breaking down the structure of customer processes and work
 - Ties into other object-technology practices, especially object modeling
 - Is now an industry standard
 - Unified Modeling Language (UML)
 - See www.omg.org
 - Atlantic Systems Guild
 - Requirements Specification template:
<http://www.volere.co.uk/rst2.htm>

Basic Terminology: Work, Task, Process, Model?

	Encyclopedia Britannica, © 2004	
Work	a sustained physical or mental effort to overcome obstacles and achieve an objective or result	Effort + Goal
Task	a usually assigned piece of work often to be finished within a certain time	Work + Actor + Time
Process	a series of actions or operations conducing to an end	Procedure + Goal
Model	a usually miniature representation of something; also, a pattern of something to be made	World in Miniature

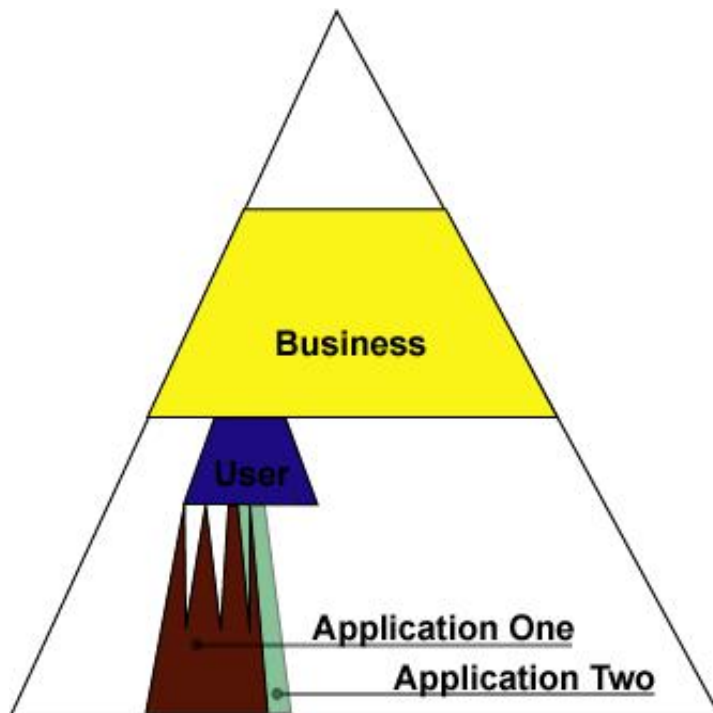
Task Modeling

A model of real-world work including:

- **How**
 - The steps required to complete the work
- **Who**
 - The actor(s) who perform the work
- **When**
 - The trigger for the work
 - An event
 - The passing of time
- **What**
 - The goal of the work
 - What motivates the actor
 - The state of the world to begin some work—what “stuff” must be available
 - The state of the world when done—what “stuff” must be around when the task is done

The Task Pyramid

- The Task Pyramid represents a System-in-the-Large including:
 - **Business**—One Business' Processes
 - **User**—A Representative User's Tasks
 - **Application One**—A representative application whose function is scattershot relative to user tasks—UI issues here!
 - **Application Two**—A representative application whose function clusters with user tasks



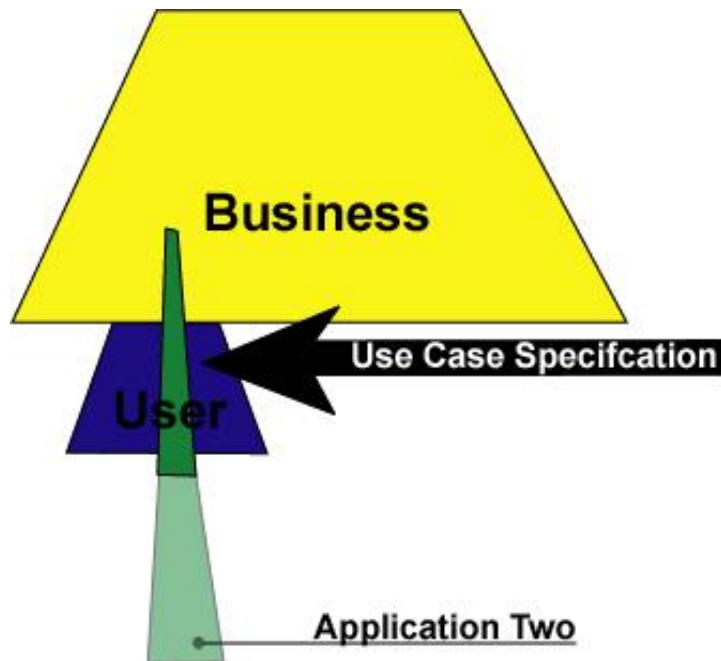
The Basics of Use Case Authoring

www.primaryview.org

from the notebooks of John Artim

Where Do Use Cases Fit?

- When used to specify an application's function, Use Cases describe:
 - The User Tasks the application must support and how it must support them.
 - The Business context into which these User Tasks fit including Business goals.
 - Just enough of the application to unambiguously define it *but not design it!*



What is a Use Case?

- A use case describes a chunk of work
 - Who is the actor who does the work?
 - What does the actor want to accomplish by completing the work?
 - What is their goal?
 - Goals focus use cases on the user's view—not the designer's
- User work is made concrete by scenarios
 - Each scenario describing one concrete way to complete the work
 - Each use case is defined by one or more scenarios

Why is “Concrete” Good?

- The world—concrete reality—is in front of everyone
 - Total agreement on concrete reality doesn’t happen
 - But consensus does—that’s what dictionaries are all about!
- Concrete descriptions keep requirements grounded in reality so that:
 - Discussions among stakeholders are productive
 - Ambiguity in discussions can be found and *stomped out!*

Why Use Cases?

- Scenarios concretely describe one task
 - Scenario structure helps simplify and find ambiguity
- Use cases group together the scenarios that describe alternatives to accomplishing the same piece of work
 - A *use case* defines the many ways to do one *task*
- The structure of use cases helps us understand how work is partitioned within the system:
 - Between human and machine
 - Among humans and across work groups

What is a System?

- A system is a group of humans, machines, applications, communication infrastructure, organizational processes, and individual tasks that collectively accomplishes a set of goals.
 - The system as a whole is *never* just an application

What is a Requirements Specification?

- A requirements specification defines the responsibilities of a part of a system that is to be constructed
 - In software, this is typically an application
 - Or a system that includes one or more applications and some hardware
- But this *system under discussion* (or SUD—a key term!) is only a part of the larger system
- This dual use of *system* is confusing, but the larger meaning of system is critical!

What is in a Requirements Specification?

- Functional Requirements
 - A description of expectations of what a system under discussion (SUD) must do
 - Often provided by a use case model
- Non-Functional Requirements
 - Look-and-Feel, UI, and Usability Requirements
 - Performance Requirements
 - Security Requirements

How are Requirements Specifications Used?

- Marketing Requirements Document
 - Produced by Marketing Organization
 - Engineering responds with Product Specification
 - Functional Specification often via Use Cases
- Internal Requirements Specification
 - Produced by Client Organization or Engineering
- Request for Proposal Response
 - Client Company Supplies Request for Proposal
 - Including Requirements Table
 - Vendor Company Supplies Proposal
 - Often including Functional Specification by Use Cases
 - Use Cases should map to Customer Requirements

What is a Use Case Model?

- A use case model specifies the System Under Discussion's (SUD) responsibilities
 - And divides the SUD from the rest of the system
 - Users, legacy systems, and everything else
- A use case model includes:
 - All of the actors who interact with the SUD
 - All of the use cases defining the work done by these actors using the SUD
 - All scenarios defining each use case
 - Plus additional use cases and scenarios defining the context the SUD operates in

What is a Use Case Model?

(cont'd)

- As a requirements specification, a use case model sketches, in miniature, the tasks of a system
 - Showing the division between an SUD and the rest of the system
 - Miniature means without excessive detail
 - This sketch *does not* show the construction of the system
 - It does show the appearance of the SUD to the rest of the system
 - That is, the SUD's interfaces!
 - And how these interfaces are used

The System Under Discussion

- Use cases provide a separation of description of what must be built (the SUD) and the context it operates in (the system as a whole)
- Use case authors accomplish this by:
 - Writing all use cases from the perspective of the human or other non-SUD actor
 - Writing scenarios so that they show how the SUD responds to activities by non-SUD actors

Non-Functional Requirements

- Look-and-Feel, UI, and Usability Requirements
 - See *Use Case Authoring and User Interface*, in this series
- Performance Requirements reflect the needs of humans and systems external to the SUD
 - That is, how fast the SUD must respond to satisfy the Actors the SUD will interface with
- Security Requirements reflect
 - Business requirements
 - Interface requirements of systems external to the SUD.
- For additional categories see the Atlantic Systems Guild Requirements Specification template:
 - <http://www.volere.co.uk/rst2.htm>



The Parts of a Use Case Model

- Actors
- Use Cases
- Scenarios
- Relationships between Actors and Use Cases
- Relationships among Use Cases

Actors

- An actor is an agent who performs one or more use cases.
- Actors can be:
 - A human
 - An organization (work group)
 - An automated agent

Human Actors

- Humans shown in a use case model are categorized into groups
 - Each Actor represents a Job Title or Role
 - Individual actors are rarely documented
- Here are some examples of human Actors:
 - Customer Service Representative
 - Warehouse Dock Crew
 - Transportation Planner
 - Benefits Coordinator
 - Programmer

Organizations

- Use cases often describe the work done by a group of human actors
- Examples of business organizations include:
 - Department
 - Division
 - Subsidiary
 - Vendor

Automated Agents

- An agent is a non-human agent that autonomously performs the tasks assigned to it
- In a world increasingly dominated by interconnected systems, automated agents must be considered when describing a system's actors
- Examples include:
 - Web Crawler
 - EDI Server
 - FAX Server

Use Cases

- A *use case* represents a chunk of work performed by a human, an organization, or an automated agent
 - The chunk of work stands on its own—that is, successful completion of the work satisfies a goal recognizable to the actor
 - But no chunk of work makes sense by itself—all work fits into a context
 - This context is the web of goals and work products that satisfy higher ultimate goals

Use Case—Create Order

A *Customer Service Representative* works with *Customers* over phone and fax to create an *Order* for the *Customer* and initiate completion of the *Order* for the *Customer*.

- *Create Order* is a Use Case in this example

Scenarios

- A Scenario is one concrete example showing how to complete a use case
 - Some represent examples of how a use case can be successfully completed, others represent examples of failure
 - Scenarios are a narrative description of one possible set of steps that might be used in an attempt at completing a chunk of work

Scenario—Create Order from Phone Call

- A *Customer Service Representative* answers a *phone call* and successfully creates an *Order*
- Example Scenarios include:
 - Create Order based on Phone Call, Normal Course
 - Create Order based on fax, Normal Course
 - Create Order based on Phone Call, Discontinued Item

Scenario Content—Narrative Steps

- The heart of a scenario is a narrative description of a concrete set of steps used to attempt completion of the use case
 - The text to the right show the narrative steps for the *Normal Course Scenario* for the *Identify Customer Use Case*
1. Ask the customer their name—first name then family name—and enter it into the *Find Customer* interface.
 2. Ask the customer their street address, city, state, and postal code and enter this into the *Find Customer* interface.
 3. Activate the search. A single *Customer* object, which is selected, is displayed in a list in the *Find Customer* interface.
 4. Verify information with customer.
 5. Activate *Create Order* command on *Find Customer* interface.

Step Sequencing

- Where possible, show steps in a typical sequence
- Indicate when this sequence, or part of the sequence, is fixed in its order
 - Shown by the marking: {ordered}
- Indicate when an individual step is optional
 - Shown by the marking: {optional}
- Also note when a step represents an entire «*included*» or «*used*» use case
 - Indicate with the step narrative wording, as with, “This step is defined by the included use case, XXX.”

Scenario Content—Brief Description

- The *Brief Description* is an operational definition for the scenario
 - A one to three sentence definition
 - Describing what the actor is trying to accomplish
 - And what makes this scenario different from the same use case's other scenarios

Quickly identify a customer who is currently with you by telephone. The identification is based on minimal information to avoid taking too much of the customer's time.

Sidebar: What is an Operational Definition?

- An operational definition concisely defines a term by characterizing the functional use of that term. Operational definitions focus on prototypical usage or usage in practice. Operational definitions should be concise—ordinarily be no more than one to three sentences in length.

Goal

- Goal: a short description of what the actor is trying to accomplish
 - Describe the actor's *motivation!*
 - Length: one to three sentences
- The goal helps the modeler understand what is driving the behavior of the actor
 - Think of the goal as an attraction pulling the actor
- Often called, Goal-in-Context

- Identify the customer to the system so that it is possible to make use of the system to quickly serve the customer's requests.



Use Case Content—Interactions

- This information describes how one use case relates to other use cases.
- Preconditions
- Postconditions
- Trigger(s)

Preconditions

- Preconditions describe what must be true of the state of the world before a use case/scenario begins
 - The state of the world typically is expressed as a list of noun phrases and state information
 - The result of one use case's completion is the state-of-the-world for the next one
- The customer is known to the SUD.
- Adequate stock exists to fill the Order.

Postconditions

- Postconditions describe the state of the world after completion of the scenario
 - The state of the world is expressed as a list of noun phrases and state information

- The customer of current interest is identified to the SUD.

Trigger(s)

- The Trigger is the event or events that initiate the start of a use case.
 - Time
 - Absolute Time
 - Passage of Time
 - Event in the World
 - Completion of a Prior Use Case
- This use case is triggered by the containing use case.
 - This use case is triggered by receipt of a *Shipping Instructions* document.
 - This use case is triggered by the RFID tag on a container passing within range of a roadside tag reader.



Use Case Content—Usability and Project Planning

- This information helps guide usability and general project-planning activities
- Frequency
- Priority

Frequency

- Describes how often a use case is performed by the actor
- Typically one of *high*, *medium*, or *low*
 - But can be *monthly*, *weekly*, *daily*, *hourly*, or *continuously* or whatever else is appropriate
 - *Provide a working definition of each level*
 - Be consistent throughout all use cases in the model!

- **High**
 - That is, *Identify Customer* occurs often.

Priority

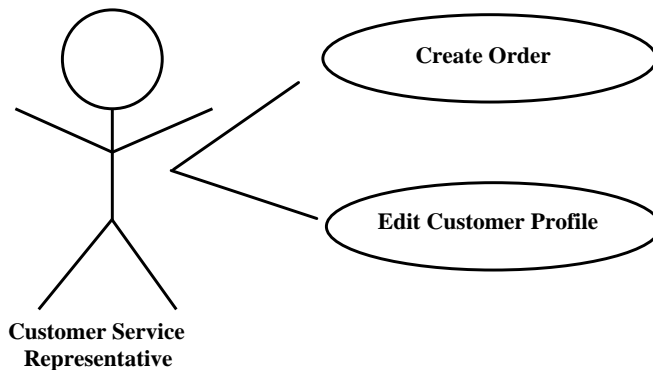
- Describes the importance of the use case
 - Typically critical, normal, or background
 - Or critical, immediate, or background
 - Or high, medium or low
- This separates the most important from the more mundane use cases
 - Remember, this is stated in terms of what is important in the user's world!

- **For example, in a large call center, *Immediate* might be defined as:**
 - Never put a customer on hold or hang up on them
 - But leave open the possibility that other use cases, if in progress, may supercede answering the phone

Relationships Between Actors and Use Cases

- The *Performs* relationship indicates that an Actor performs a Use Case

Performs Relationship in a Use Case Diagram

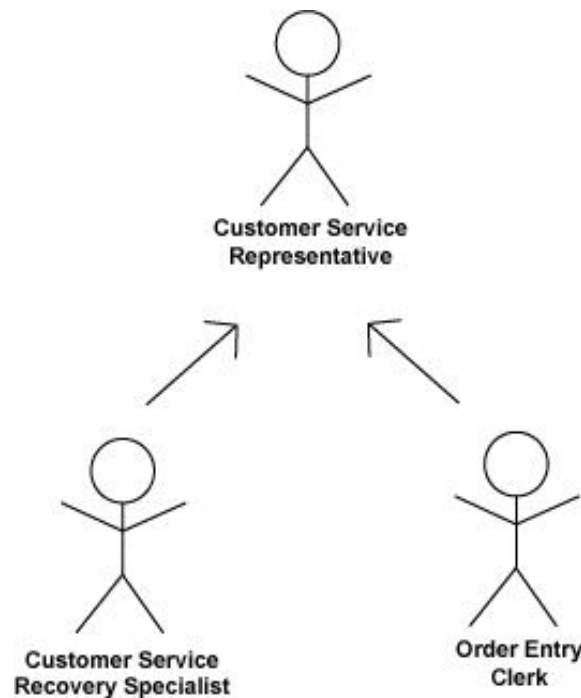


- This diagram depicts an Actor:
 - Customer Service Representative
- And two Use Cases (tasks):
 - Create Order
 - Edit Customer Profile
- And two Performs relationships between the Actor and the Use Cases indicating that the Actor *performs* both Use Cases

Relationships Among Actors

- A Generalization relationship between Actors indicates that one Actor is a type of another Actor
 - The specific Actors are Concrete Actors
 - The names of these actors should be recognizable to Subject Matter Experts (SMEs) as job titles or the equivalent
 - The more generic Actors are Abstract Actors (or Roles)
 - These describe a common bundle of job responsibilities
 - Where possible, these names should also be recognizable at least as sensible bundles of responsibility
 - They often materialize as a security grouping in final design

Actor Generalization in a Use Case Diagram



- This diagram shows *Customer Service Representative* as a Role
- And shows *Order Entry Clerk* and *Customer Service Recovery Specialist* as two Concrete Actors who can play the Role of *Customer Service Representative*



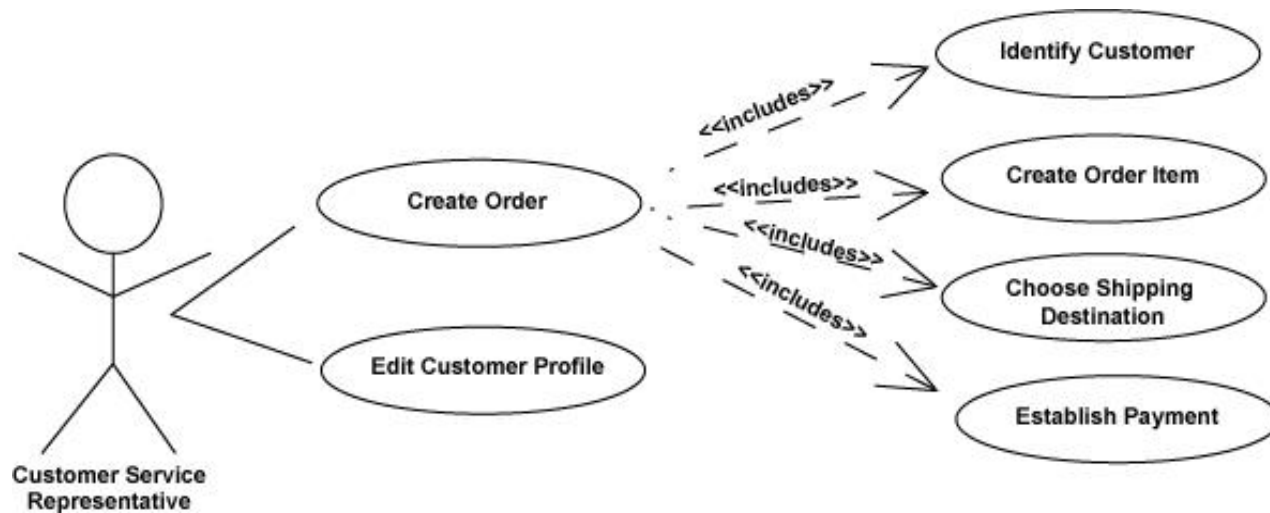
Relationships Among Use Cases

- Includes
- Generalization
- Extends

Use Case *Includes* Relationship

- This relationship shows that one use case acts as a step in another use case.
 - That is, one use case provides a detailed description of a small part of another use case
 - This relationship is depicted by a directed arc between two use cases and labeled with *«include»*

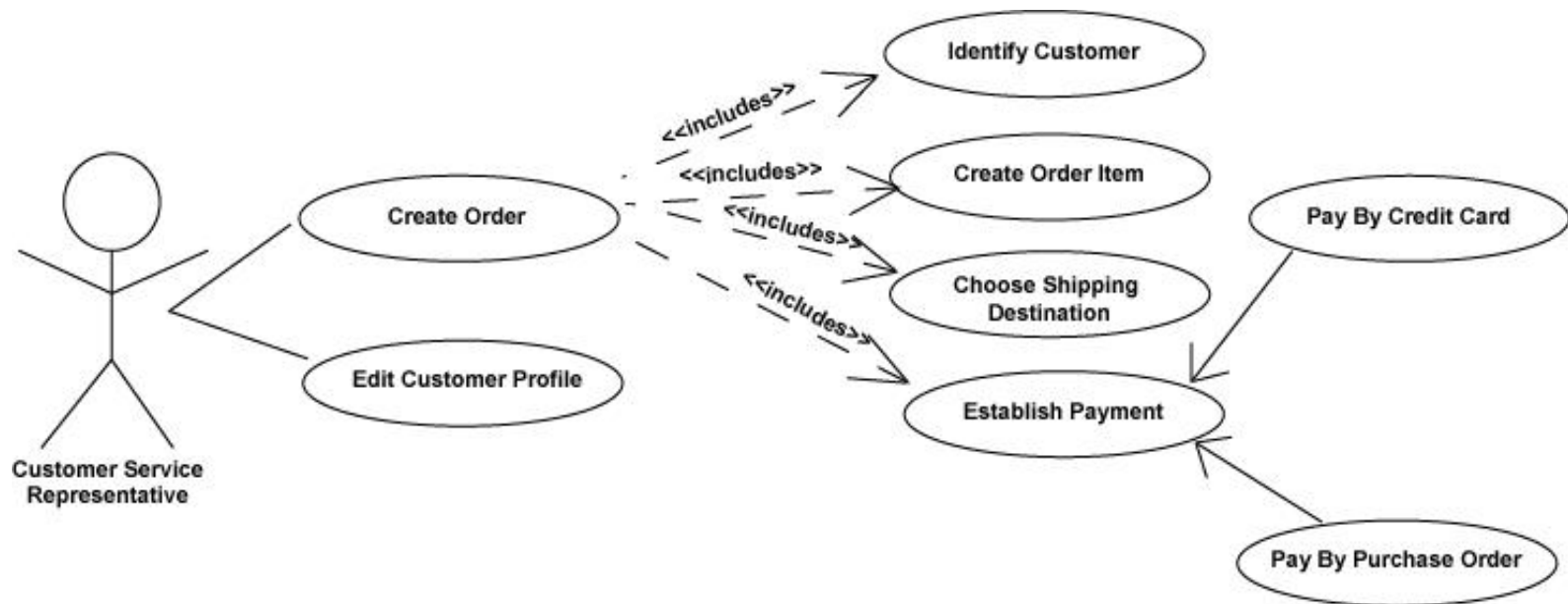
Includes Relationship in a Use Case Diagram



Use Case Generalization Relationship

- This relationship indicates that one use case defines a generalized task and another use case defines a specialized version of the task

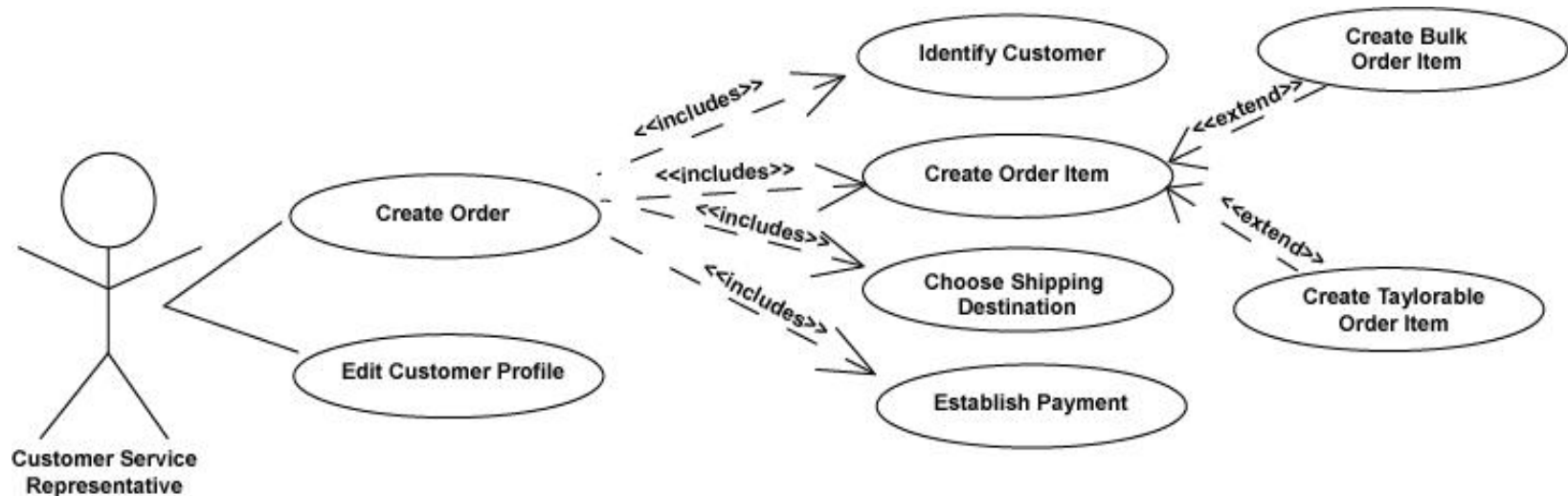
Generalization Relationship in a Use Case Diagram



Use Case Extension Relationship

- Used where one use case *extends* another by replacing one or more procedural steps
 - Offers an alternative way to accomplish a similar use case
 - Can involve ambiguity in just what the use case is accomplishing
 - Marked by a link with a dashed body and the marking, «extends»

Extends Relationship in a Use Case Diagram



Extension Scenario

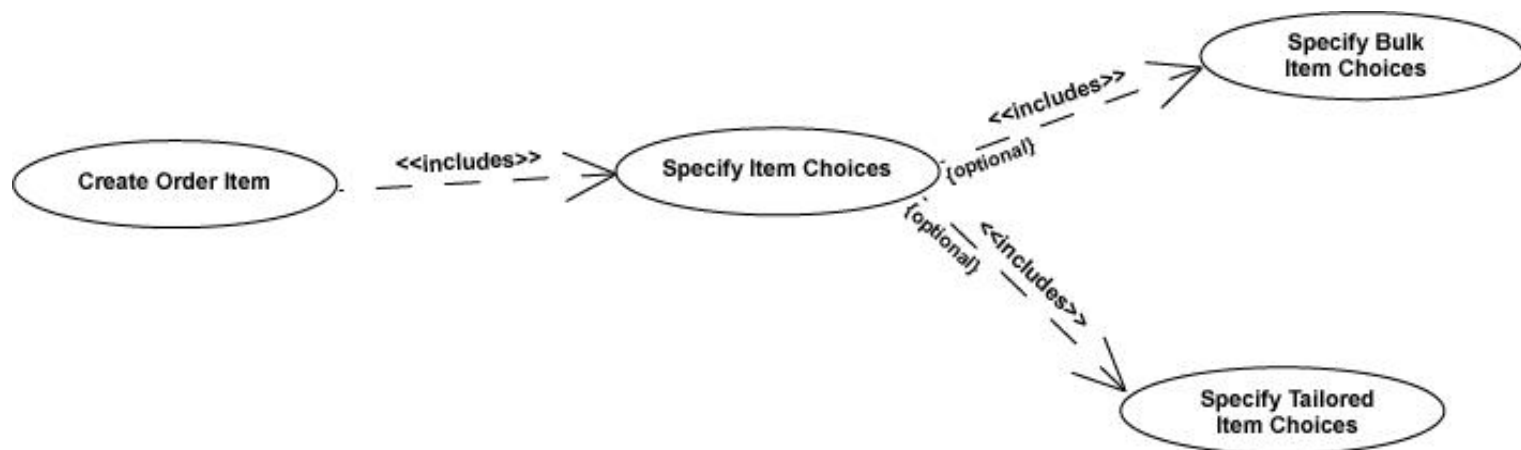
- **Create Order Item**

1. Ask customer for item number
2. Enter *Item Number* into *Add Item* interface
3. Ask customer for quantity of item
4. Enter *Item Quantity* to the interface

- **Specify Bulk Item Choices**

3. Ask customer for amount of item
4. Enter *Item Quantity* and *Units* (one of *Grams, Kilograms*) to the *Add Item* interface

Extends Relationship Alternative



Specify Bulk Item Choices

Scenario

- **Specify Item Choices**

1. Ask customer for quantity of item
2. Enter *Item Quantity* to the interface

- **Specify Bulk Item Choices**

1. Ask customer for amount of item
2. Enter *Item Quantity* and *Units* (one of *Grams, Kilograms*) to the *Add Item* interface

Specify Tailored Item Choices

Scenario

- **Specify Item Choices**

1. Ask customer for quantity of item
2. Enter *Item Quantity* to the interface

- **Specify Tailored Item Choices**

1. Ask customer for quantity of item
2. Enter *Item Quantity* to the *Add Item* interface
3. Ask customer for inseam length in cm.
4. Enter *Inseam Length* (in cm.) to the interface

Extends or Includes?

- Extends makes it easy to specify alternative scenarios that differ by simple substitution or addition of steps
 - But Extends also can lead to ambiguity in Pre- and Post- Conditions
 - *If you use this feature, use it sparingly!*
- Includes typically requires copying and pasting of scenario text
 - But Includes doesn't lead to ambiguity in Pre- and Post- Conditions

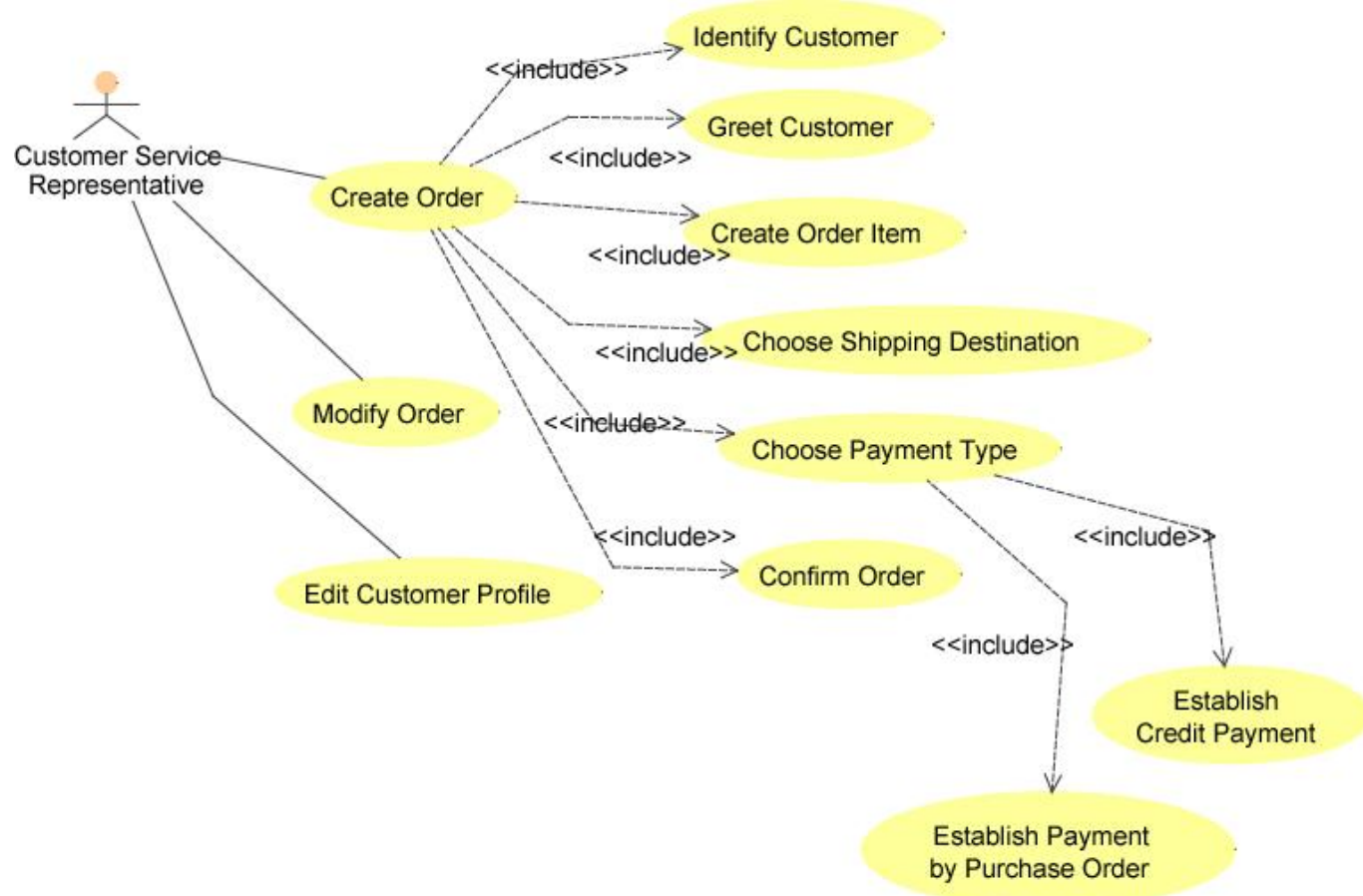
Primary versus Secondary Actors

- Primary Actor
 - The actor who initiates steps in the scenarios.
 - There can be more than one primary actor.
- Secondary Actors
 - The actors who appear within the steps of scenarios.
- This separation makes distinct who is doing the task and who is participating in it.
 - This is where the goal of a use case is critical.
 - The goal reflects the motives of the primary actor.

Example Use Case Model

- Let's take a moment to look at a sketch of a simple example of a use case model based on a generic, catalog-based phone order company

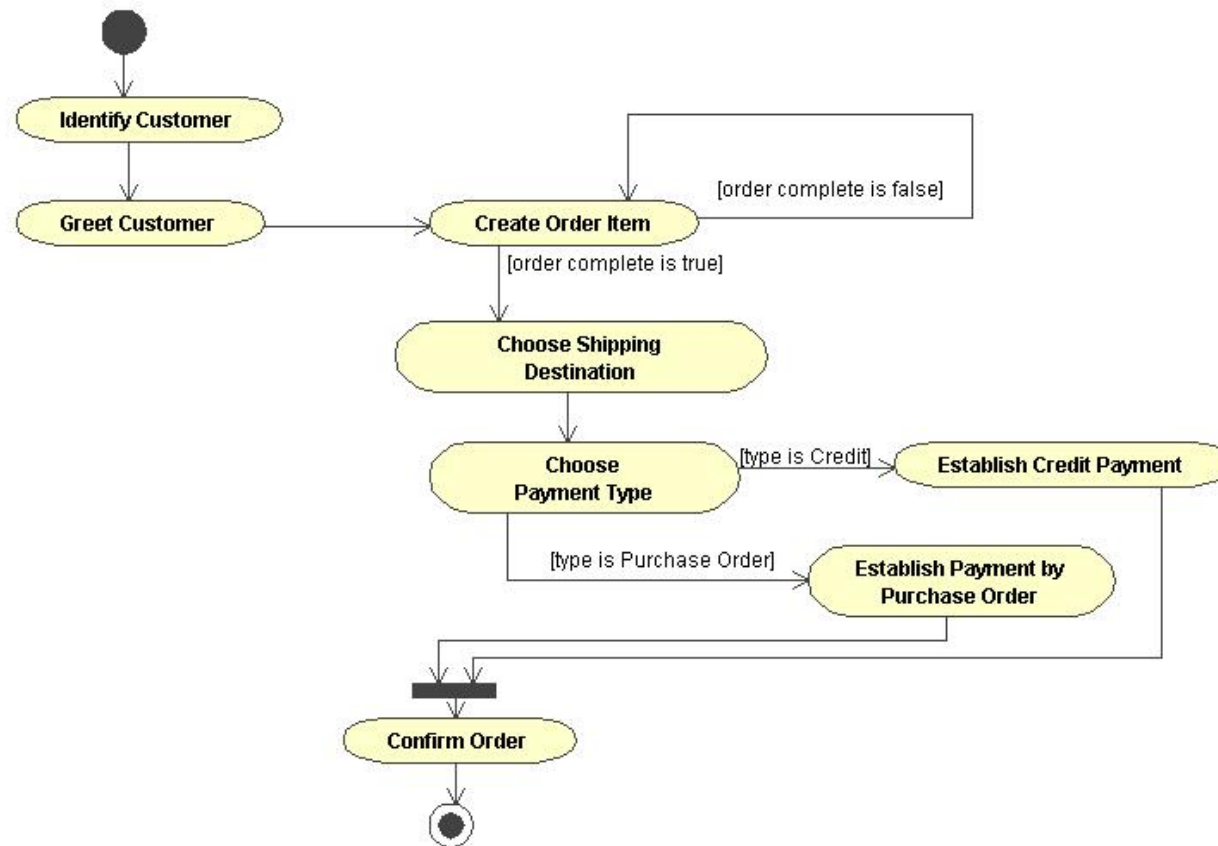
Customer Service Support—Use Case Diagram



Flow Among *Included* Use Cases

- *Create Order* has many included use cases
- Order of the use cases in the diagram can help by implying typical order
- Or an Activity Diagram can explicitly show the flow
 - The next slide illustrates an activity diagram
 - The notebook—*Business Process Modeling*—in this series, explains activity diagrams in more detail

Create Order—Activity Diagram



Use Case Structure

- Flow among use cases
 - Preconditions, Postconditions, Triggers
 - Flow described within Narrative Steps, such as included use cases
- Consistency within a use case
 - Goal, Brief Description, Narrative Steps

How Does this Structure Help Evaluate Use Cases?

- Compare narrative steps and the brief description—do they say the same thing?
- Compare narrative steps to the goal-in-context—do the steps naturally lead to the satisfaction of the goal?
 - Only for success scenarios!
- Compare narrative steps to the preconditions—do the narrative steps follow from the preconditions?
- Compare narrative steps to the postconditions—do the narrative steps naturally lead to establishing the postcondition state?
- For success scenarios only, if the postconditions are true, is the actor's goal satisfied?

How Does this Content Help Compare Use Cases?

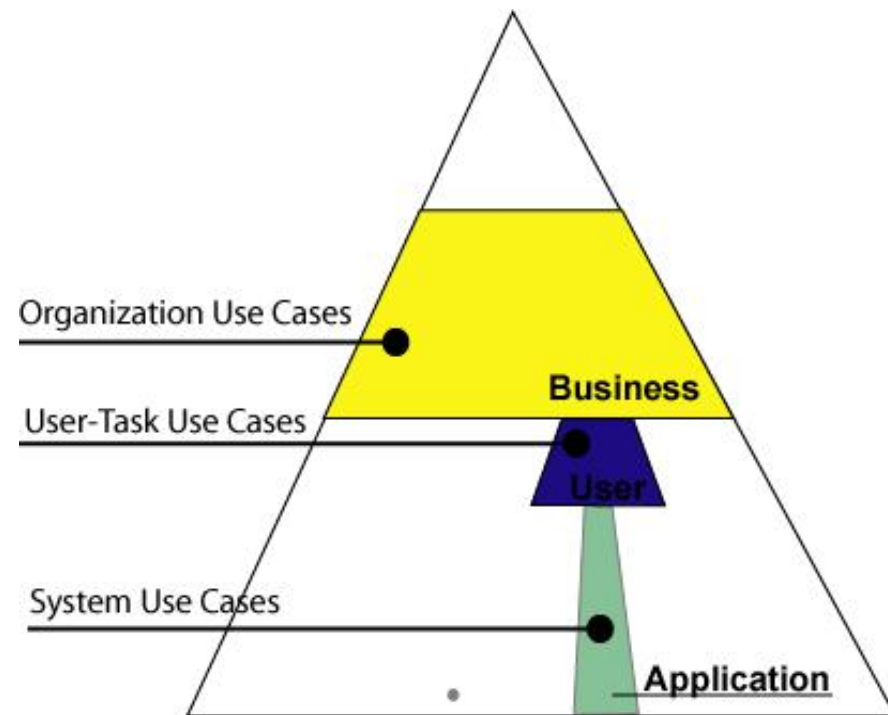
- When one use case leads to another:
 - Do the postconditions of one use case satisfy the preconditions of the next use case?
 - Do the narrative steps of one use case lead to the trigger for the next use case?
 - If not, is there some apparent disconnect?
- Holes and inconsistencies point to errors within a region of scenarios and use cases.

Use Case Levels of Abstraction

- User-Task Use Case
 - A concrete activity performed by one human actor
 - These use cases focus on fulfilling a human goal
- Organization Use Case (or Business Use Case)
 - A use case performed by one organization
 - And fulfilling an organization goal
- System Use Case
 - An activity performed by the SUD
 - Very concrete and usually mechanistic
 - Defines how but doesn't address why

Use Case Levels of Abstraction (cont'd)

- This corresponds to the use case's position in the Task Pyramid.



Use Case Levels of Abstraction—How it Helps

- User-task use cases specify the bulk of requirements
 - These describe what real people are going to do with the SUD
- Organization use cases specify the remainder of the requirements
 - Requirements imposed by business process and not captured by end-user tasks
- Organization use cases also specify the business context

Levels of Abstraction—How it Helps (cont'd)

- Successful approaches to use case authoring use levels of abstraction to guide construction of a use case model.



Identify Your Primary Purpose

- Before you begin any use case work, you must identify your primary purpose:
 - Describing business processes?
 - Exploring a potential product or system?
 - Specifying a product or system?
- Each of these tasks focuses on a different level and scope of use case.
 - This affects where you are in the Task Pyramid.
 - This is covered in detail in the *Pragmatics of Use Case Authoring* notebook.

What You Should Know

- After this notebook, you should know:
 - What a *Use Case Model* is, including *Actors*, *Use Cases*, and *Scenarios*
 - And know some ways to describe a use case model in diagrams or words
 - How the parts of a *Scenario* are used to understand how one *Use Case* relates to the other *Use Cases* in a model