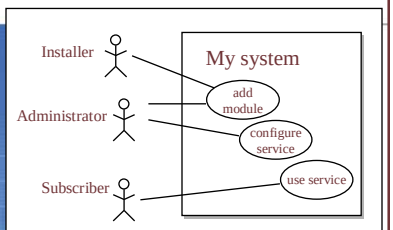
Alcatel·Lucent

# Use Cases Tutorial




Use Case Title: Withdraw cash
Main Scenario:
1. User inserts his/her ATM card
2. System reads and validates the card information
3. User selects transaction and enters transaction details
4. ….

Dennis Mancl

(presented at Trenton Computer Festival, Apr. 24, 2009)

## Requirements

When you develop software, do you have written requirements?

- need to have an agreement between the developers and the clients about what features the software will deliver…
- verbal agreements are subject to misunderstanding

Formats:

- checklist of features
- sample screen shots
- scenarios, stories, use cases

## Use Cases

Scenario-based requirements process:

- Ivar Jacobson, Ericsson, 1980s
- became part of the Rational Unified Process in the late 1990s
- #1 use cases expert: Alistair Cockburn

Focus on functional requirements

- "what" the system will do… not "how"
- describe the interactions between the system and its users
- A use case is "a set of scenarios that describe externally-visible behavior that delivers positive value to one or more actors."





## Use Case Terminology

**Actors:**

- The people or external computer systems that will communicate with your system.

**Goals:**

- The things that the Actors want to achieve.

**Use Case Title:**

- Each Goal turns into a Use Case Title, which is a verb phrase that describes what the actor wants to do.

**Use Case Body:**

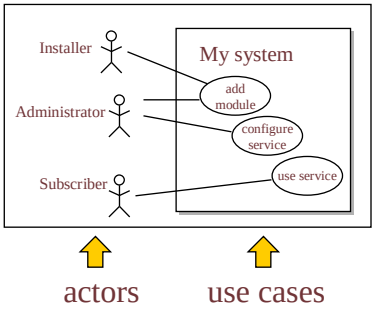
- A bunch of things that describe the "joint requirements" of the system and its actors: main sunny-day scenario, preconditions, postconditions, frequency, performance, business rules, notes.

**Non-use-case requirements:**

- Use Cases are usually only one-third of the volume of requirements. You also need business rules, computational algorithms, constraints, user interface guidelines and prototypes, and non-functional requirements.

## Some basic concepts

A use case diagram




A simple text use case

Use Case Title: Withdraw cash
Main Scenario:

1. User inserts his/her ATM card
2. System reads and validates the card information
3. User selects transaction and enters transaction details
4. System validates transaction details
5. User collects cash and withdraws card
6. System updates the account and resets the system





## Some basic concepts

Systems Requirements Document

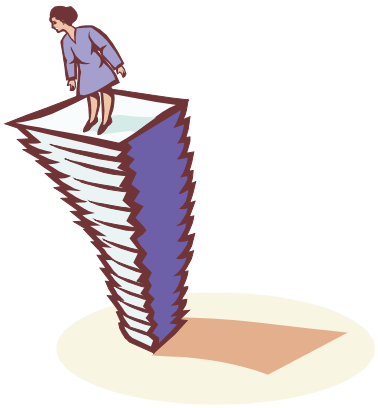
- A bunch of paper documentation that spells out what the system must do
  - Problem: requirements are often too long and tedious, yet still very incomplete
  - The requirements need to explain the behavior that the users really need to use

Traditional

- lots of “shall” statements

New (with use cases)

- short “stories” that outline each user goal



## Important rules in the format of use case documentation

1. For each use case, write a "sunny-day scenario" (a scenario where everything works well).
2. You can add failure branches later:
   - pieces of scenarios that describe how to recover from failing steps
   - pieces of scenarios that describe how to clean up when a scenario can't reach its goal
   - but don't spend a lot of time on failures until you write and review the success scenarios.

   *Use case writing is an iterative process…*





## Notes on use case format

Use cases are "scenario-based requirements"

- Use cases are a way to explain what the system must do, based on a set of small stories
- Each story has a "cast of characters" (the "system" and the "actors")
- Each story starts from a set of initial conditions
  - it finishes with one of the actors "achieving a goal"
  - or it finishes with the actor "giving up" (abandoning the goal because of failures)

Use cases are the easiest way to describe functional requirements

- in a language that makes sense to developers and users

## Important rules in the format of use case documentation

3. Make it clear "who has the ball" at each step of a scenario.
   - Use active voice in each step.
   - The "subject" of each sentence is either the name of an actor or it is "the system".
4. How many use cases is enough? Make sure you cover all of the main goals of each of the actors.
   - Ask if there is an actor who will be upset of they get a system that ***only*** performs the functions in each of the use cases.





## Use case format example

- Goal: Buy a book on the web
- Who wants to do it? A Customer
- What do they need? A web connection and a credit card
- How to they start? Go to http://www.amazon.com
- What is the normal sequence of events?
  1. Customer clicks on search, types in a title
  2. System displays a list of books that match
  3. Customer chooses a book from the list, clicks on "buy this book"
  4. System prompts the user for credit card and delivery details
  5. Customer provides the requested information
  6. System checks that the credit card is valid
  7. System displays the finished order, sends the order number to the Customer

## More notes on use case format

Search for the actors: think about the most important "users" of the system

- customers, administrators, repair people

Goals: Choose goals that the actors really care about

- try for goals that can be achieved by the user and the system working together in "one sitting" (2 to 20 minutes)
- if the goals are too fine-grained, your requirements will be very thick and complicated





## Brainstorm actors and use case titles

System: remote control for a satellite TV system

- Actors:
  - Couch potato
  -
  -
- Goals:
  - Select a program to watch
  - Record a program
  -
  -
  -

## Write a use case

Take one of the goals and write the main scenario

Then think about what can go wrong (failure scenarios)





## Important rules in the *process* of use case documentation

1. Talk to as many people as possible – users, stakeholders (especially the people who will pay the bill), architects, testers, and trainers.
2. Work in four stages:
   - Define the initial actor list and initial list of "goals" (use case titles).
   - Write down sunny-day scenarios for the most important use cases, ***and*** review these scenarios with users/stakeholders.
   - Start filling in the missing parts: missing goals (new actors and use case titles), alternative success scenarios, performance constraints, business rules.
   - Analyze the most important failure cases.

## Notes on use case process

What is a “failure scenario”?

- it is a description of how the system can fail
  - one of the steps of the main scenario fails, the system performs a smooth shutdown
  - one of the steps fails, the system “recovers”
  - failure might be caused by a user (invalid input)
  - failure could be resource exhaustion (ATM runs out of cash, airline website sells the last airline seat)
  - failure could be a real external failure (power outage, network failure, virus)
- Important: don’t overdo the failure scenarios…





## Notes on use case process

What is a “business rule”?

- it is some fact about the system that doesn’t fit into a scenario or a scenario step
  - some fact about the “structure” of the problem domain
    “the system shall have a color display”
  - action triggering and action restricting rules
    “the system will go into power-save mode if there is no activity in 15 minutes”
  - computational algorithms
    “the next operation will be selected using the XYZ algorithm”
- Business rules can be in a separate section of the requirements documentation, with a link to the pertinent use cases

## Important rules in the *process* of use case documentation

3. Within the four stages – try figure out when to leave out some of the details.
   - A requirements document should be "accurate" (it should contain correct information) even if it isn't extremely "precise" (it might not contain lots of detail).
   - It is dangerous to put details into the requirements documentation when the details aren't really required.
   - The scenarios are supposed to describe the ***essential*** behavior.
4. Write down "things that you don't know yet" throughout the use case writing process.
   - One of the most important things about use cases – they help you discover questions that you need to discuss with users and stakeholders.





## Important rules for *reviewing* use cases

1. There are two important kinds of reviewers:
   - users/stakeholders: they are looking for things that might be missing in the requirements
   - technical users of the requirements (architects, designers, coders, and testers): they want the requirements to be complete and unambiguous
   - in a large company: "testers are your friends" – testers provide the most important feedback in use case reviews
2. Pretend to be one of the actors – can you write the first chapter of the user manual from the use cases?

## One format for documenting a use case

Main scenario – extensions – variations

- Each use case has one "main scenario" (sunny-day behavior, describes the events from the beginning state to the "successful completion of the goal")
- Extensions:
  - for each place where a step might fail, define a separate small branch scenario
    - Extension 4a.  Book is out of stock
    - 4a1. System displays "out of stock" indication, asks customer if he/she would like to back-order
    - 4a2. Customer says yes, fills in customer information
    - 4a3. System records order
- A single use case might have six or seven extensions for different kinds of failures





## Failures scenarios and extensions

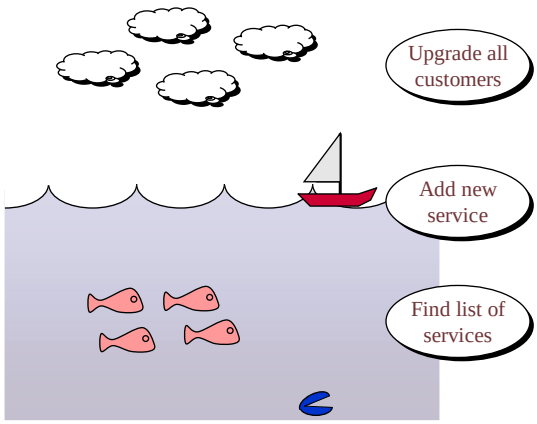
Have you ever used an "unforgiving" system?

- Any time you make a mistake, you have to cancel and go back to the beginning?
- Or you don't receive any notification of input errors, so at some point in the middle of the scenario, the system hangs

How could this situation be avoided?

- Include the most important failure scenarios in the requirements…
- Perform some analysis to find the most common scenarios (and the most common user problems)

## Levels of scope




Summary-level goals
- "cloud-level"
- might be things that operate over hours, days, or weeks

User goals
- "sea-level"
- usually 2 to 20 minutes

Subfunctions
- "under water"
- not a real "user goal": these scenarios describe common suboperations





## Example of use case levels of scope

Simple example: sunny-day scenario for "Order a book from an Internet bookseller"

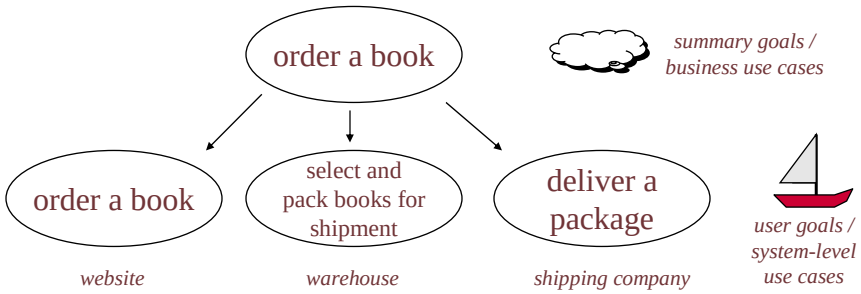
Main scenario

1. Customer navigates to bookseller's website
2. Website displays a list of books to choose from
3. Customer navigates to his/her choice of books, clicks on that book
4. Website displays the details of the selected book
5. Customer clicks on "add to shopping cart" and "check out"
6. Website displays total bill (book price plus tax and shipping cost); prompts for shipping and payment information
7. Customer enters address and credit card information
8. Website commits the purchase – Website displays the "order number" of the purchase to the Customer and sends a confirmation email

## Levels of scope

There is a “cloud-level” use case which starts with the Customer ordering a book on the website and ends with the delivery of the book to the Customer

- Three “system-level” use cases (website, warehouse, shipping company)
- Question: Who are the actors and what are the goals in each “domain”?








## Good books on use cases

*Writing Effective Use Cases* by Alistair Cockburn

*Use Cases: Requirements in Context* by Daryl Kulak and Eamonn Guiney

*Patterns for Effective Use Cases* by Steve Adolph and Paul Bramble

*Use Case Modeling* by Kurt Bittner and and Ian Spence

## Good articles on use cases

Alistair Cockburn's use case pages

- http://alistair.cockburn.us/Structuring+use+cases+with+goals

  Article: "Structuring Use Cases with Goals" – it is a "theory of use cases" which originally appeared in Journal of Object Oriented Programming in two parts (Sept.-Oct. 1997 and Nov.-Dec. 1997).

Alistair Cockburn's use case template (for text-style use cases)

- http://alistair.cockburn.us/Basic+use+case+template

Article by Karl Wiegers:

- http://www.processimpact.com/articles/usecase.html

  Article: "Listening to the Customer's Voice" – describes how to run a use case workshop with real life users

Article by Rebecca Wirfs-Brock:

- http://www.wirfs-brock.com/PDFs/Art_of_Writing_Use_Cases.pdf

  Article: "The Art of Writing Use Cases"





## Wrap-up

Use cases: a technique for writing software requirements

- define actors, write down the main goals of the actors, write scenarios for each use case, think about failures and recovery

Review the use cases

- with customers / stakeholders
- with developers and testers (and listen to their questions)

Use cases are also useful for "business modeling"

- http://www-128.ibm.com/developerworks/rational/library/2784.html