CHAPTER 3 – Use Cases

Introduction
- When, Why, Where, What

Iteratively Developing Use Cases
- Inception
  + Scope Definition + Risk Identification
  + Actors & Use cases + Project Plan
- Elaboration
  + Primary & Secondary Scenarios

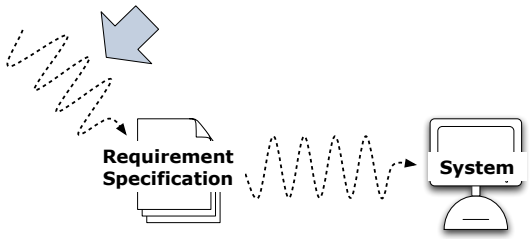
Conclusion
- Use Cases
  + Correctness & Traceability

Literature

Books
- [Ghez02a], [Somm04a], [Pres01a]
  + Chapters on Specification/ (OO)Analysis/ Requirements
- [Schn98a] Applying Use Cases - a Practical Guide, Geri Schneider, Jason, P. Winters, Addison-Wesley, 1998.
  + An easy to read an practical guide on how to iteratively develop a set of use cases and how to exploit it for project planning.
- [Jaco92a] Object-Oriented Software Engineering: A Use-Case Driven Approach, I. Jacobson et. al., Addison-Wesley, 1992.
  + The book that introduced use-cases

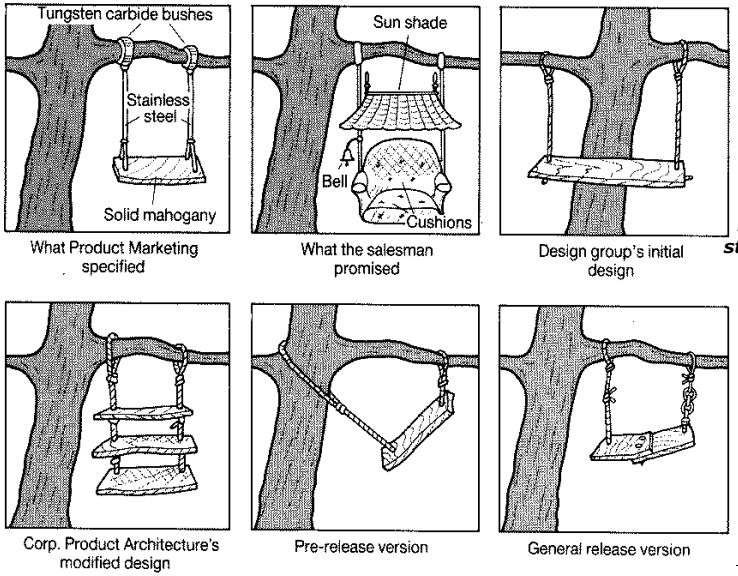
Following article is available on the web
- [Cock97a] "Structuring Use Cases with Goals", Alister Cockburn, Journal of Object-Oriented Programing Sept-Oct & Nov-Dec 1997
  + See http://alistair.cockburn.us/Structuring+use+cases+with+goals
  + Some practical guidelines concerning use-cases, referring to a template to use when writing use-cases
    - http://alistair.cockburn.us/Basic+use+case+template
    - http://www.cs.colorado.edu/~kena/classes/6448/s01/examples/edmonb3.pdf

When Use Cases ?



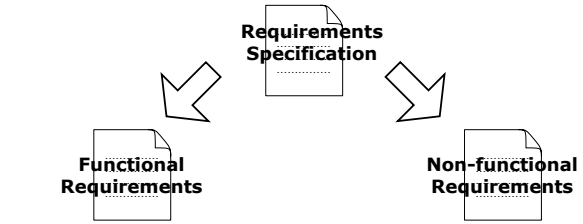
A requirements specification technique must be
- *understandable*: for all parties involved incl. its users
- *precise*: so that parties agree what's inside and outside the system
  Can you write a (regression) test for each requirement?
- *open*: so that developers have enough freedom to pick an optimal solution
  Requirements specify the "what", not the "how ".

Why Use Cases ?



Numerous stakeholders & Limited resources

Where Use Cases ?



- functionality as demanded by the end users

- constraints placed on the global system or the development process.
- quality attributes, such as performance, user-friendliness, maintainability, …

What are Use Cases?

Use Case
- A use case describes outwardly visible requirements of the system
- A use-case is a generic description of an entire transaction executed to achieve a *goal* (= the use case goal) and involving several *actors*.

Actors
- Actors have responsibilities
- To carry out responsibilities, an actor sets goals
- Primary actor (= stakeholder) has unsatisfied goal and needs system assistance
- Secondary actor provides assistance to satisfy the goal

Scenario
- Scenario = an instance of a use-case, showing a typical example of its execution
  - + Use case = Primary "success" scenario and secondary "alternative" scenarios
  - + Scenario shows how objects interact to achieve the use case goal = UML Sequence diagrams & Collaboration Diagrams

Kinds of Use Cases

There is not a "one size fits all": use cases depend on your purpose

Scope
- Brain-storm mode vs. full-fledged detailed specification

Intended Audience
- end-user vs. system development team vs. internal documentation

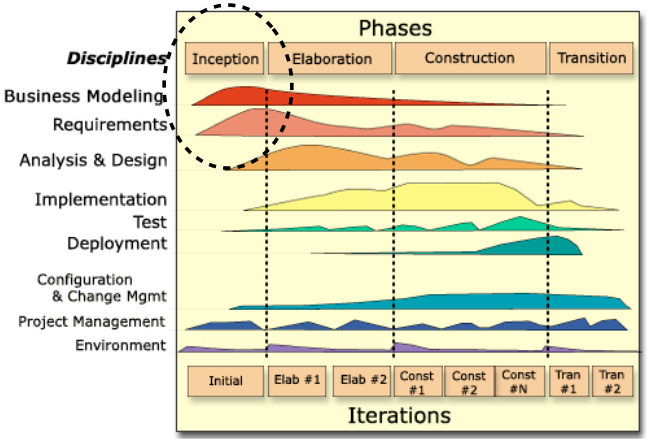
Granularity
- summary vs. detailed; overall system function vs. specific feature
- Brief Use Case — Casual Use Case — Fully Dressed Use case

Black-Box vs. White-Box
- with or without knowledge about (business) processes used to achieve goal

➡ Depending on your purpose some kind of Use Case Template is selected

Unified Process: Inception

Use cases work well when starting an iterative/incremental process !
- e.g. *inception* & elaboration phase in Unified Process



Inception: System Scope

- During inception you must define the system's scope
  + used to decide what lies inside & outside the system

Scope

- should be *short*
  + (1 paragraph for small projects;
    1/2 a page for mid-size projects;
    2-3 pages for large projects)
    ➡ long statements are not convincing
- should be *written* down
  + later reference when prioritizing use cases
- should have end-user *commitment* (*)
  + end-user involved in writing
    ➡ formally approved by a project steering committee

(*) The difference between "involvement" and "commitment" is like an eggs-and-ham breakfast: the chicken was involved; the pig is committed.[Anonymous]

System Scope: Example

(Example from [Schn98a])

- "We are developing order-processing software for a mail-order company called National Widgets, which is a reseller of products purchased from various suppliers.
  + Twice a year the company publishes a catalogue of products, which is mailed to customers and other interested people.
  + Customers purchase products by submitting a list of products with payment to National Widgets. National Widgets fills the order and ships the products to the customer's address.
  + The order-processing software will track the order from the time its is received until the product is shipped.
  + National Widgets will provide quick service. They should be able to ship a customer's order by the fastest, most efficient means possible."

Analyzing the Example

The previous example of a system scope description is

- short (1/2 a page)
  ➡ quick assessment of what's the system supposed to do
- goal-oriented (track orders)
  ➡ open for various solutions
- includes criteria (*quick* service, track *all* of the ordering process, …)
  ➡ will be used to evaluate whether we accomplished the goals
- provides context
  + National Widgets is reseller ⇒ *external* suppliers & shipment
  ➡ the system will not solve everything,
  some problems are out of scope

… and very importantly

- imperfect (*twice* a year? *on-line* catalogue?)
  + may be improved when understanding increases
  + … but goal and main criteria should not change once approved

Inception: Risk Factors

During inception you must identify the project's *risk factors*

- you do not have control over the system's context and it will change
- projects never go according to plan
  ➡ identify potential problems early (… including wild success)

Example

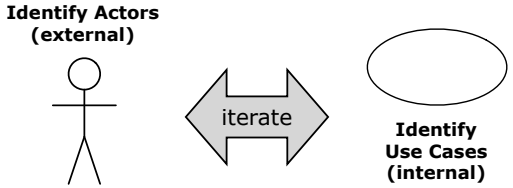
Context	Risk Factors	Impact	Likely ?
Competitors	Time to market (too late/too early)		
Market trends	More internet at home		
Potential disasters	Suppliers don't deliver on time		
	System is down		
Expected users	Too many/few users		
Schedule	Project is delivered too early/too late		
Technology	Dependence on changing technology		
	Inexperienced team		
	Interface with legacy systems		

Inception: System Boundaries

During inception you must specify the system boundaries
- what functionality is *internal* to the system (= use cases)
- what functionality is *external* but necessary for internal functionality (= actors)

⇒ the distinction is often not as clear as you would like it

⇒ iterate: identifying actors + identify use cases



At least one actor must benefit from the use case (i.e. sees the use case value). The corresponding stakeholder will argue to keep the use case in the requirements !

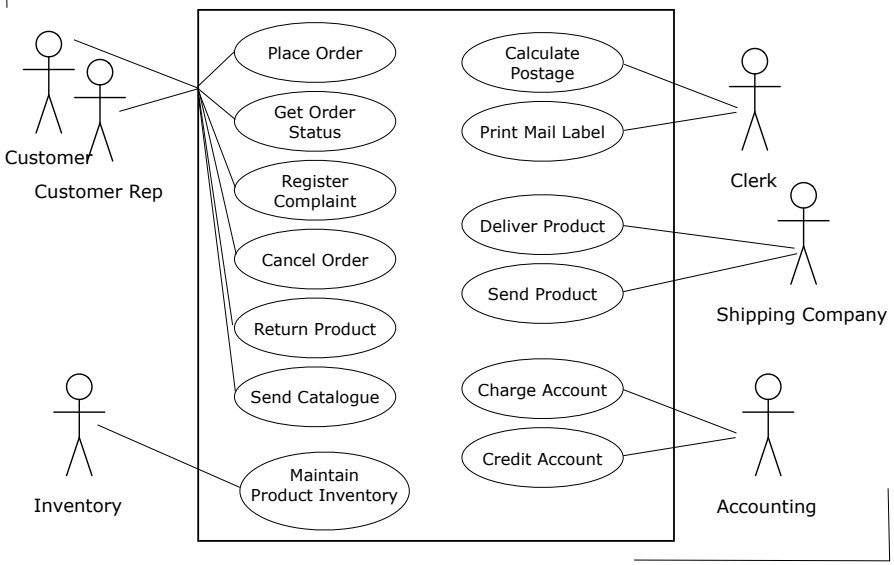
Identifying Actors & Use cases

Following questions may help during the identification process.

Actors
- Who uses the system?
- Who installs the system?
- Who starts up/shuts down the system?
- Who maintains the system?
- What other systems use this system?
- Who provides information to this system ?
- Does anything happen automatically at a preset time ?

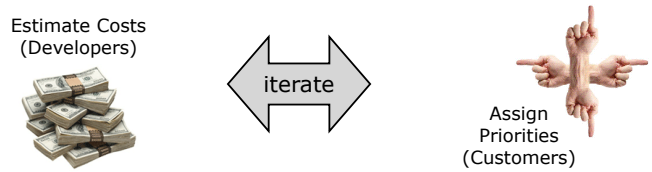
Use Cases
- What functions will the actor want from the system ?
- What actors will create, read, update, or delete information stored inside the system ?
- Does the system need to notify actors about changes in its internal state ?
- Who gets information from this system ?
- Are there any external events the system must know about ?
- What actor informs the system about those events ?

Example: Actors & Use cases



Inception: Project Plan

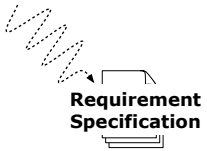
During inception you must specify the project plan
= when to develop which use case
- Includes intermediate milestones
- based on Scope Definition & Risk Factors
- may result in splitting/merging use cases
- negotiate: estimate costs (=developer) + assign priorities (=customer)



Good negotiations obey 2 strict rules
- *Developers estimate cost; customers do not interfere.*
  - ➡ Schedule slips are the responsibility of development team.
- *Customers assign priorities; developers do not interfere.*
  - ➡ Deciding where the money is spent is the customers responsibility.

Terminating the Inception Phase

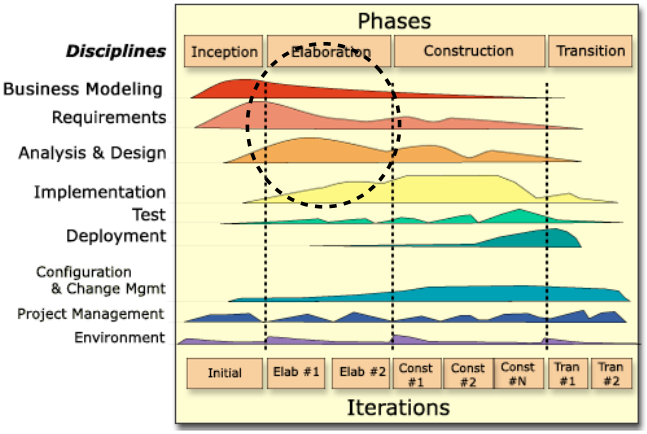
- After inception the requirements specification consists of
  - + Scope definition
    - Short description involving goals, criteria, context
  - + Risk Factors
    - Events that may cause problems during project
  - + Actors
    - Represent the various stakeholders in the project
  - + Use cases
    - Represent transactions; valuable for at least one actor
  - + Project Plan
    - For each use case
      - ➡ Cost estimate (assigned by development team)
      - ➡ Priority (assigned by customers)
    - Time plan including intermediate milestones



Formal approval by project steering committee

Unified Process: Elaboration

Use cases work well when starting an iterative/incremental process !
- e.g. inception & *elaboration* phase in Unified Process



Elaboration: Primary & Secondary Scenarios

- During elaboration you must refine the use cases via scenarios
  - + Scenario is one way to realize the use case
    - ➡ From the actors point of view!
  - + = List of steps to accomplish the use case goal

Primary "success" scenario
- = Happy day scenario
- Scenario assuming everything goes right (i.e., all input is correct, no exceptional conditions, …)

Secondary "alternative" scenarios
- Scenario detailing what happens during special cases (i.e., error conditions, alternate paths, …)

Example: Place Order Scenario (1/2)

USE CASE 5	Place Order
Goal in Context	Customer issues request by phone to National Widgets; expects goods shipped and to be billed.
Scope & Level	Company, Summary
Preconditions	National Widgets has catalogue of goods
Success End Condition	Customer has goods, we have money for the goods.
Failed End Condition	We have not sent the goods, Customer has not spent the money.
Primary Actors	Customer, Customer Rep, Shipping Company
Secondary Actors	Accounting System, Shipping Company
Trigger	Purchase request comes in.

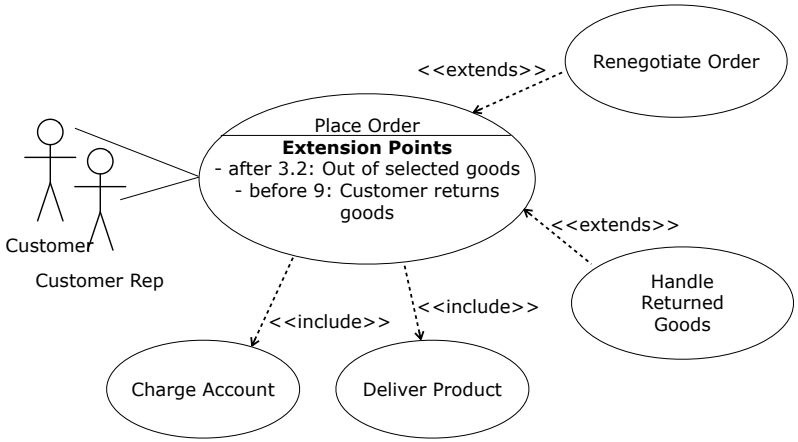
DESCRIPTION	
Step	Action
1.	Customer calls in with a purchase request.
2.	Customer Rep captures customer info.
3.	WHILE Customer wants to order goods.
3.1.	Customer Rep gives Customer info on goods, prices, etc.

Example: Place Order Scenario (2/2)

3.2.	Customer selects good to add to order list.
4.	Customer approves order list.
5.	Customer supplies payment details.
6.	Customer Rep creates order.
7.	Customer Rep requests Accounting System to Charge Account.
8.	Customer Rep requests Shipping Company to Deliver Product.
9.	Customer pays goods.
Branch	*SUBVARIATIONS*
1.	Customer may use: (a) phone in, (b) fax in, (c) use web order form.
4.	Customer may pay via: (a) credit card; (b) cheque; (c) cash.
Branch	*ALTERNATIVE PATHS*
any	Customer may cancel transaction.
Branch	*EXTENSIONS*
After 3.2	Out of selected good: 3.2.a. Renegotiate Order (Use case 44).
Before 9	Customer returns goods: 9a. Handle returned goods (Use case 45).

Place Order Use Case Diagram

Stereotypes <<extends>> and <<include>> to specify use case relationships.
- Beware the direction of the arrows; it specifies change dependencies!



Conclusion

Use cases help you to specify good requirements because it is easier to make them …
- Understandable
  + Actors provide an end users perspective
- Precise
  + Scenarios are sufficiently detailed to test (path coverage)
- Open
  + Actors perspective emphasizes the what (and much less the how)
    ➡ Beware to over specify scenarios; may result in answering "how"

But there is no guarantee, it still requires
- close interaction with various *stakeholders*
  + Project plan negotiation
- iteration to improve earlier misconceptions
- … and lots of hard work

Use Cases & Correctness

Are we building the system right?
- Good use cases will help to validate solution against requirements.
  + Testing
    Writing black box regression tests via use cases should be easy.
  + Design by Contract
    Pre- and postconditions form initial basis for contracts.
- … however, step to system design (architecture) + detailed design (objects) is hard.
  + Use cases tend to result in hard to maintain systems

Are we building the right system?
- Good use cases help to verify the requirements against users needs.
  + Understandable & precise
    … use cases will of course help to verify requirements
- … however
  + Identifying actors and use cases may omit requirements
    ➡ Completeness is not guaranteed !
  + Focus on scenarios (= control flow) restricts evolving requirements
    ➡ Requirements should specify "what" not "how"

Use Cases & Traceability

Requirements ⇔ System

- Via proper naming conventions
  + … including names of regression tests

Requirements ⇔ Project Plan

- Use cases form good milestones
- Estimating development effort for use cases is feasible
  - ➡ Balancing Numerous stakeholders against Limited resources

Use cases form a good base for negotiating the project plan.

Summary(i)

You should know the answers to these questions

- Why should the requirements specification be understandable, precise and open ?
- What's the relationship between a use case and a scenario ?
- Can you give 3 criteria to evaluate a system scope description ? Why do you select these 3 ?
- Why should there be at least one actor who benefits from a use case ?
- Can you supply 3 questions that may help you identifying actors? And use cases?
- Which two basic rules apply to the project plan negotiation ? Why ?
- Which sections should be included into the requirements specification at the end of the inception phase ?
- What's the difference between a primary scenario and a secondary scenario ?
- What's the direction of the <<extends>> and <<includes>> dependencies ?

You should be able to complete the following tasks

- Write a requirements specification (incl. scope definition, actors & use cases and project plan) for your year project.

Summary(ii)

Can you answer the following questions?

- Can you explain the difference between the two definitions for use cases ?
- Why do use cases fit well in an iterative/incremental development process ?
- Why do we distinguish between primary and secondary scenarios ?
- Assume that you work for a company that does not want to apply use cases in their requirements specification. Which principles would you still apply regardless of the use case notation ?
- What would you think would be the main advantages and disadvantages of use cases ?
- How would you combine use-cases to calculate the risky path in a project plan ?
- Do use-cases work well with agile methods ? Explain why or why not.