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**

- [Ghez02a], [Somm04a], [Pres01a]  
  ⇒ Chapters on Specification/ (OO)Analysis/ Requirements

  ⇒ 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.

  ⇒ The book that introduced use-cases

Following article is available on the web

  See [http://members.aol.com/acockburn/papers/usecases.htm](http://members.aol.com/acockburn/papers/usecases.htm)  
  ⇒ Some practical guidelines concerning use-cases, referring to a template to use when writing use-cases ([http://members.aol.com/acockburn/papers/uctempla.htm](http://members.aol.com/acockburn/papers/uctempla.htm))
**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

Tungsten carbide bushes
Stainless steel
Solid mahogany

What Product Marketing specified

Sun shade
Bell
Cushions

What the salesman promised

Design group’s initial design

Corp. Product Architecture’s modified design

Pre-release version

General release version

Universiteit Antwerpen
Where Use Cases?

- **Functional Requirements**
  - functionality as demanded by the end users

- **Non-functional Requirements**
  - 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 + 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

**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**

Use cases work well when starting an iterative/incremental process.

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Inception</th>
<th>Elaboration</th>
<th>Construction</th>
<th>Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Implementation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test</td>
<td>Iter. #1</td>
<td>Iter. #2</td>
<td>...</td>
<td>Iter. #n-1</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
**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
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.”
Analysing 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
**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
  $\Rightarrow$ identify potential problems early (... including wild success)

**Sample**

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

<table>
<thead>
<tr>
<th>Actors</th>
<th>Use Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Who uses the system?</td>
<td>What functions will the actor want from the system?</td>
</tr>
<tr>
<td>Who installs the system?</td>
<td>What actors will create, read, update, or delete information stored inside the system?</td>
</tr>
<tr>
<td>Who starts up/shuts down the system?</td>
<td></td>
</tr>
<tr>
<td>Who maintains the system?</td>
<td>Does the system need to notify actors about changes in its internal state?</td>
</tr>
<tr>
<td>What other systems use this system?</td>
<td></td>
</tr>
<tr>
<td>Who gets information from this system?</td>
<td>Are there any external events the system must know about? What actor informs the system about those events?</td>
</tr>
<tr>
<td>Who provides information to this system?</td>
<td></td>
</tr>
<tr>
<td>Does anything happen automatically at a preset time?</td>
<td></td>
</tr>
</tbody>
</table>
**Example: Actors & Use cases**

- **Customer Rep**
  - Place Order
  - Get Order Status
  - Register Complaint
  - Calculate Postage
  - Print Mail Label
  - Send Product

- **Customer**
  - Place Order
  - Get Order Status
  - Return Product
  - Send Catalogue
  - Maintain Product Inventory
  - Charge Account
  - Credit Account

- **Clerk**
  - Deliver Product
  - Charge Account

- **Shipping Company**
  - Send Product

- **Inventory**
  - Maintain Product Inventory

- **Accounting**
  - Charge Account
  - Credit Account
**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
Primary & Secondary Scenarios

During elaboration you must refine the use cases via scenarios
- Scenario is one way to realise 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, ...)

Scenarios

A scenario consists of

Preconditions/Postconditions

- Indicate the state of the system before starting /after completing the use case

Flow of Events

- Declarative statements listing the steps of the use case
- May include branches (if) & repetition (loop)

Alternative Paths

- Alternatives for the flow of events
- Typically for events that can happen all the time (cancel, help, ...)

Extensions

- Describe secondary scenarios
- Primary scenario lists (a) the step, (b) the condition (triggers alternative) and (c) a reference to the secondary scenario

*** Select a use case template which makes it easy to supply all above information. Choose template and writing style depending on target reader. 

see for example http://members.aol.com/acockburn/papers/uctempla.htm
## Place Order Scenario

**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

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Customer calls in with a purchase request.</td>
</tr>
<tr>
<td>2</td>
<td>Customer Rep captures customer info.</td>
</tr>
<tr>
<td>3</td>
<td>WHILE Customer wants to order goods.</td>
</tr>
<tr>
<td>3.1</td>
<td>Customer Rep gives Customer info on goods, prices, etc.</td>
</tr>
<tr>
<td>Step</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>3.2</td>
<td>Customer selects good to add to order list.</td>
</tr>
<tr>
<td>4</td>
<td>Customer approves order list.</td>
</tr>
<tr>
<td>5</td>
<td>Customer supplies payment details.</td>
</tr>
<tr>
<td>6</td>
<td>Customer Rep creates order.</td>
</tr>
<tr>
<td>7</td>
<td>Customer Rep requests Accounting System to Charge Account.</td>
</tr>
<tr>
<td>8</td>
<td>Customer Rep requests Shipping Company to Deliver Product.</td>
</tr>
<tr>
<td>9</td>
<td>Customer pays goods.</td>
</tr>
</tbody>
</table>

**SUBVARIATIONS**

<table>
<thead>
<tr>
<th>Branch</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Customer may use: (a) phone in, (b) fax in, (c) use web order form.</td>
</tr>
<tr>
<td>4</td>
<td>Customer may pay via: (a) credit card; (b) cheque; (c) cash.</td>
</tr>
</tbody>
</table>

**ALTERNATIVE PATHS**

<table>
<thead>
<tr>
<th>Branch</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>any</td>
<td>Customer may cancel transaction.</td>
</tr>
</tbody>
</table>

**EXTENSIONS**

<table>
<thead>
<tr>
<th>Branch</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>After 3.2</td>
<td>Out of selected good: 3.2.a. Renegotiate Order (Use case 44).</td>
</tr>
<tr>
<td>Before 9</td>
<td>Customer returns goods: 9a. Handle returned goods (Use case 45).</td>
</tr>
</tbody>
</table>
**Place Order Use Case Diagram**

Stereotypes `<<extends>>` and `<<include>>` to specify use case relationships.

⇒ Beware the direction of the arrows; it specifies change dependencies!

**Customer Rep**

**Customer**

**Place Order**

**Extension Points**
- after 3.2: Out of selected goods
- before 9: Customer returns goods

**Renegotiate Order**

**Handle Returned Goods**

**Charge Account**

**Deliver Product**
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)
  \[\Rightarrow\] Beware to overspecify scenarios; may result in answering “how”

But there is no guarantee, it still requires
- close interaction with various stakeholders
  \[\Rightarrow\] 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
  - Focus on scenarios (= control flow) restricts evolving requirements
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

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?