Use Case Modeling
The required techniques of effective reasoning are pretty formal, but as long as programming is done by people that don't master them, the software crisis will remain with us and will be considered an incurable disease. And you know what incurable diseases do: they invite the quacks and charlatans in, who in this case take the form of Software Engineering gurus.

- E. Dijkstra
Program testing can be used to show the presence of bugs, but never to show their absence! - E. Dijkstra
Use Case

• A collection of possible interactions between the system under discussion and its external actors related to a particular goal.
• A story of using the system to meet a goal.
• What the system will do at a high level with a user focus.
• Describes what your system does to accomplish a particular customer goal.
Use Case

Use case includes:

- Clear value
- Starting and stopping point
- External initiator

Use case:

- Focus on WHAT not HOW
Use Case

Actor

- Something with behavior
- Often people/users
- System is an actor

Scenario

- A specific sequence of actions and interactions between actors and the system
- One possible path through the use case
- May describe successful achievement of goal or may describe a failure
Use Case

• Primarily captures functional requirements
• Does not capture all requirements
• Is not inherently object-oriented
• Helps determine scope of project and give application structure
• Keys: *simple* and *usable*
Use Case - Formatting

Text documents – not diagrams

3 formats:
  – Brief
  – Casual
  – Fully dressed
Use Case - Formatting

Fully dressed format:
- Name – name in terms of a user goal
- Primary actor
- Stakeholders and interests
- Preconditions
- Success guarantees (postconditions)
- Main success scenario (main path or “happy path”)
- Alternate paths (extensions)
- Special requirements
Use Case - Formatting

Three kinds of steps in a scenario:

– An interaction between actors (typically between primary actor and system)
– A validation
– A state change by system
Use Case - Guidelines

1. Write in an essential UI-free style
2. Write terse use cases
3. Write black-box use cases (specify WHAT not HOW)
4. Take an actor and actor-goal perspective (focus on user goals)
5. How to find use cases:
   • Choose system boundary
   • Identify primary actor(s)
   • Identify goals for each actor
   • Define use cases to satisfy user goals (name them using goal)
   • One use case per goal with exception of CRUD (create, retrieve, update, delete), e.g. Manage <X>
Use Case - Guidelines

6. Tests to find useful use cases:
   • Boss test
   • EBP (elementary business process) test
   • Size test
Fig. 6.3

NextGen POS

Process Sale

Handle Returns

Cash In

Analyze Activity

Manage Security

Manage Users

...
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.

Fig. 6.4

Cashier

NextGen

Process Sale

Payment Authorization Service

primary actors on the left

supporting actors on the right
**Fig. 6.1**

Sample UP Artifact Relationships

**Domain Model**

<table>
<thead>
<tr>
<th>Sale</th>
<th>Sales</th>
<th>LineItem</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>Linetem</td>
<td>quantity</td>
</tr>
</tbody>
</table>

**Operation**: `enterItem(…)`

**Post-conditions**: `- . . .`

**Operation Contracts**

- **Sale date**
- **Sales LineItem quantity**

**Business Modeling**

**Use-Case Model**

Process Sale
1. Customer arrives ...
2. Cashier makes new sale.
3. ...

**Use-Case Diagram**

**Use Case Text**

System events

Operation: `enterItem(…)`

Post-conditions: `- . . .`

**Operation Contracts**

- `System` requires `make NewSale()`
- `enterItem (id, quantity)`

**Design Model**

- `Register`
- `ProductCatalog`
- `Sale`

- `spec = getProductSpec( itemID )`
- `addLineItem( spec, quantity )`

**Sample UP Artifact Relationships**

- **System Sequence Diagrams**
- **Vision**
- **Glossary**
- **Supplementary Specification**

**Requirements**

- `use case names`
- `terms, attributes, validation`
- `scope, goals, actors, features`

**Use-Case Text**

- `: System`
- `make NewSale()`
- `enterItem (id, quantity)`

**System Sequence Diagrams**

- `Cashier` events
- `Customer` arrivals
- `Cashier` makes new sale
- `3. ...`

**Supplementary Specification**

- non-functional reqs, quality attributes