
# Use Cases from Requirements to Test

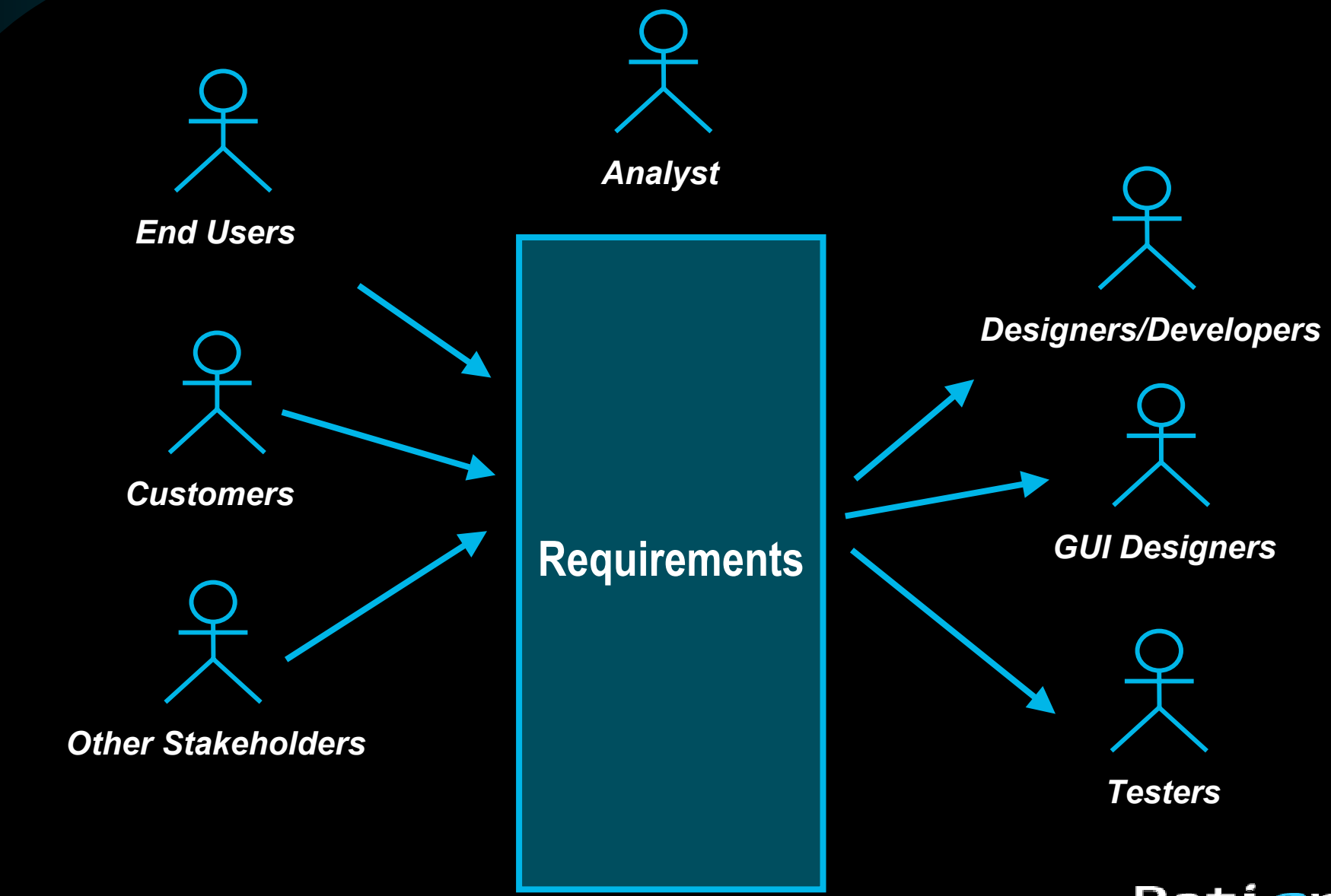
Ivar Jacobson
Vice President Process Strategy
Rational Software Corporation
*E-mail: ivar@rational.com*
*URL: www.rational.com*

# Use Cases are part of a LifeCycle Process

The Rational Unified Process

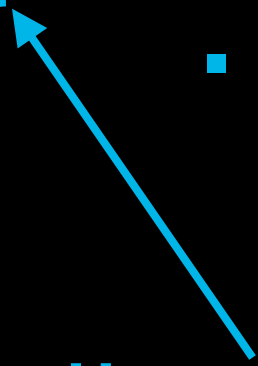
- It is the UML process
- It is component-based
- It is
  - **Use-case driven**
  - Architecture-centric
  - Iterative and incremental
  - Configurable

# Stakeholders of Requirements



# There Are Lots of Types of Requirements

- **FURPS**
  - Functionality
  - Usability
  - Reliability
  - Performance
  - Supportability
- **Design Constraints**
  - Operating systems
  - Environments
  - Compatibility
  - Application standards
- **Compliance with Legal and Regulatory requirements**
  - Federal Communication Commission
  - Food and Drug Administration
  - Department of Defense

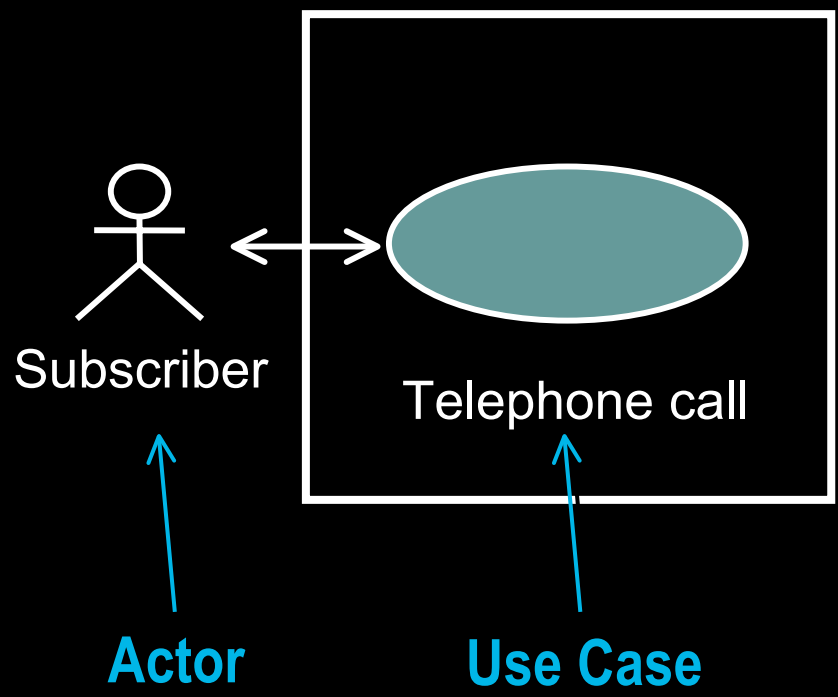


**Use cases address these requirements!**

# Use Cases Specify Requirements

**A use case is a sequence of actions a system performs that yields an observable result of value to a particular actor**

- Use cases reside inside the system
- A use case describes the actions the system takes to deliver to the actor
- Taken together, all use cases constitute all ways of using the system

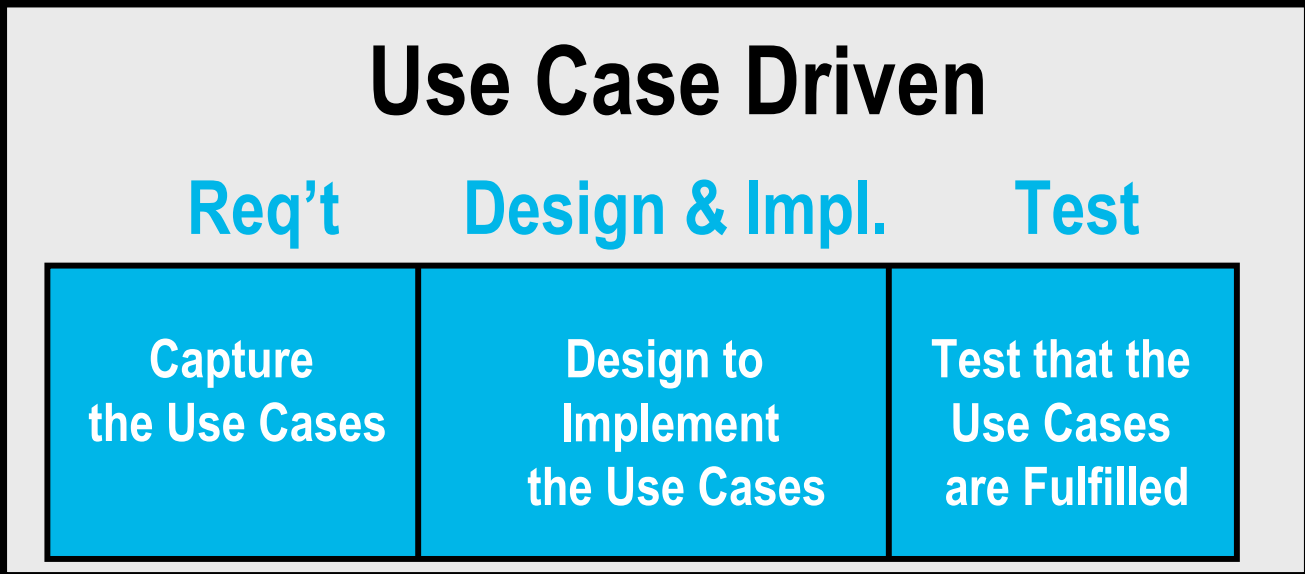


# Use Case Driven

Any product development should follow three steps:

- Capture the users' needs
- Design to fit those needs
- Test that the needs are fulfilled

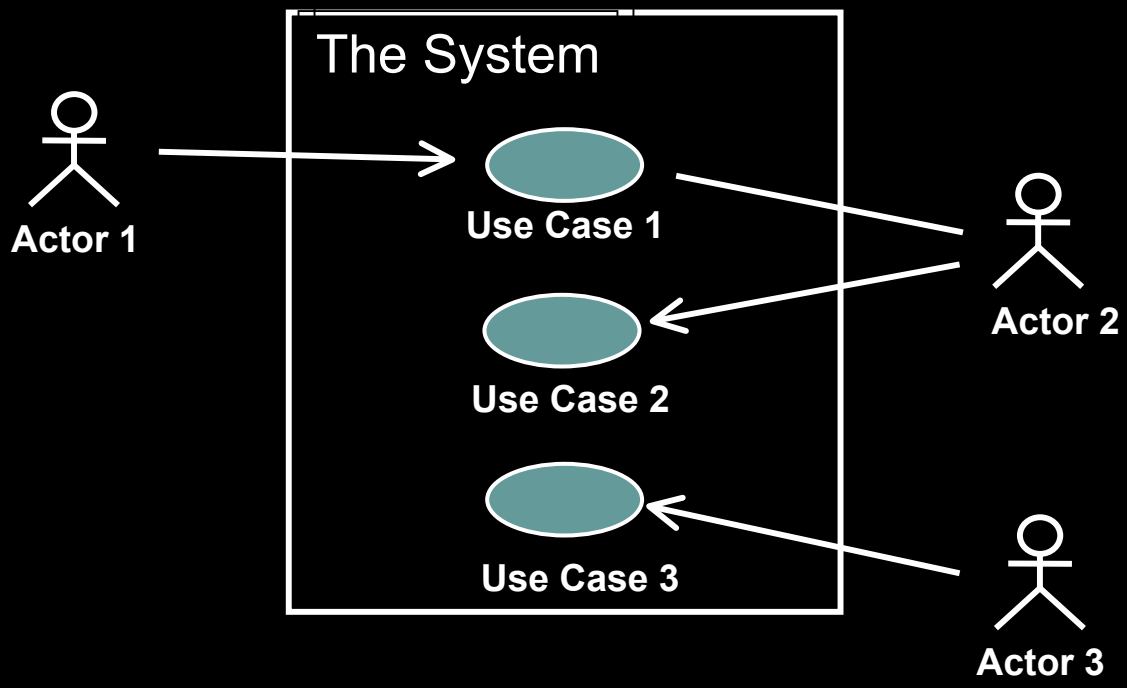
} Users' Needs are Use Cases !



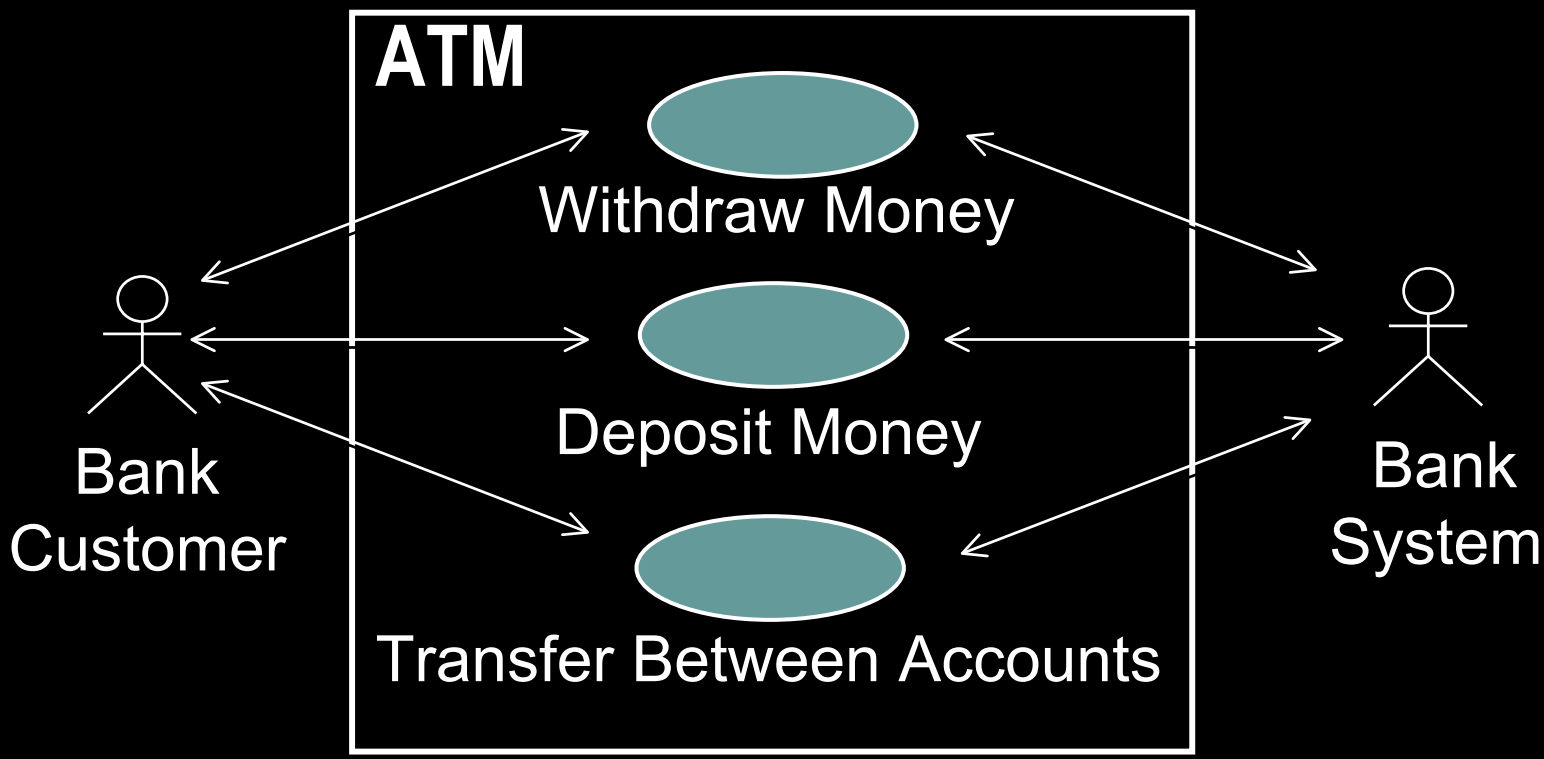
# Use Case Driven

| Req't | Design & Impl. | Test |
| --- | --- | --- |
| Capture the Use Cases | Design to Implement the Use Cases | Test that the Use Cases are Fulfilled |

# A Use-Case Model Contains Diagrams

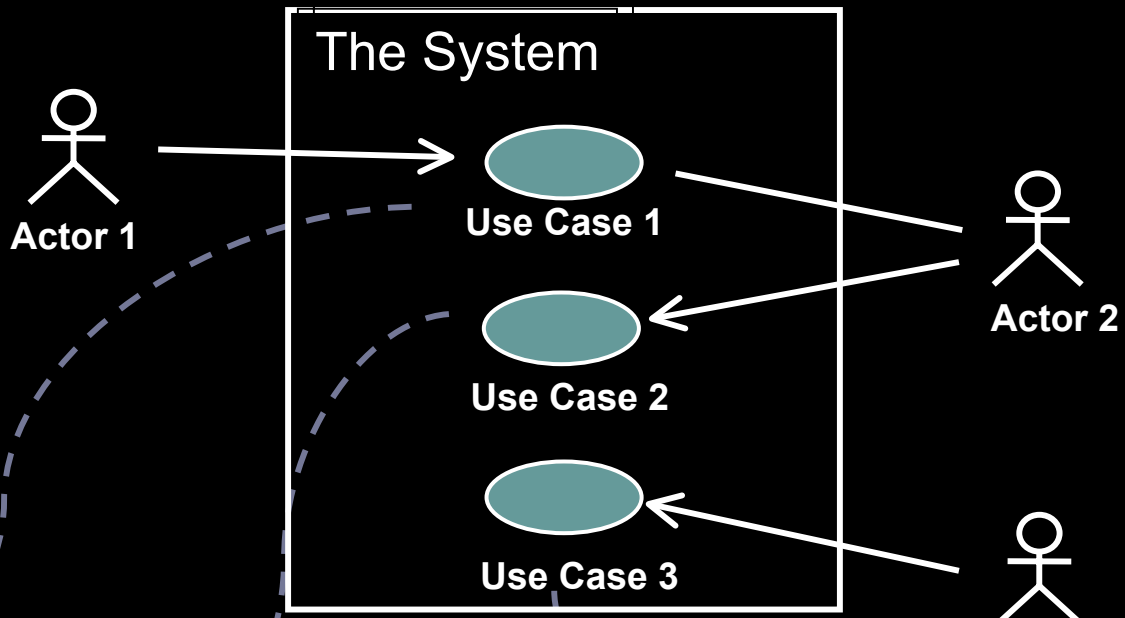


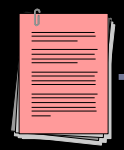
# An Example: The Automatic Teller Machine



# A Use-Case Model Also Contains Text

**Use-Case-Model Survey**
- survey description
- list of all actors
- list of all use cases



**Use Case 1 Report**
- brief description
- flow of events

**Use Case 2 Report**
- brief description
- flow of events

**Use Case 3 Report**
- brief description
- flow of events

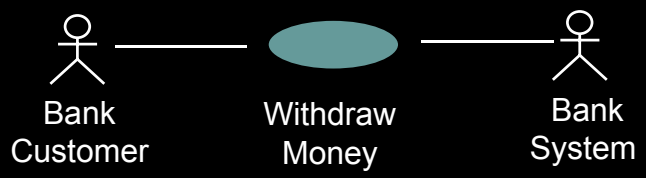
# Use-Case Report Template

**Use-Case Name**

1. Brief Description
2. Flows of Events
   - Basic Flow of Events
     - Step 1
     - Step 2
     - Step …
   - Alternative Flows of Events
     - Alt Flow 1
     - Alt Flow 2
     - Alt Flow 3
     - Alt Flow …
4. Special Requirements
5. Pre-Conditions
6. Post-Conditions

- One Basic Flow
  - Happy-Day Scenario
- Many Alternative Flows
  - Regular variants
  - Odd cases
  - Exceptional (error) flows

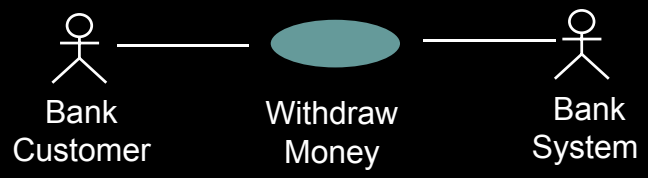
# Use Case Report: Withdraw Money – Basic Path

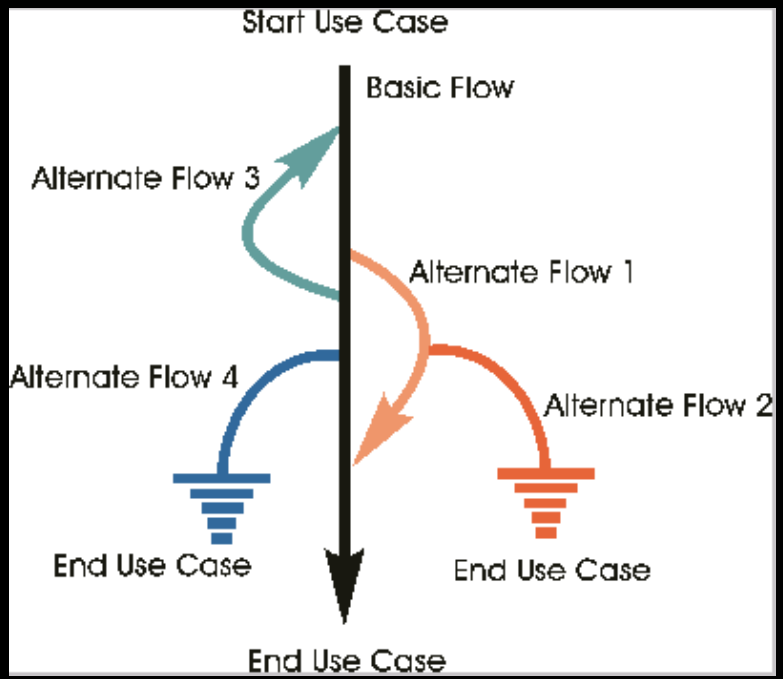



## Withdraw Money

- When a customer inserts a card, the machine reads the code from the card.
- When the PIN code is received, the machine asks for which transaction the customer wishes to perform.
- When the customer selects cash withdrawal, the machine asks for the amount. Only multiples of $20 are allowed (may change).
- When …

# Use Case Report: Withdraw Money – Alternative Paths




## Withdraw Money

- When a customer inserts a card, the machine reads the code from the card and checks if it is an acceptable card. If the card is not acceptable, the machine eats the card. Otherwise the machine asks for the PIN-code.
- When the PIN code is received, the machine checks whether the PIN code is correct for the specific card. If the PIN code is invalid, the machine eats the card. Otherwise the machine asks for which transaction the customer wishes to perform.
- When the customer selects cash withdrawal, the machine asks for the amount. Only multiples of $20 are allowed (may change).
- When …

# Use Case Modeling

- Done for, and together with, the users, customers and designers of the system

- Leads to a complete “what” model, an outside view, of the system
- The Use Case Model allows us to validate the system already in the analysis work.

# A Use Case Model vs a Traditional Functional Spec.?

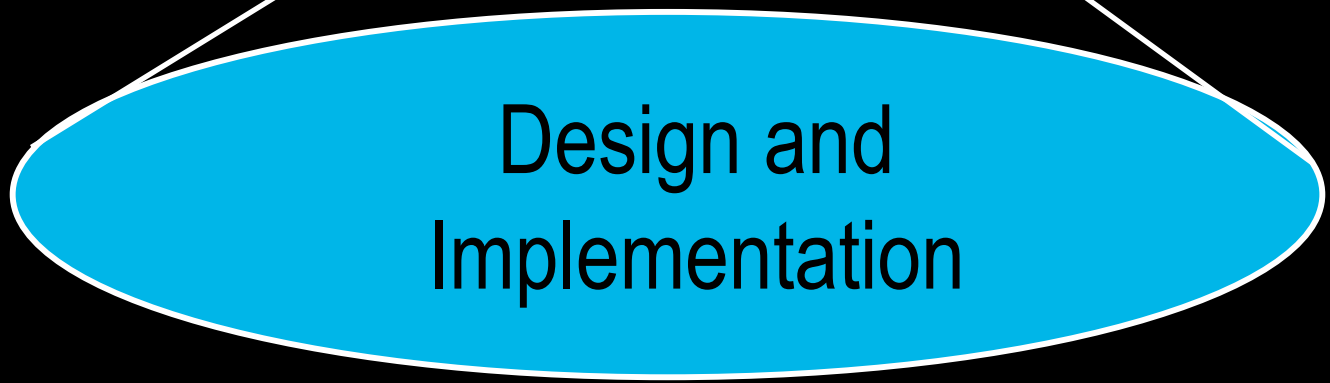
- A functional specification attempts to reply to the question:
  *"What is the system supposed to do?"*
- A use case strategy forces us to add three words to the end of that question:
  *"… for each user?"*

# Benefits of Use Cases in Requirements

- Easy to understand – they are a means of communication
- Focus on observable result of value to an actor
- Facilitate agreement with the customer on system requirements

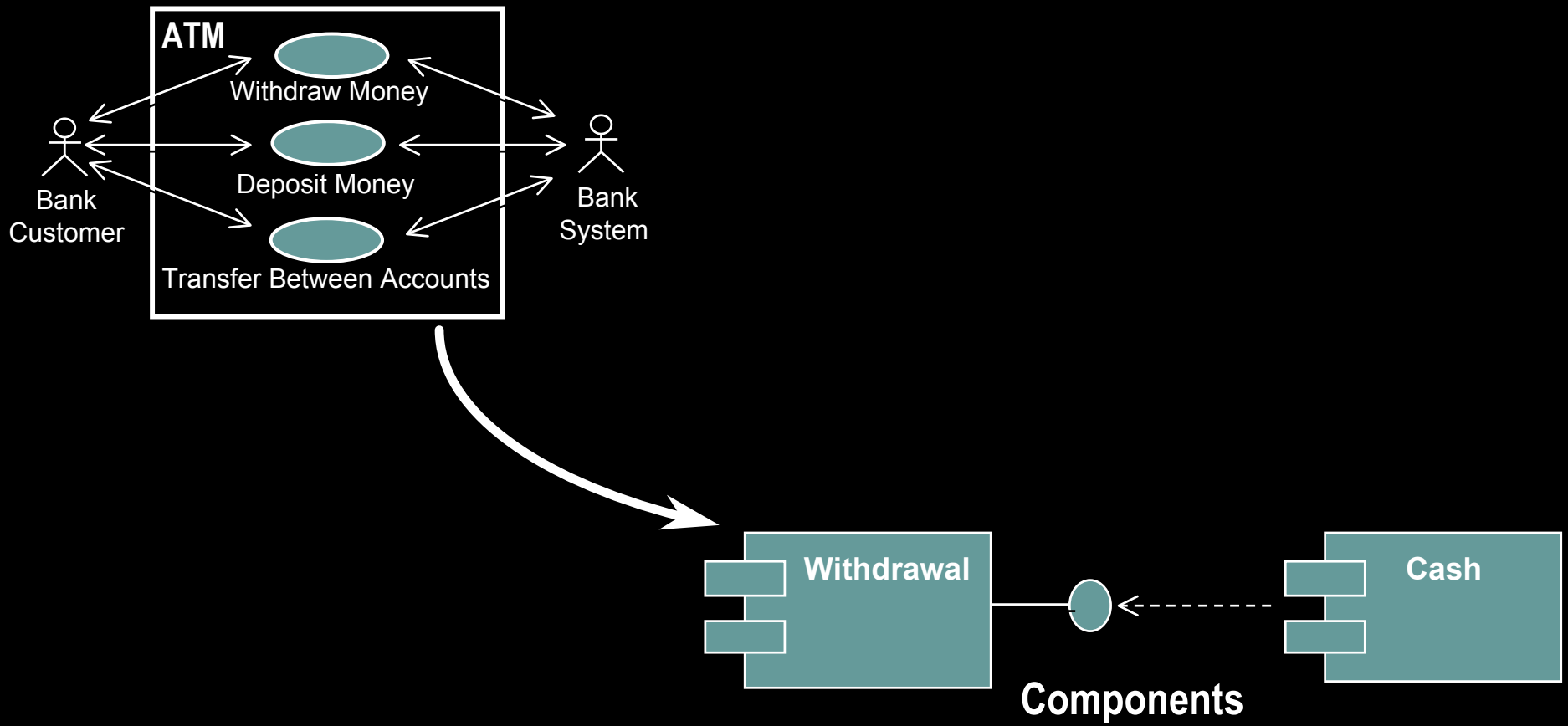
# Use Case Driven

| Req't | Design & Impl. | Test |
| --- | --- | --- |
| Capture the Use Cases | **Design to Implement the Use Cases** | Test that the Use Cases are Fulfilled |



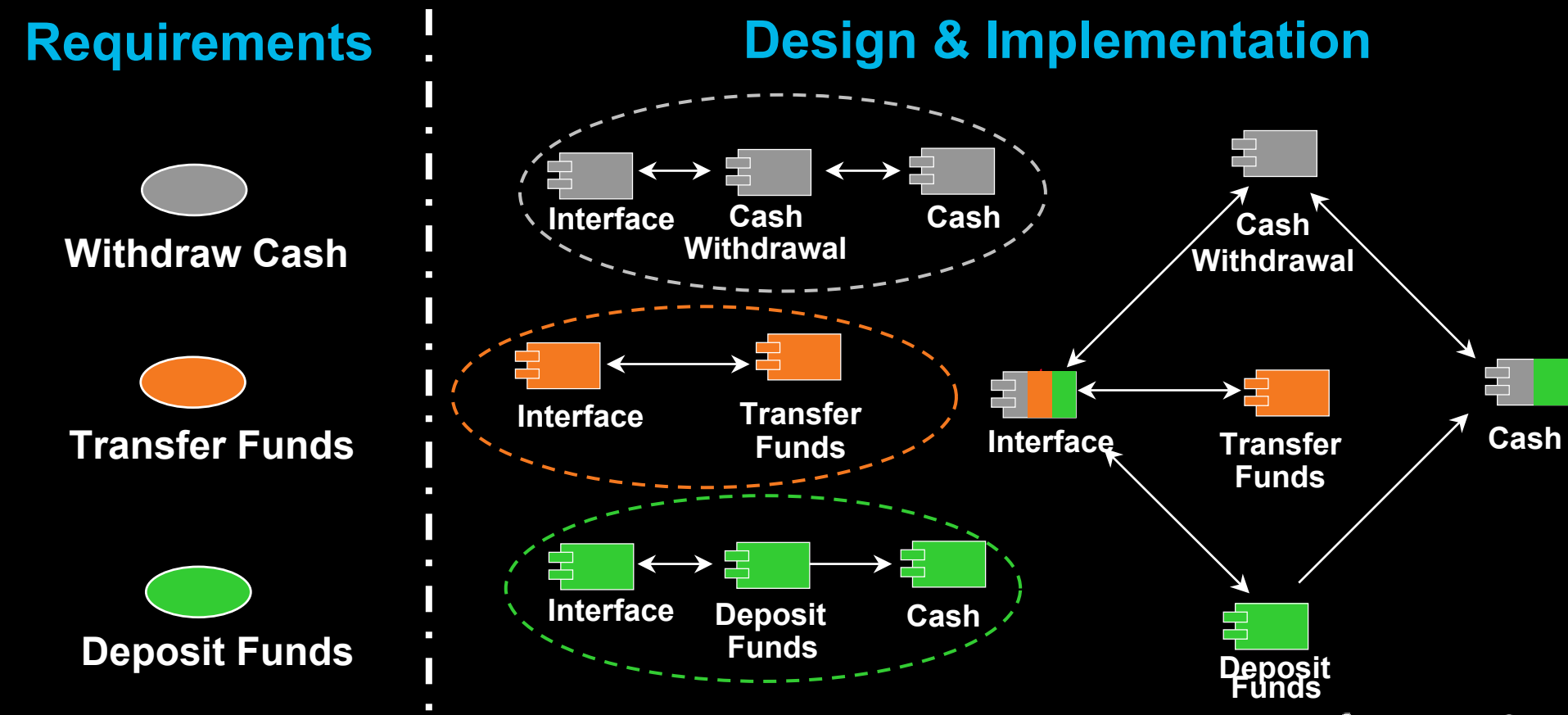
# What is Use Case Driven Design?

- Use cases are eventually realized as components
- Components of the implementation



# Use Cases to Design & Implementation

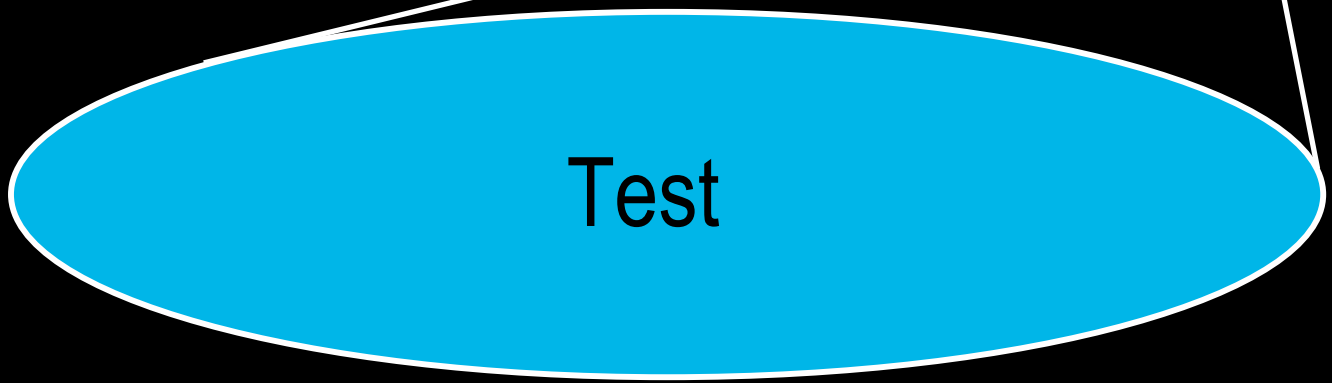
- Each use case is realized by a collaboration - a set of classes
- A class plays different roles in different use case realizations
- The total responsibility of a class is the integration of these roles



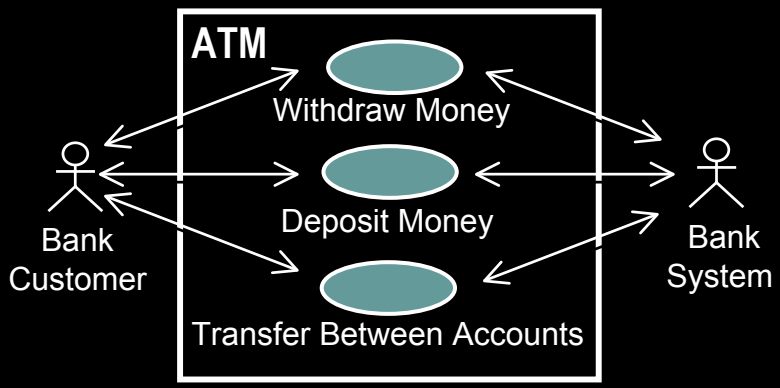

Rational
the software development company

# Use Case Driven

| Req't | Design & Impl. | Test |
| --- | --- | --- |
| Capture the Use Cases | Design to Implement the Use Cases | Test that the Use Cases are Fulfilled |



# What is Use Case Driven Test?



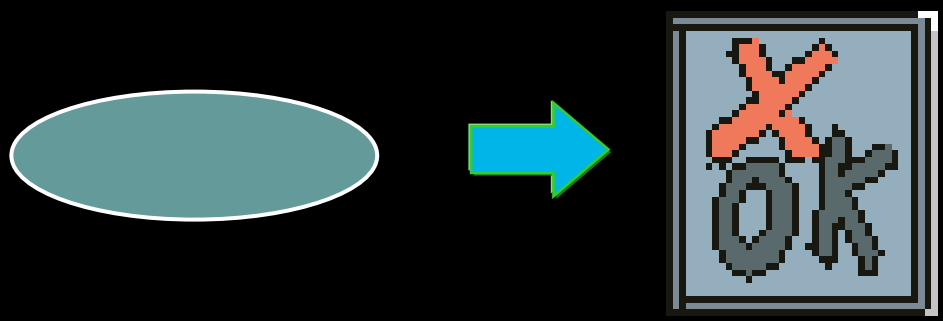

**Test Cases**

- Cash Withdrawal of a pre-set amount
- Cash Withdrawal of custom amount
- Etc. Many Test Cases for every Use Case

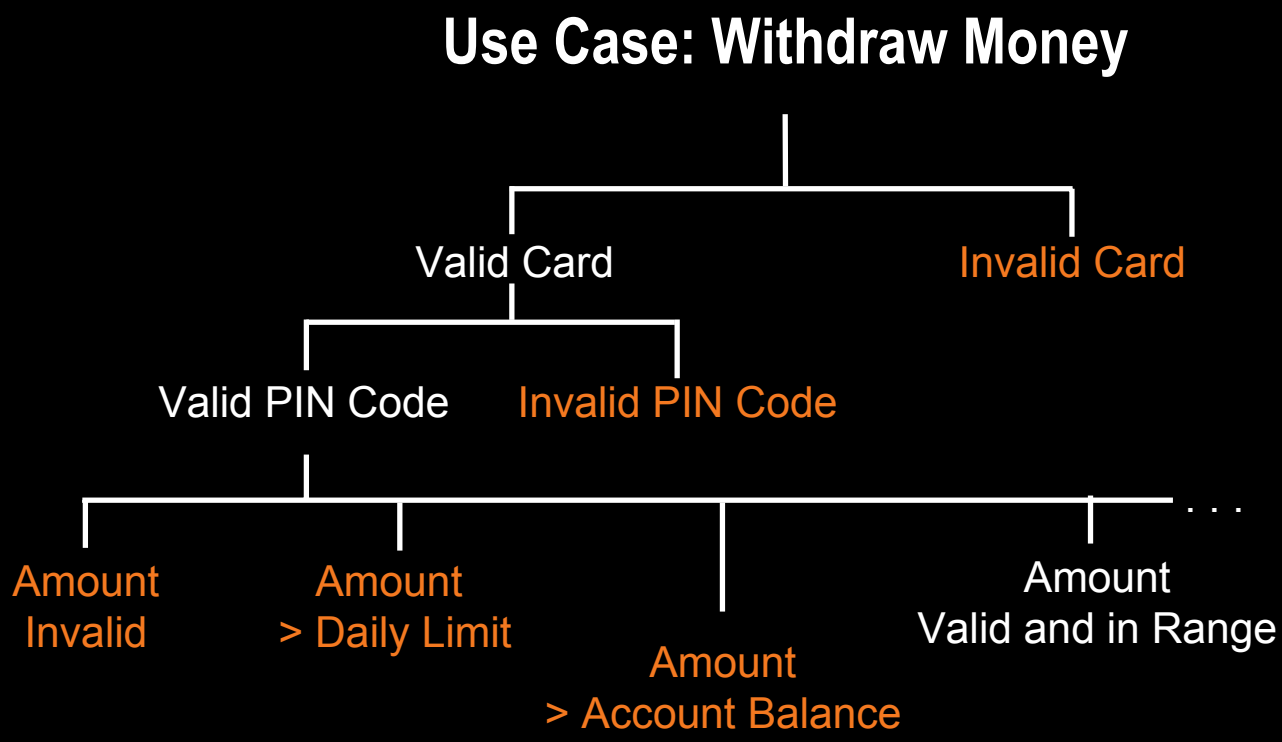
- Use Case Modeling Done! → Plan Testing & Define Test Cases
- Design Done! → Generate Test Cases From Sequence diagrams and State-Chart diagrams
- Basis for the Test Specification

# Deriving Test Cases from Use Cases

- Step one: Identify all end-to-end paths through the use case (identify all use case instances)
- Step two:  For each identified/enumerated path create one or more test cases
- Step three: Add data values for the conditions in the test cases



# Identifying Use Case Instances



# Withdraw Money – Use Case Instance Matrix

| Use Case Instance Name | Starting Flow | Alternate Flow | Pre-cond: Card Valid | Pre-cond: PIN Valid | Pre-cond: Amount | Post-cond. |
|---|---|---|---|---|---|---|
| UCi1- Invalid card | Basic Flow | A1 | N | | | Eat card |
| UCi2- Invalid PIN code | Basic Flow | A2 | Y | N | | Eat card |
| UCi3- Invalid amount | Basic Flow | A4 | Y | Y | invalid | Cashier Class rejects |
| UCi4- Exceeds daily limit | Basic Flow | A4 | Y | Y | > daily limit | Cashier Class rejects |
| UCi5- Requested amount exceeds account balance | Basic Flow | A5 | Y | Y | > account balance | Withdrawal Class rejects |
| UCi6- Successful withdrawal | Basic Flow | - | Y | Y | valid & in range | Accept; Debit Acct; Dispense |

# System (Black Box) Test is Use Case Test

- Operation Test
  - Long duration, MTBF
- Full Scale Test
  - Maximum Load
- Performance Test
- Stress test
  - Extreme Loads
- Negative Test – abuse cases
  - Try to break the system, find weak spots

# Summary

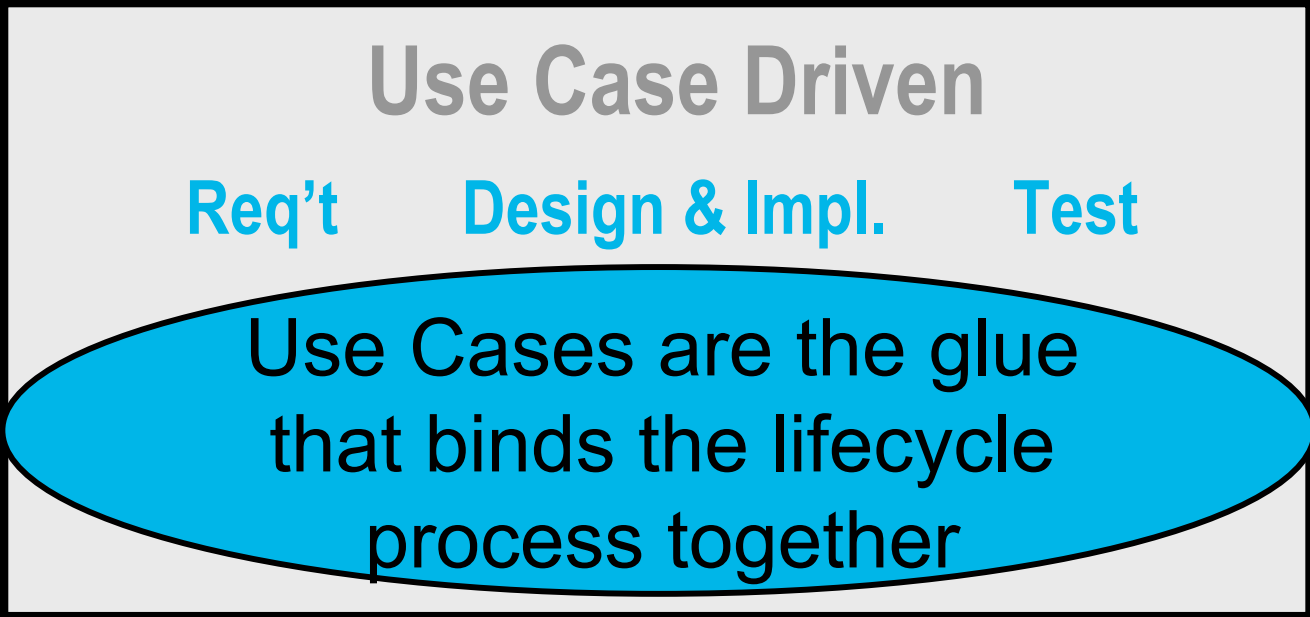
# The Role of Use Cases

- Use cases capture most of the requirements
  - All functional requirements
  - Also non-functional requirements specific for a use case, for instance response times, performance, many utilities
- The use cases drive the development through all activities (iteration by iteration):
  - Architecting the system
  - Design
  - Implementation
  - Test

# The Role of Use Cases cont'd

- Envisioning the system
- Communication between different parties
- Designing user interfaces
- Determining development increments
- Tracing requirements
- Estimating project size and resources
- Defining database-access patterns
- Dimensioning system properties and capacity
- Facilitates reuse

# In One Sentence




**You can't do without it!**

# For More Information

- www.rational.com
- The UML Books (*Booch, Jacobson, Rumbaugh with Addison Wesley*)
  - *The UML User Guide*
  - *The UML Reference Manual*
  - *The Unified Software Development* Process

# Other Readings by Ivar Jacobson

- Object-Oriented Software Development--A Use Case Driven Approach (Addison Wesley)
  Jacobson et al, Addison Wesley Longman (1992)
- The Object Advantage: Business Process Reengineering with Objects (Addison Wesley)
  Jacobson et al, Addison Wesley Longman (1994)
- Software Reuse: Architecture, Process and Organization for Business Success (Addison Wesley)
  Ivar Jacobson, Martin Griss & Patrik Jonsson, Addison Wesley Longman (1997)
- Unified Software Development Process
  Jacobson, Booch, Rumbaugh, Addison Wesley Longman (1999)
- The Road to the Unified Software Development Process
  Ivar Jacobson, Stefan Bylund, Cambridge University Press, 2000