

2. Requirements Capture

- First activity of the process
- Goal: find out what to build
- Communication between users and developers
 - thus: no complex notations possible (unless scientific domain)
 - mostly natural language
 - contract
- Methods for finding requirements
- Methods for formulating requirements
 - Scenarios, Use Cases, Mockups / Prototypes, Feature Lists
- Stakeholders
 - everybody interested in the product

Sources of Requirements: Customers

- Interviewing customers
 - the people who pay you
 - other stakeholders
 - users
 - managers
- Problems:
 - customers may not know what they want / need
 - software is abstract and complex
 - may often change their minds
 - may not be able to express their needs in technical terms
 - communication between computer scientists / regular people
- Techniques
 - mockups, prototypes
 - walkthroughs
 - differences to existing systems

Sources of Requirements: Market

- Evaluating competing products
 - what has been done before?
 - where is a niche for us?
 - but take care not to violate copyrights, trademarks, patents
- Evaluating own abilities
 - what are we better at than the competition?
 - what knowledge, skills do we have?
- Market survey
 - interview consumers
 - difficult to do right; polling companies
 - marketing and advertising: create the demand
 - consider future trends
- Problems:
 - people don't know what they want
 - example: video telephones
 - markets change quickly (UMTS)

Sources of Requirements: Standards

- Standards and interoperating systems
 - system standards, file formats, network protocols
 - usability standards
 - domain standards
- Official standards
 - written by a standards body
 - ANSI
 - ISO (e.g. Unicode)
 - IEEE (e.g. Posix)
- Industry standards
 - Wintel Platform
 - Java, Dot-Net
 - Wimp user interface

Kinds of Requirements

- Functional
 - features
 - user interface
 - input / output
- Non-functional
 - user qualities
 - performance
 - precision
 - reliability
 - developer qualities
 - maintainability
 - reusability
 - interoperability

Feature Lists

- List of properties of the system
 - taken from similar products, earlier versions
 - taken from user's requests
- Functional features
 - "A dialog to enter the user's address"
 - "can calculate wage deductions"
- Nonfunctional features
 - "runs on Linux..."
 - "responds in less than 1 second"
 - technical requirements
 - supported standards
 - required amount of memory
- Format
 - numbered list
 - sorted by topic

Scenarios

- A sequence of events describing an interaction between the program and a user
- Example:

Successfully retrieve cash from ATM

1. User enters ATM card
 2. ATM asks for PIN
 3. User enters PIN
 4. ATM displays menu
 5. User selects “Get cash - \$100”
 6. ATM disburses 5 bills for \$20
 7. User takes cash and ATM card
- Scenario captures a typical event sequence

Mockups / Prototype

- A draft of the user interface
- Used to get feedback from users
- Talking to users in their terms
- May be realized as:
 - a running prototype (“rapid prototyping”)
 - a sketch on paper
- Prototype can also contain some functionality
- Tool: Gui builders
 - for example contained in MS Visual Studio
 - allow quick development of a running Gui that doesn’t do anything
 - uses drag-and-drop, requires almost no technical skill
 - limitations: non-standard Guis, dynamic elements

Use Cases

- Set of related scenarios with a common goal
- For example:
 - Successfully retrieve cash from ATM
 - Failed attempt to retrieve cash because ATM out of service
 - Failed attempt to retrieve cash because of insufficient funds
 - Failed attempt to retrieve cash because of invalid ATM card
 - Failed attempt to retrieve cash because of invalid PIN

⇒ **grouped together as use case “Retrieve cash from ATM”**
- Elements:
 - Describe main success scenario (numbered steps)
 - List exceptions and variations separately
 - Short title that shows main goal, using imperative form
 - Start state, end state, additional conditions

Example of a Use Case

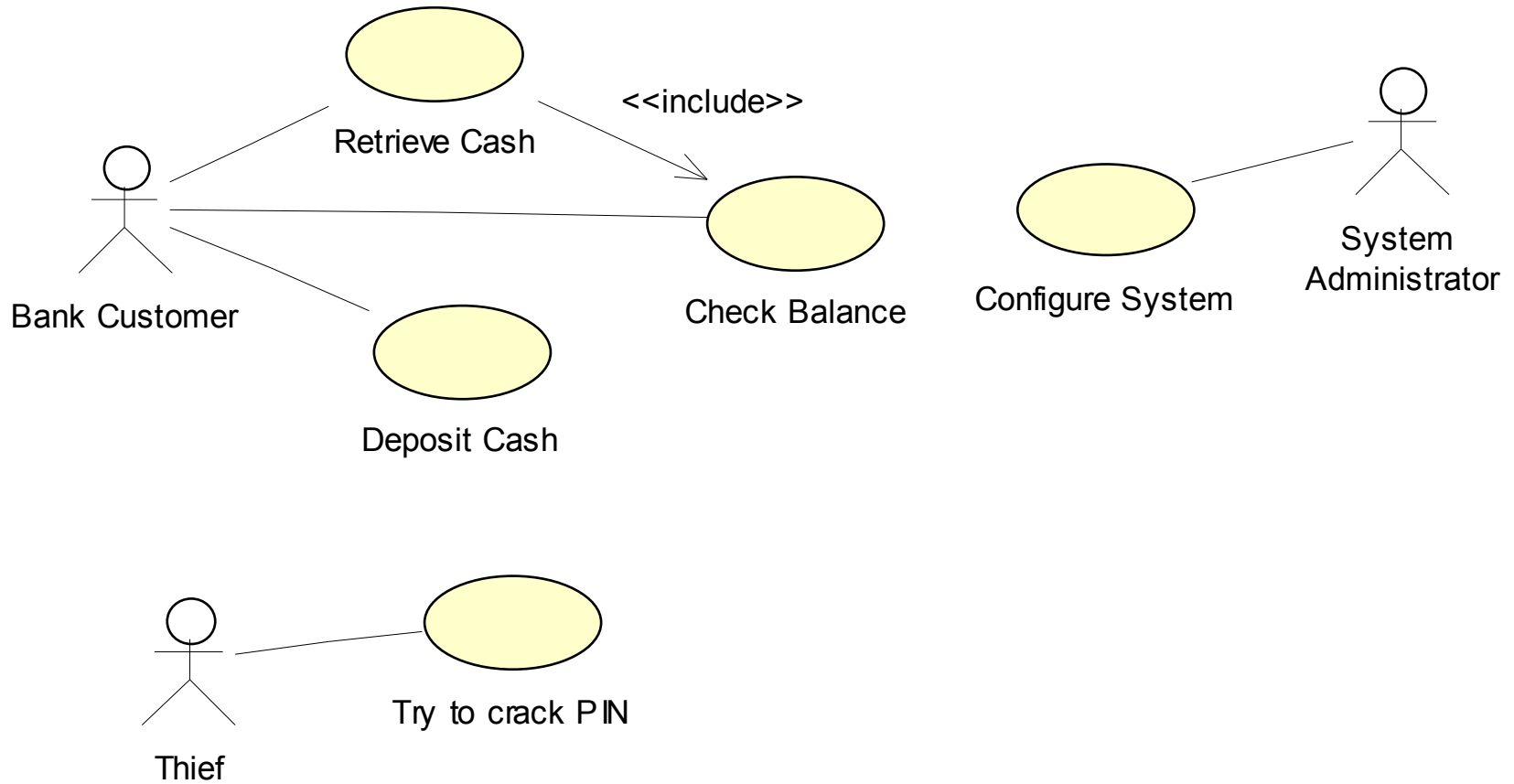
Retrieve cash from ATM

1. User enters ATM card
2. ATM asks for PIN
3. User enters PIN
4. ATM displays menu
5. User selects “Get cash - \$100”
6. ATM disburses 5 bills for \$20
7. User takes cash and ATM card

Variations:

- 1a. ATM says card is invalid or ATM out of service, end.
- 3a. ATM says PIN is invalid, repeat 3.
5. User may select \$20, \$40, or \$100. All amounts will be disbursed in \$20-bills.
- 5a. ATM says user’s account has insufficient funds, end.

Use Case Diagram



Use Case Diagrams

- UML diagrams to show relationships between use cases and actors
- Actor
 - a role that a person (or machine) has when interacting with the program
- Use case relations
 - include: one use case includes another
 - other relations exist
 - but: most use cases are unrelated
- Really simple so that users don't get scared
 - no sequence, no control-flow
- Very imprecise (like most things in requirements)

ArgoUML - Untitled

File Edit View Create Diagram Arrange Generation Critique Tools Help

Package-centric

- untitledModel
 - class diagram 1
 - use case diagram 1
 - IAmAnActor
 - a Use Case
 - another One
 - (anon Association)
 - (anon Association)

Use Case Diagram: use case diagram 1

```

graph LR
  Actor[IAmAnActor] --- UC1(a Use Case)
  Actor --- UC2(another One)
  
```

As Diagram

By Priority 1 Item

- High
- Medium
- Low

To Do Item
 Properties
 Documentation
 Style
 Source
 Constraints
 Tagged Values
 Checklist

No To Do Item selected

Back Next Finish Help

Qualities of Requirements: Unambiguity

- Clear, understandable
- For both you and your customer!
- Many computer science terms are ambiguous to others
 - example: “a or b”

in CS:	can be: a and b
rest of the world (often):	a or b, but not both
- Most English words have many different meanings
- Grammar can often be parsed in different ways
 - example (from Michael Jackson):

Signs at an escalator:

“Shoes must be worn”

“Dogs must be carried”

Qualities of Requirements: Completeness

- You need to plan the project
 - hard if you don't know all the requirements
 - at least for the current increment
- Missing requirements...
 - may change the time plan
 - may change the architecture
 - may change risk
- Reasons for incompleteness
 - customers changing their minds
 - bad communication
 - forgot features needed by other features
 - example: user administration