## Requirements engineering

- "Requirements engineering is a systematic way of developing requirements through an iterative process of analyzing a problem, documenting the resulting observations, and checking the accuracy of the understanding gained." – IEEE Standard
- Software requirements engineering provides the techniques for:
  - -understanding what a customer wants
  - analyzing it
  - assessing feasibility
  - negotiating a reasonable solution
  - specifying the solution unambiguously
  - validating the specification
  - managing the requirements as they are transformed into an operational system

## Roles of requirements

#### Customers

-show what should be delivered; contractual base

#### Managers

—a scheduling / progress indicator

### Designers

—provide a spec to design

### Programmers

—list a range of acceptable implementations / output

### • QA / testers

—a basis for testing, validation, verification

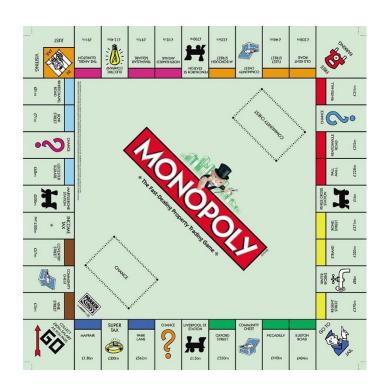
## Requirements classification

- Functional: specify a function that a system or system component must be able to perform
  - —"The user can search either all databases or a subset."
  - -"Every order gets an ID the user can save to account storage."

- Nonfunctional: are not concerned with the functionality of a system but place restrictions on the product
  - —dependability, reusability, portability, scalability, performance, safety
  - -"Our deliverable documents shall conform to the XYZ process."
  - —"The system shall not disclose any personal user information."

## **Example: Functional requirement**

- When a player passes or lands on the Go cell, the player shall get paid \$200.
- When a player lands on the Free Parking cell, nothing shall happen.
- When a player lands on an available property cell, the player shall have a chance to purchase it. The price shall be the land value of that property.



## Non-functional requirements

### Security:

- -protection of information and data
- —unauthorized persons or systems cannot read or modify them
- —authorized persons or systems are not denied access to them
- **Privacy**: "the right to be let alone."
  - -Which information can be collected
  - What must not be stored
- **Usability**: the ease with which a user can
  - -learn to operate
  - prepare inputs
  - —interpret outputs

## Non-functional requirements

- Reliability: the ability to perform required functions under stated conditions
  - -Types: low, medium, high
  - —Example: Cell-phone
    - Call: high reliability
    - Playing games: low reliability
- Availability: the degree to which a system is operational and accessible
  - -99.99% availability
- **Performance:** measures how a system accomplishes its designated functions within given constraints
  - -Speed
  - -Accuracy
  - -Memory usage

## **Example: Non-functional requirement**

- The system must conform to ISO/IEC 27034-1:2011 security standard.
- An user should receive search results within 0.01 seconds with more than 80% accuracy.
- The system must be 100% operational 99.9% of the calendar year during its first year of operation.

## **Stakeholders**

 Any person or group who will be affected by the system, directly or indirectly

- Examples
  - —End users
  - —System administrators
  - Engineers maintaining the system
  - Business managers

## Stakeholders' viewpoints

- Different set of requirements for different types of stakeholders
- Interactor viewpoints
  - People who interact with the system
- Indirect viewpoints
  - —People who influence the system's requirements in some other way
- Domain viewpoints
  - Domain characteristics and constraints

## **Viewpoints: Bank ATM**

- Interactor viewpoints
  - -Consumer
  - Bank manager (policy settings)
  - -People who fill up the ATM
  - -Bank server
- Indirect viewpoints
  - —Bank manager (policy settings)
  - —Federal reserve
  - Thieves or security managers
- Domain viewpoints
  - Laws governing ATM use
  - —Agreements with other banks

## How to gather requirements?

- Interviews
  - —Talk to the users
  - —Ask questions
- Observation
  - Observe current business activities
  - —How current system works
- Examine documents and artifacts
  - Ideas on automating manual forms
  - —Ask security / privacy policies
- Alternate scenarios
  - —Failures

## How to gather requirements?

- Questionnaire
  - -Clarification questions
  - -Goal / purpose
- Prototypes
  - —Paper mock-ups/ screenshots
  - -Static websites
- Onsite customer
  - -Working with the requirement team
- Brainstorm
  - —Think what users might want

# **How \*NOT\* to gather requirements?**

- Describe complex business logic or rules of the system.
- Be too specific or detailed.
- Describe the exact user interface used to implement a feature.
- Try to think of everything ahead of time. (You will fail.)
- Add unnecessary features not wanted by the customers

## Requirements validation

- Concerned with demonstrating that the requirements define the system that the customer really wants.
- Requirements error costs are high so validation is very important
  - —Fixing a requirement related error after delivery may cost up to 100 times the cost of fixing an implementation error.

## Requirements checking

- Validity. Does the system provide the functions which best support the customer's needs?
- Consistency. Are there any requirements conflicts?
- Completeness. Are all functions required by the customer included?
- Realism. Can the requirements be implemented given available budget and technology
- Verifiability. Can the requirements be checked?

## Requirements validation techniques

- Requirements reviews
  - —Systematic manual analysis of the requirements.

- Prototyping
  - —Using an executable model of the system to check requirements.
- Test-case generation
  - Developing tests for requirements to check testability.

## Requirements reviews

- Regular reviews should be held while the requirements definition is being formulated.
- Both client and contractor staff should be involved in reviews.
- Reviews may be formal (with completed documents) or informal. Good communications between developers, customers and users can resolve problems at an early stage.

### **Review checks**

- Verifiability
  - —Is the requirement realistically testable?
- Comprehensibility
  - —Is the requirement properly understood?
- Traceability
  - —Is the origin of the requirement clearly stated?
- Adaptability
  - —Can the requirement be changed without a large impact on other requirements?

## Good or bad requirements?

- The system will enforce 8.5% sales tax on Illinois purchases.
- The system shall display the elapsed time for the car to make one circuit around the track within 5 seconds, in hh:mm:ss format.
- The product will never crash. It will also be secure against hacks.
- The server backend will be written using PHP or Ruby on Rails.
- The system will support a large number of connections at once, and each user will not experience slowness or lag.
- The user can choose a document type from the drop-down list.