

# CS3773 Software Engineering

## Lecture 02 Software Requirements

## Requirements Engineering

- Requirements engineering is usually the first stage of software life cycle
- Requirements engineering is the process of understanding and defining functionalities and constraints of proposed systems
- Requirements engineering process produces a document, software requirements specification (SRS)
  - Customers need a high level specification
  - Software designers and developers need a more detailed specification

2

UTSA CS3773

## Software Requirements

- Requirements are desired behaviors
  - Customers "know" what the system shall do
  - Software engineers "know" what to build
- "Requirements are means of communication with customer and many other stakeholders"
  - by Helene Wong, PhD thesis, 1994
- Requirements deal with
  - Objects
  - States
  - Functions

3

UTSA CS3773

## Software Requirements Stakeholders

- Requirements analysts or system analysts determine requirements
- Stakeholders contribute to requirements of systems
  - Clients
  - Customers
  - End-users
  - Software engineers
  - Domain experts
  - Lawyers or auditors
  - Market researchers

4

UTSA CS3773

## Types of Requirements

- Functional
  - What is the system supposed to do
  - Mapping from input to output
- Non-functional (quality)
  - Usability
  - Performance
  - Security
  - Reliability
  - Maintainability
  - Portability

5

UTSA CS3773

## Types of Requirements

- Process constraints
  - Resources
  - Documentation
  - Standards
- Design constraints
  - Physical environment
  - Interface
  - Users

6

UTSA CS3773

## Requirements Are Important

The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements, including all interfaces to people, to machines, and to other software systems. No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later.

-- by Frederick Brooks, "No silver bullet: essence and accidents of software engineering", 1986.

7

UTSA CS3773

## Requirements Are Important

- 80% of all software errors are requirements errors
  - These are software errors detected after unit testing - i.e., in integration testing, in system testing, and after the software is released
  - Most errors can be traced to unknown, wrong, or misunderstood requirements

8

UTSA CS3773

## Requirements Are Important

- Requirements usually affect large portions of the implementation; they are rarely encapsulated into modules
- Requirements errors may be fundamental assumptions built into the design or code
- Expensive requirements errors are often not fixed; they become "features"

9

UTSA CS3773

## Requirements Are Important

- Requirements errors are expensive to fix

Stage discovered	Relative repair cost
Requirements	0.1 – 0.2
Design	0.5
Coding	1
Unit test	2
Acceptance test	5
Maintenance	20

10

UTSA CS3773

## Requirements Problems

- Over-specification
- Under-specification (unintended)
- Contradictory requirements
- Ambiguous requirements
- Unknown requirements
- Bad assumptions about environment
- Changing requirements

11

UTSA CS3773

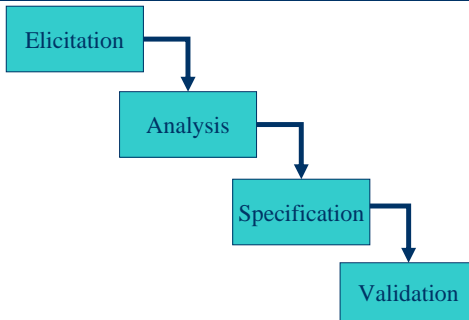
## Characteristics of Requirements

- Correct
- Consistent
- Complete
- Concise
- Traceable
- Unambiguous
- Understandable
- Verifiable

12

UTSA CS3773

## Requirements Engineering Process



13

UTSA CS3773

## Requirements Engineering Process

- Determine the requirements of a system, and specify what behavior is realized
  - Work with customers to elicit the requirements
  - Analyze and model the requirement
  - Document the requirements in a software requirements specification
  - Validate the requirements specification

14

UTSA CS3773

## Requirements Tasks

- Understand problem from each stakeholder's point of view
- Extract the essence of the stakeholders' requirements
- Negotiate a consistent set of requirements with agreement from all stakeholders; set relative priorities
- Record results in an SRS

15

UTSA CS3773

## Requirements Elicitation

- Elicitation is to gather
  - Functions that the system should perform
  - Non-functional requirements that the system should exhibit
- Elicitation is critical but difficult
  - Customers are not good at describing what they want
  - Software engineers are not good at understanding what customers want
  - Customers and software engineers speak different languages

16

UTSA CS3773

## Requirements Elicitation

- Requirements analysts have to understand the problem from each stakeholder's point of view
  - Stakeholders have different views of the system
- Requirements analysts resolve conflicting views
  - Essential requirements
  - Desirable requirements
  - Optional requirements

17

UTSA CS3773

## Elicitation Techniques

### Understand problems

- For existing system
  - Review documentation
  - Observe current system
  - Questionnaires and Interviews
  - Apprenticeship
- For new systems - brainstorming

18

UTSA CS3773

## Analyze Existing System

- What is used, what isn't, what's missing
- What works well, what doesn't
- How the system is used, how it was intended to be used, what new ways we want it to be used
- Risks
  - Users might not be happy with too much change from the old system
  - Might miss real usage patterns
  - Might miss obvious possible improvements

19

UTSA CS3773

## Analyze Existing System - Review

- Review all available documentation
  - For an automated system, review its requirements specifications and user manuals, as well as development documentation, internal memos, change histories, etc.
  - For a manual system, review any documented procedures that the workers must follow
- Gain knowledge of the system before imposing upon other people's time, before bothering the stakeholders

20

UTSA CS3773

## Analyze Existing System - Observation

- Identify what aspects to keep and to understand the system you are about to change
- System contains a lot of useful functionality that should be included in any future system
- Documentation rarely describes a system completely and not up to date and
- Current operation of the system may differ significantly from what is described

21

UTSA CS3773

## Analyze Existing System - Interview

- Questionnaires are useful when information has to be gathered from a large number of people
- The answers to questions need to be compared or corroborated.
- Ask problem-oriented questions during interview
- Interview groups of people together to get synergy

22

UTSA CS3773

## Analyze Existing System - Apprentice

- The requirements analyst is the apprentice and the user is the master craftsman.
- The user can
  - Describe the task precisely
  - Explain why the task is done this way
  - List the exceptions that can occur

23

UTSA CS3773

## Brainstorm

- Brainstorm is used to gather ideas from every stakeholder and prune ideas
- When you have no idea, or too many ideas, sit down and thrash it out, but with some ground rules
- Most useful early on, when terrain is uncertain, or when you have little experience, or when novelty is important

24

UTSA CS3773

## Brainstorm

- Keep the tone informal and non-judgmental
- Encourage creativity
- Keep the number of participants "reasonable", if too many, consider a "playoff"-type filtering
- Invite back most creative to multiple sessions

25

UTSA CS3773

## Brainstorm - the Storm

- Generate as many ideas as possible
- Quantity, not quality, is goal at this stage
- No criticism or debate is permitted
- Write down all ideas where all can see
- Participants should NOT self-censor or spend too much time wondering if an idea is practical
- Original list does not get circulated outside of the meeting

26

UTSA CS3773

## Brainstorm - the Calm

- Go over the list and explain ideas more carefully
- Categorize into "maybe" and "no" by pre-agreed consensus method
- Be careful about time
  - Meetings tend to lose focus after 90 to 120 minutes
- Review, consolidate, combine, clarify, and expand
- Rank the list by priority somehow; choose a winner

27

UTSA CS3773

## Brainstorm - Pruning

- Vote with threshold
  - Each person votes up to  $n$  times
  - Keep those ideas with more than  $m$  votes
  - Have multiple rounds thereof with smaller  $n$  and  $m$
- Vote with campaign speeches
  - Each person votes up to  $j < n$  times
  - Keep those ideas with at least one vote
  - Have multiple rounds thereof with smaller  $j$

28

UTSA CS3773

## Reading Assignments

- Sommerville's Book
  - Chapter 6, "Software Requirements"