The Essential Software Requirement

LECTURE # 1
Chapter 1 – Karl Wiegers
Chapter 1,2 - Reference

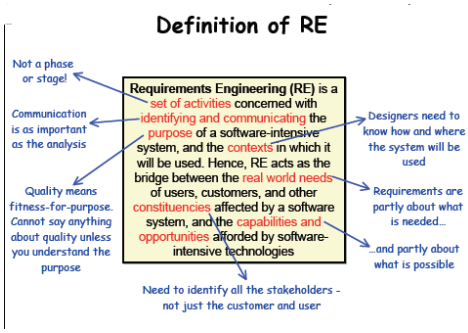
SOME TRUE PROBLEMS

- Using a software product that doesn't let you perform essential task is frustrating.
- Learning of functionality after the system has just been implemented is again frustrating.
- Interruption for modifying the request during the project to replace something told before.

CONSIDERATION

- Expectation Gap
- Cost for change
- 40 – 60% errors in s/w due to requirement process

Definition of RE



Stakeholders

- Customers & User
- Requirement / System Analyst
- Designers, Developers &Testers
- Documentation Writers
- PM
- Legal Staff –
- Manufacturing People, Sales, Marketing..
- Subject Matter Experts / SME

RECAP

- Waterfall v/s Iterative
- SDLC
- FAST Methodology
- RAD, IE, SAD, RUP
- Prototyping
- Agile Methodology

Initiation of Problems

- Project scope & vision not clearly defined
- Busy customers
- User surrogates
- Customers say all requirements critical & don't prioritize

Initiation of Problems

- Ambiguities & missing information
- Signoff requirements but still change
- Requested changes get lost and the team & user doesn't know the status of request
- Specs are satisfied but customer isn't

RULE!!!!!

- Always document the requirement

WHAT ARE REQUIREMENTS ?

- Requirements are specifications of what should be implemented. They are descriptions of how the system should behave or of a system property or attribute. They may be a constraint on the development process of the system.

LEVELS OF REQUIREMENTS

- Three levels besides nonfunctional requirements
- Business Requirements – High level objectives of org/customer who requested the system.
  - WHAT ARE THE GOALS/ OBJECTIVES?
- User Requirements – Describe what the user will be able to do with the system
- Functional Requirements: Also called behavioral requirements… Developers must build enable users to do tasks

REQUIREMENTS

- System requirements are a part of functional requirements
- Business rules – policies, procedures, regulations
- SRS – documents functional requirements & non functional requirements

NON FUNCTIONAL REQUIREMENTS

- Performance Goals & Quality Attributes
- Quality Attributes – usability, portability, integrity, efficiency, robustness
- External Interfaces b/w system & outside world
- Design & Implementation constraints
- Even business rules

Software requirements

- **requirements**: specify what to build
  - tell "what" and not "how"
  - tell the system design, not the software design
  - tell the problem, not the solution (in detail)
- What are some goals of doing requirements?

1. understand precisely what is required of the software
2. communicate this understanding precisely to all development parties
3. control production to ensure that system meets specs (including changes)

Requirements abstraction

"If a company wishes to let a contract for a large software development project, it must define its needs in a sufficiently abstract way that a solution is not pre-defined. The requirements must be written so that several contractors can bid for the contract, offering, perhaps, different ways of meeting the client organization's needs. Once a contract has been awarded, the contractor must write a system definition for the client in more detail so that the client understands and can validate what the software will do. Both of these documents may be called the requirements document for the system."

Software lifecycle, review

- The software lifecycle (Faulk's view):

```
requirements -> system testing -> deployment ->
maintenance
/                          ^
/                          /
+- preliminary design -> integration testing
   /                       ^
   /                       /
   +- detailed design -> unit testing
      /                    ^
      /                    /
      +- coding ----------+
```

Requirement roles to people

- roles of requirements
  - customers: show what should be delivered; contractual base
  - managers: a scheduling / progess indicator
  - designers: provide a spec to design
  - coders: list a range of acceptable implementations / output
  - QA / testers: provide a basis for testing, validation, and verification

Classifying requirements

- Faulk doesn't like the normal way to classify requirements, which is the following:
  - **functional**: map inputs to outputs
  - **nonfunctional**: other constraints
    - performance, dependability, reusability, safety
- How does Faulk prefer to classify them?
- **behavioral**: everything about implementation
  - features, performance, security
  - can be objectively observed / measured
- **development quality attributes**: things about internal construction
  - flexibility, maintainability, reusability
  - subjective, relative; who says what design is more maintainable?

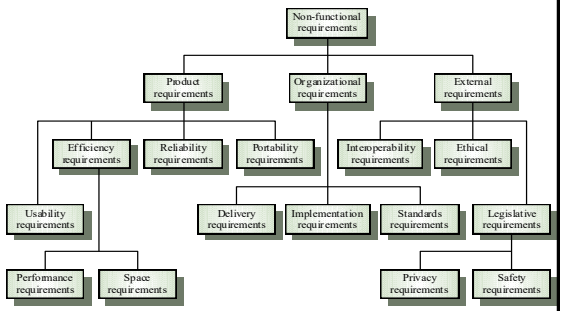
Functional requirements

- Examples of functional requirements:
  - The user shall be able to search either all of the initial set of databases or select a subset from it.
  - The system shall provide appropriate viewers for the user to read documents in the document store.
  - Every order shall be allocated a unique identifier (ORDER_ID) which the user shall be able to copy to the account's permanent storage area.

Non-functional requirements

- Examples of non-functional requirements:
  - It shall be possible for all necessary communication between the APSE and the user to be expressed in the standard ASCII character set.
  - The system development process and deliverable documents shall conform to the process and deliverables defined in XYZCo-SP-STAN-95.
  - The system shall not disclose any personal information about customers apart from their name and reference number to the operators of the system.

Non-functional requirements



Requirements example

- Identify the requirements in the following text
  - We will implement our bank teller ATM software in Java. It should handle cash withdrawals and deposits in under 5 seconds wait time. It should be done in such a way that we can adapt it to our other types of ATMs and machines for our bank. We will use encrypted network connections to avoid hackers spying on users' account information.

Some requirement measures

Property	**Measure**
Speed	Processed transactions/second User/Event response time Screen refresh time
Size	K Bytes Number of RAM chips
Ease of use	Training time Number of help frames
Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target dependent statements Number of target systems

Essential difficulties

- What does **"essential" difficulty** mean?
  - something that is hard about reqs, by nature
- What are some of the essential difficulties?
  - comprehension: people don't know what they want
  - communication: hard to specify clearly what is wanted
  - control: can't see which requirements will dominate schedule
  - inseparable concerns: can't divide up problem, or freeze from change
  - must compromise on divide-and-conquer, and make tough trade-offs

Accidental difficulties

- What does **"accidental" difficulty** mean?
- something made difficult by writing reqs poorly
- What are some of the accidental difficulties?
  - written as an afterthought: devs write requirements after coding, instead of at beginning
  - confused in purpose: has too much marketing info, too imprecise (to let the customer change it later), or has too much design/implementation info in it
  - not designed to be useful: no time or thought put in; makes requirements poor and useless
  - lacks essential properties: incomplete

Problem analysis, basic issues

- This phase describes the problem that must be solved by the software we will produce
- elicit requirements from customer
  - how might this be done?
- decompose problem into pieces
- organize info, communicate to involved parties
- resolve conflicting needs
  - what is an example of some conflicting needs?
- know when to stop

Requirements exercise

- Let's sketch out some requirements for a bank ATM software program. This is the software that appears on the screen of the ATM and walks you through deposits and withdrawals.

  With a partner, come up with 4 requirements for such software, that you think are important. Try to be specific. Write them down and classify each as functional/non-functional or behavioral/development quality, to the best of your ability. Then we'll discuss them together.

END OF LECTURE # 1

-COMING UP!!!!!!
-Requirement Engineering Process
-Requirements from the customer perspective
- Good Practices for requirement engineering

28