Chapter 4

Requirement Engineering

-Requirement analysis

-Requirement use case

-Requirement specification

-Requirement Validation

Requirement Engineering

- The art and sciende of developing an accurate and complete difinition of the behavior of software that can serve as the basis for software development

$$\begin{tikzcd}
\text{Requirements elicitation and analysis} \arrow[r] & \text{Use case} \arrow[r] & \text{Requirement specification} \arrow[r] & \text{validation}
\end{tikzcd}$$

this reposibility is for Requirement analyst or business system analyst

change control

Definition

Elicitation: discovering requirements to define problem and scope

Specification: converting the requirement into standard form

Validation: checking if the requirement actually define that need

Change control: handling changes that happen all over the project

Requirement Gathering Techniques

- interviews
- prototyping
- Focus group
- Workshop
- Document analysis
- Surveys
- Observation

Interviews (are not good way to reach consensus)

open ended questions to find information and gaps

- what does the current system look like?
- What are the challenges
- How do you see the solution

Close end questions to confirm and validate

- are you satisfied with this product

Focus group (save time and cost by not having many interview)

- Elicit information from a select group via facilitator
- very formal process
- Usually 6 - 12 attendees
- Engage all members
- Remain neutral
- Promote discussion

prototype

- model, sample.
- Visually represents UI

Survey

- gathering data from a large group

Document analysis

- Existing documentation
- Stake holder and experts are not availble

Discution Summary

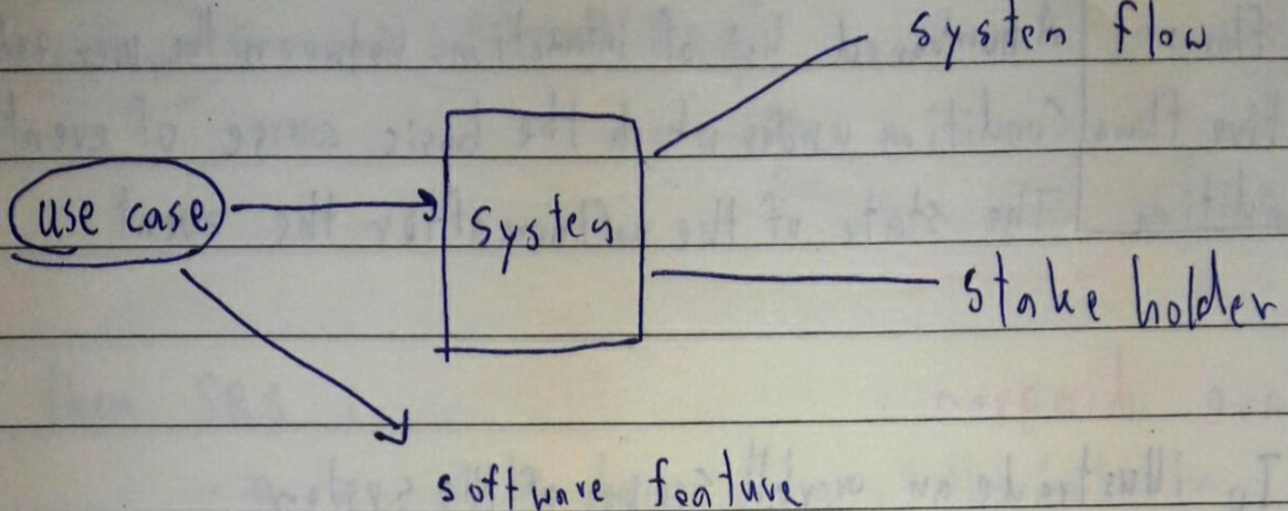
- project backgro
- Perspective
- Objective
- Risk
- Future enhance
- Reference
- TBD issue

observation

- study a stakeholder's work environment
- explore user work flow
- Identify ways to enhance and streamline

Use case

- Requirement tool for describing the behavior of the software
- textual description explaining the way users interact with the software.



Develop the use case

- Identify the basic features that will be developed
- how software should behave

Analysis of library management syste

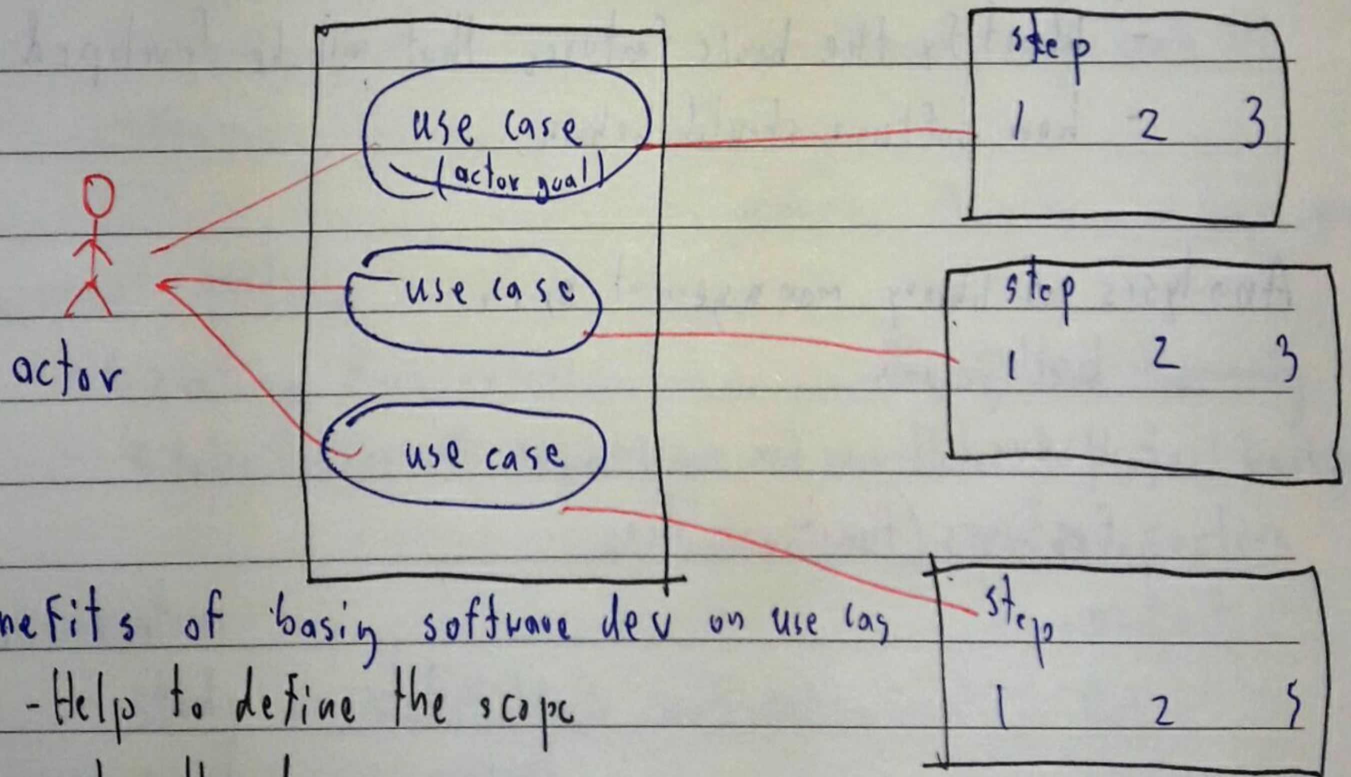
- background
- Users
- Features / functionalities

Use case Template

Name	Use case number and name
Description	Brief description of the use case and why it needed
User	A list of all the categories of user that interact with
pre condition	The state of the software before the use case begin
Normal flows	A numbered list of interaction between the user and softw
Alternative flows	Condition under which the basic course of event
Post condition	The state of the software after the event

Use case diagram

- To illustrade an overall scope of the systen
- actor specifies a role played by a user or any other system
- a use case describe a scenario of how user interact

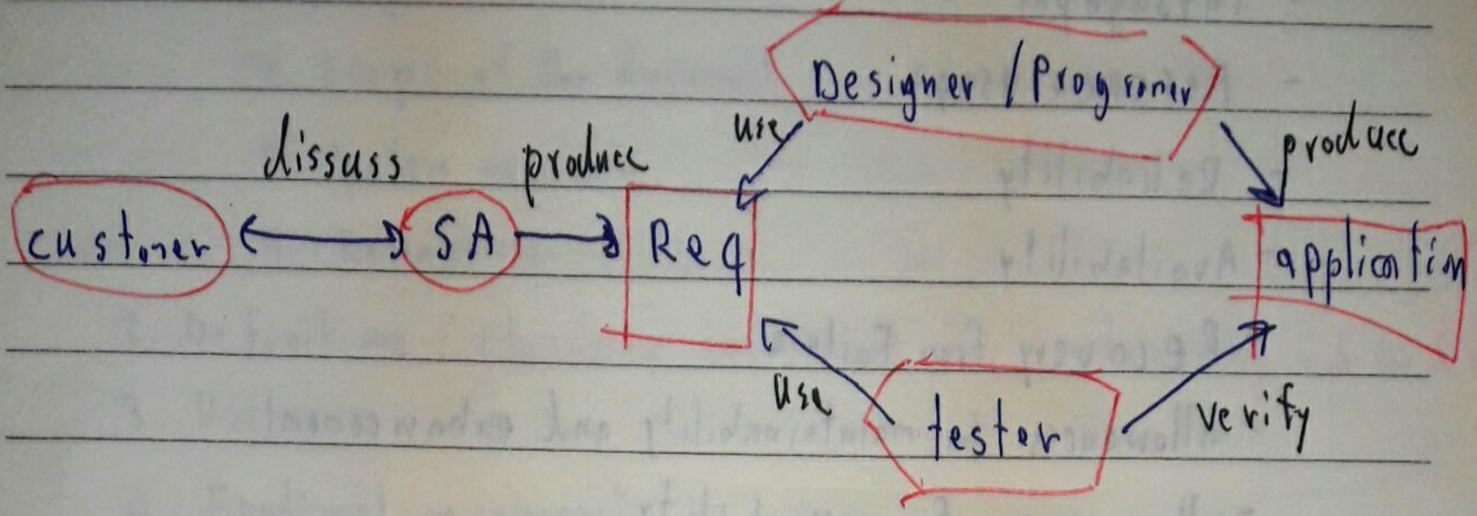


benefits of basing software dev on use cas

- Help to define the scope
- plan the dev process
- used in dev and validate
- Form test case
- structure user manuals

Software Requirement Specification (SRS)

draw ~~on~~ Software Requirement in picture



then SRS is

- a complete description of the behavior of the software to be developed
- There are 2 types of SRS
    - functional requirement
        - internal working of the software
    - Non - Functional requirement
        - constraints on the design or implementation

Functional-Requirement	Non-functional-Requirement
- I/O of the system	- UI/UX
- Data that system use	- Response time
- Where Data should store	- Size of data base
- Computations the system	- hour of system work per day
- timing and synchronization	

Non Functional requirement Quality requirements

- Response time
- Throughput
- Resource usage
- Reliability
- Availability
- Recovery from failure
- allowances for maintainability and enhancement
- allowances for reusability

• all must be verifiable

Nonfunctional requirement Platform requirement

- Platform
- Technology to be use

Nonfunctional requirements Process requirement

- Development process (methodology) to be use
- Cost and delivery date
- often put in contract or project plan instead

SRS template

1. Introduction
    1. purpose of the document
    2. scope of the document
    3. System overview
    4. References
2. Definitions (glossary term that the reader may ~~not~~ not be famil
3. Use case
4. Fundional requirement
5. Non functional requirement

Functional and non functional requirement template

Name: Name and number of the functional requirement

Summary: Brief description of the requirement

Rationale: Description of the requirement and why it needed

Requirement: The behavior that requireed of the softwara

References: Use case and other functional and non functional

Requirement Validation

- remove many defects as possible
- defect is any planned software behavior, a project team member user, stakeholder, desision makr

Requirements documents

- sufficiently complete
- well organized
- clear
- agreed to by all the stakeholders
- traceability

avoid overly specifi selection

Too specific requirement

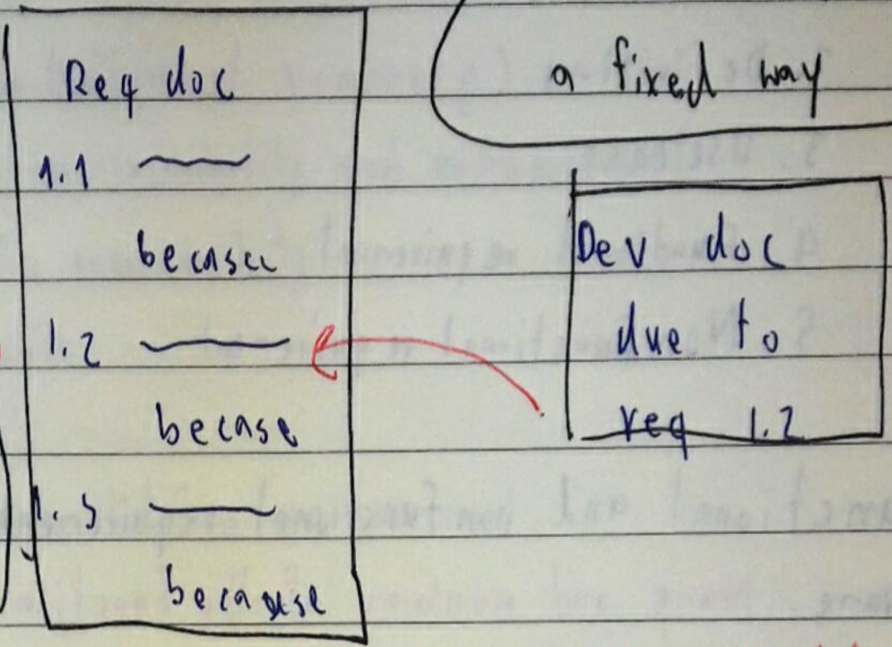
~~##~~ should not have a few req

Fixed requirement

should not have

a fixed way

rational



Prioritized

"customer sometime have trouble deciding which req they can live without"

Reviewing Requirements (each individual req doc should)

- outweigh the cost of dev
- ~~be~~ be important for the solution
- Clear and consitent notation
- un ambiguous
- logically consistent
- sufficient quality
- be real
- be verifiable
- uniquely identifiable
- Does not over-constrain the design of the system

Word to Avoid

comparatives

- faster, better, more etc

Imprecise adjectives

- user-friendly, efficient

Flexible

- vague commands

minimize, improve, optimize

The MoSCOW method

an acronym to help you remember a common system for prioritizing application feature

- Must
  - there are required features that must be include
  - They are necessary for the project
  - feature considered a success
- Should
  - there are important feature that should be include if possible
  - these feature may defferud next release
- Could
  - there are desirable feature that can be omitted
  - they can be pushed back in release next
  - not as important as the "should"
- Won't
  - there are completely optional features that the customers agree
  - They may be included in future

Managing Changing Requirement

Requirement change be cause
- Business process change
- Technology change
- problem become more understood

"Requirement Never STOP!!

need change control broad to make decision on the project