# Requirements

Tuesday, 4 December 2018 9:43 AM

## Requirements

- Specifies a need to be fulfilled by the product (within its scope)
- Product = brownfield/ greenfield?
- Stakeholders may not be aware of their needs/ how to communicate them
    - Steve Jobs - it's not the consumer's job to know what they want
- Functional requirement = what system should do
    - NFR = constraints of development and operation

> Data requirements e.g. size, volatility , persistency etc.,
> Environment requirements e.g. technical environment in which system would operate or need to be compatible with.
> Accessibility, Capacity, Compliance with regulations, Documentation, Disaster recovery, Efficiency, Extensibility, Fault tolerance, Interoperability, Maintainability, Privacy, Portability, Quality, Reliability, Response time, Robustness, Scalability, Security, Stability, Testability, and more ...

    - NFRs tend to be easier to miss, but may be critical
- Prioritise requirements categorically (e.g. Essential/ Typical/ Novel, High/ Medium/ Low)

### *Quality of Requirements*

- Unambiguous, Testable (verifiable), Clear (concise, terse, simple, precise), Correct, Understandable, Feasible (realistic, possible), Independent, Atomic, Necessary, Implementation-free (i.e. abstract)
- As a whole: Consistent, Non-redundant, Complete

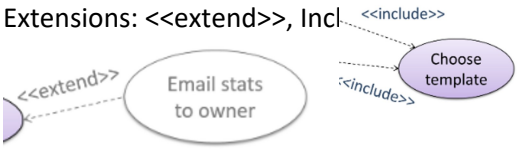
### Gathering Requirements

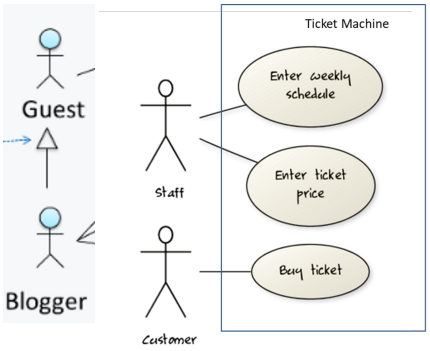
- Define target user/ problem scope
    - Avoid details/ bias from preconceived product ideas/ implementation plans
- Brainstorming: generate ideas, do not validate them (no bad ideas)
- User surveys: solicit responses and opinions from numerous stakeholders
- Observation/ usage data of existing system: users are in natural work environment
- Interviews: stakeholders and domain experts
- Focus groups: informal interview, interactive group setting -> more qualitative
- Prototype: mock-up/ scaled-down/ partial system
    - Proof-of-concept, get feedback, preview, early and controlled field testing
    - Uncover how users interact with system
    - Can also specify requirements
- Product surveys: study existing products (e.g. via technical documentation like product manuals)
- Scope-creep: things get added over time

### Specifying Requirements

- Prose: textual description, more useful for abstract components like product vision
- Feature list: features grouped by criteria (aspect, priority, order of delivery)
- User story: short and simple description of feature from user perspective (usually not detailed enough)
    - As a {role}, I want to {function} so that {benefit}
    - {benefit} can be omitted, but good to confirm if there is a concrete benefit
    - {role} can include characteristics (e.g. Expert/ forgetful)
    - Epics - bigger functionality
    - Optional: conditions of satisfaction/ priority/ estimated effort/ urgency
    - Convenient for scoping, estimation of effort, scheduling and delivery
        - Able to capture NFRs
        - Useful at early stages of gathering requirements
- Use case: *interaction* between actors (role) and system, based on specific functionality
    - Capture functional requirements involve user interaction with system

        - No UI design, UI prototype based on use case
    - System, Use Case ID, Actor
    - Main Success Scenario (MSS)/ Basic Course of Action/ Main Flow of Events
        - Assume nothing goes wrong, hence this scenario should be optimised
        - Steps specify the intention of the actor in externally visible behaviour
        - Should be self-contained (complete usage scenario)
        - Use case ends.
    - Extensions - exceptional/ alternative flow
        - Follow same number, e.g. 3a. Error detected
            - *a. At any time, user cancels transaction
        - Use case resumes from step 4.
    - Preconditions, Guarantees
    - Represented in use case diagrams
    - Extensions: <<extend>>, Incl <<include>> de>>




    - Actor generalisations, e.g. Blogger can do everything Guest can do
- Glossary: common understanding of noteworthy terms/ abbreviations/ acronyms
    - Use customer's terminologies
- Supplementary requirements
    - Mostly NFRs (e.g. Security - not what it does, but how it does, Performance, Budget, Timelines)

## Product Design

- Value to user, not you
- Minimise work for users (NUS Library example)
- Match user intent (e.g. Microsoft zooming, postpone 2 days vs. Edit and change dates)
- Sometimes less is more (even choices)
- Make users productive within minutes of contact - don't force them to read the manual (rtfm)
- Don't make users feel stupid (rely on common muscle memory, show help to correct)
- Implement benefits, not features (searching vs sorting)
- Be everything to Somebody