Requirement

What is requirement

Requirement is a singular documented physical and functional need that a product or process must be able to perform.

A good quality requirements should

- Establish a common understanding between the project sponsor, customers, developers, and other stakeholders.
- Improve customer confidence in the products to be delivered.
- Provide a roadmap to development.

https://www.ibm.com/developerworks/mydeveloperworks/blogs/requirementsmanagement/entry/good-requirements-writing?lang=en

Importance of (good)requirement

Customer requirement

1. Have one trunk
2. Have four legs
3. Should carry load both passenger & cargo
4. Black in color
5. Should be herbivorous

Our Solution

1. Have one trunk ☑
2. Have four legs ☑
3. Should carry load both passenger & cargo ☑
4. Black in color ☑
5. Should be herbivorous ☑

Our Value add:

Also gives milk ☺

http://lolyard.com/4771/customer-requirement

Type of requirement

(in terms of technical management)

- Customer Requirements
- Architectural Requirements
- Structural Requirements
- Behavioral Requirements
- Functional Requirement
- Non-functional Requirement
- Performance Requirements
- Design Requirements
- Allocated Requirements
- Derived Requirements

Type of requirement (1/5)

Customer Requirements

Statements of fact and assumptions that define the expectations of the system

- *Operational distribution or deployment*: ***Where will the system be used?***
- *Mission profile or scenario*: ***How will the system accomplish its mission objective?***
- *Performance and related parameters*: ***What are the critical system parameters to accomplish the mission?***
- *Utilization environments*: ***How are the various system components to be used?***
- *Effectiveness requirements*: ***How effective or efficient must the system be in performing its mission?***
- *Operational life cycle*: ***How long will the system be in use by the user?***
- *Environment*: ***What environments will the system be expected to operate in an effective manner?***

Systems Engineering Fundamentals **Defense Acquisition University Press, 2001**

Type of requirement (2/5)

- Architectural Requirements
  - *A requirement describes a condition or capability to which a system must conform; either derived directly from user needs, or stated in a contract, standard, specification, or other formally imposed document.*

    (Rational Unified Process® (RUP®)
- Structural Requirements
  - A requirement explain what does system done in view of structural.
- Behavioral Requirements
  - A requirement explain what does system done in view of behavioral.
  - Describes how system will reactive with user.

Type of requirement (3/5)

- Functional Requirement (“system must do *<requirements>*”)
  - Functional requirements drive the *application architecture* of a system
  - Define what a system is supposed to *do*
  - Cover the requirements needs

- Non-functional Requirement (“system shall be *<requirements>*”)
  - Non-functional requirements drive the *technical architecture* of a system.
  - Define how a system is supposed to *be*
  - Other operate function that besides the main purpose of the software

Type of requirement (4/5)

- Performance Requirements
  - How well software can execute measured by quantity, quality, coverage, timeliness or readiness
- Design Requirements
  - How to code, execute, test ,etc….
- Allocated Requirements
  - Dividing high-level requirements into multiple lower-level requirements

Type of requirement (5/5)

- Derived Requirements
  - This requirement does not explicitly stated by the customer, but has been derived from as a part from software analysis process

Example

User Requirement is ***"The system must work outdoors, 12 months a year in Alaska."***

Some of derived Requirements are:

1. The system must work in temperatures below 32 F / 0 C.
2. The system must work in the snow.

Another good example: http://dwaynephillips.net/advisors/dr.htm

Requirement Analysis

- Process of analyzing requirements to
  - Detect and resolve conflicts between requirements
  - Discover the bounds of the software and how it must interact with its environment
  - Elaborate system requirements to derive software requirements

- It’s very important to identify all stakeholder involved in the project
- Requirements analysis is critical to the success of a systems or software project
- Requirements analysis can be a long process which many delicate psychological skills are involved because during analyst, have to involve with many people with many characteristic.
- Analysts, sometimes has to use several techniques (by many experts) to identify the requirements from the customer, which can lead to the develop of scenarios or so called ***user stories***

Things to concerned
Stakeholder

Who is stakeholder ?

- Person or organization that may affected directly or indirectly
- Anyone who operates the system (**Ex:** normal and maintenance operators)
- Anyone who benefits from the system (**Ex:** political, financial and social beneficiaries)
- Anyone involved in purchasing or procuring the system. In a mass-market product organization, product management, marketing and sometimes sales act as surrogate consumers (mass-market customers) to guide development of the product
- Organizations which regulate aspects of the system (financial, safety, and other regulators)
- People or organizations opposed to the system (negative stakeholders)

Why important ?

- The developer have to make sure that all the software requirements satisfied all stakeholder. And the requirements can be obtain from stakeholder

Things to concerned
Stakeholder

Stakeholder Interview

Most common techniques and cheapest way used to acquire useful information about stakeholder's business, and their requirements. Not only just interviews in the begining, but can asks them anytime, in case they want to add some more requirements.

A good interview will get your project started in the right direction. Try to make the stakeholder speak freely, sometime it discover information that you had not anticipated or even stakeholder didn't think about it. Try to make the single documents that contain the summary of the requirements during interview and let the stakeholder take a look that we and them understand in the sameway

Things to concerned
Joint Requirements Development (JRD) Session

Also called Requirement Workshops

Many times that the stakeholder requirements can lead to *cross-functional implications* and they often miss to defined it during interview.

JRD session is the process that can elicit the cross-function implication, developer and stakeholder work together, analyze to uncover the cross-function implication

***cross-functional implications : the requirements that need many expert to work together and solved the requirements*

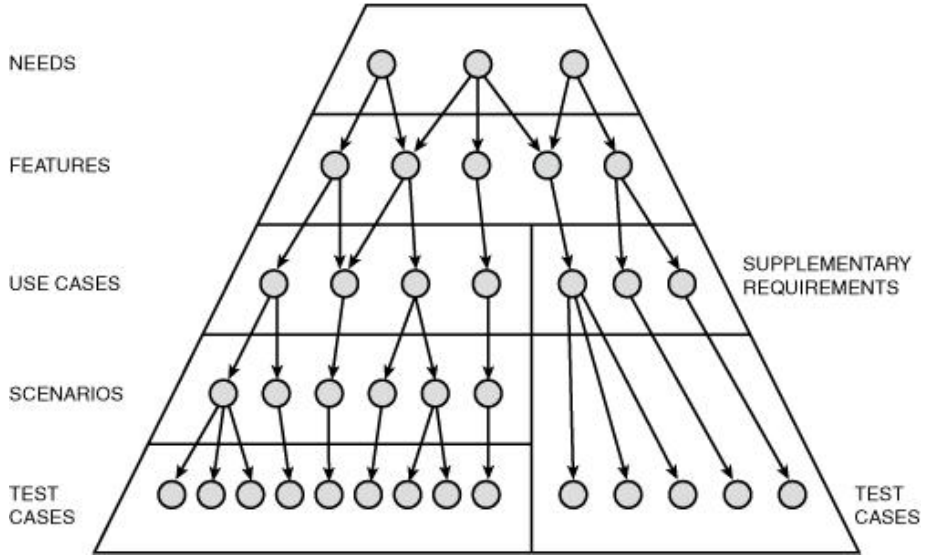
Things to concerned
Prototyping

- Help the future user and stakeholder to get the idea of what the system will look like
- Easier to make the design decision
- View the application early can lead to fewer changes later and reduced overall costs
- Customer can compare to ensure that software made matches the software specification
- May be completely different from the final product

The process of prototyping

1. Identify basic requirements - Determine basic requirements including the input and output information. Other details like security can be ignore
2. Develop Initial Prototype -The initial prototype includes only user interfaces.
3. Review - The customers examine the prototype and provide feedback for some additions or changes.
4. Revise and Enhance the Prototype - Using the feedback to improved the prototype. Negotiation about what is within the scope of the contract/product may be necessary.

Writing a systematic requirement



https://www.ibm.com/developerworks/community/blogs/requirementsmanagement/entry/good-requirements-writing?lang=en

Writing a good requirement

- **Simple, concise and easily understood**
  - Vague and ambiguous requirements
    - can be misinterpreted and result in a faulty design solution that fails to satisfy the customers' needs and your contractual obligations.
    - cannot form the basis for an objective verification criteria.
  - Each requirement must specify one and only one function.

- **Achievable**
  - If it's not achievable, you cannot satisfy your contractual obligations.
    - A quality supplier never signs up to something that they cannot deliver.
  - To be achievable, a requirement must be technically feasible, affordable, and fit within schedule and other constraints.

"The product shall be able to ~~survive~~ the temperature of the surface of the Sun

Survive => "withstand but unusable"?
=> "Badly perform after the exposure"?

"The product shall perform as specified after exposure to the non-operating temperature range of -20C to +200C

http://webcache.googleusercontent.com/search?q=cache:wHi1XsYtRpAJ:www.incose.org/chicagoland/docs/Writing%2520Effective%2520Requirements.pdf+&cd=1&hl=en&ct=clnk

Writing a good requirement

- **Unique**
  - Verifying redundant requirements adds unnecessary cost.
  - Redundant requirements may result in contradictions.

- The auto feeder shall be able to feed a baby panda with bamboo.
  - Some panda may prefer pizza.
- The auto feeder shall be able to feed up to 6 baby panda.

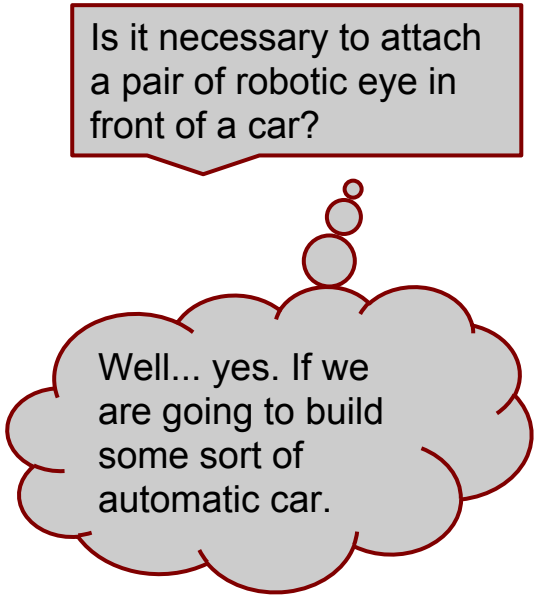
- **Traceable**
  - Relationship with parent requirement(s) is defined and documented.
  - Relationship with child requirement(s) is defined and documented.
  - Rationale describing allocation is documented.

- The auto feeder shall be able to feed up to 6 baby panda
  - As stated on the report SCP-162-B. The feeder shall able to feed baby panda with both bamboo and pizza.

http://webcache.googleusercontent.com/search?q=cache:wHi1XsYtRpAJ:www.incose.org/chicagoland/docs/Writing%2520Effective%2520Requirements.pdf+&cd=1&hl=en&ct=clnk

Writing a good requirement

- **Necessary**
  - A condition or capability needed to solve a problem or achieve an objective
    - Mission success cannot be achieved without meeting the requirement.
    - Avoid over-specifying (restricts design space, adds cost) – ask “Why is this a requirement?”, “What is the need it reflects?”
  - Specify the “what,” not the “how.”
- **Verifiable**
  - A requirement must be objectively verifiable (by test, analysis, inspection, or demonstration) to prove compliance; the requirement should be written in a fashion such that the means of verification is clearly understood.
  - A desired capability that cannot be objectively verified should be written as a design goal (“should”), not a requirement (“shall”).



http://webcache.googleusercontent.com/search?q=cache:wHi1XsYtRpAJ:www.incose.org/chicagoland/docs/Writing%2520Effective%2520Requirements.pdf+&cd=1&hl=en&ct=clnk

Common Errors

- **Bad or missing information**
  - Leads to over-specifying requirements or failing to specify needed requirements.
  - Remedies
    - Identify and involve the stakeholders and subject matter experts in every step of the product lifecycle.
    - Develop and validate concepts-of-operations (ConOps), mission scenarios, linkage and flow, constraints (cost, schedule, and technical).
    - Employ requirements analysis tools, checklists (comprehensive specification templates are good for this purpose).

Product shall be able to punch.

Product shall be able to punch the designated target.

http://webcache.googleusercontent.com/search?q=cache:wHi1XsYtRpAJ:www.incose.org/chicagoland/docs/Writing%2520Effective%2520Requirements.pdf+&cd=1&hl=en&ct=clnk

Common Errors

- **Specifying the “how,” not the “what.”**
  - Remedy
    - Determine the need statement and write the requirement accordingly.

- **Verbosity**
  - Extra words mean extra chances to misunderstand!
  - Remedies
    - Be concise.
    - Use simple, common terms whenever possible – avoid using “buzz words” and “project-speak.”

The device is installed with a “next-gen” wireless receptor.

The device is installed with a **<Insert a random technology standard here>** wireless receptor.

http://webcache.googleusercontent.com/search?q=cache:wHi1XsYtRpAJ:www.incose.org/chicagoland/docs/Writing%2520Effective%2520Requirements.pdf+&cd=1&hl=en&ct=clnk

Common Errors

- **Using a vague or ambiguous words, phrases, and statements.**
  - Subject to multiple interpretations.
  - Not objectively verifiable.
  - Creates an opportunity for the customer to require additional work without additional compensation, or the contractor to demand additional compensation for in-scope work.

  - Example words and phrases to avoid
    - “Achievable”, “adequate”, “approximately”, “complete”, “damaged”,“degraded”, “efficient”, “effective”, “minimize”, “maximize”, “flexible”,“modular”, “nominal”, “normally”, “optimum”, “survive”, “typically”,“usually”, “generally”, “often”, “easy”, “to the maximum (or minimum), extent”, “as much (or little) as possible”, “user-friendly”, “scalable”, “versatile”, “approximately”, “and/or”, “shall be designed to”, “shall be capable of”.

  - **Remedies**
    - **Be precise.**
    - **State the real need.**

http://webcache.googleusercontent.com/search?q=cache:wHi1XsYtRpAJ:www.incose.org/chicagoland/docs/Writing%2520Effective%2520Requirements.pdf+&cd=1&hl=en&ct=clnk

Common Errors

- **Multiple (“shall”) per statement.**
  - Increases risk that a requirement may be missed in the design.
  - May present problems in verification if the requirements have to be verified by different methods, at different times, or at different levels of assembly.
  - Remedy
    - **Only specify a single requirement per statement.**

- **Use of negative or passive sense**
  - i.e., “The system shall not perform the following maneuver when ….”
  - Remedy
    - **Reword to positive statements; use active verbs.**

- **Misuse of the terms “shall,” “will,” and “should.”**
  - **Requirements use “shall,” statements of fact use “will,” and goals use “should.”**
  - Terms such as “are,” “is,” “was,” and “must” don’t belong in a requirement.
  - Remedy
    - Stick with the government and industry standard defined above – to deviate from it will only invite confusion.

http://webcache.googleusercontent.com/search?q=cache:wHi1XsYtRpAJ:www.incose.org/chicagoland/docs/Writing%2520Effective%2520Requirements.pdf+&cd=1&hl=en&ct=clnk

Common Errors

- Inconsistent use of phrases to reference, specify alternative courses of action, or state limitations.
  - Can create confusion for the reader.
  - Remedy
    - Pick a phrase and be consistent throughout a specification or family of specifications.
      - For example, use “as specified in” when referencing external documents, use **“as specified herein”** or **“as specified in x.x.x”** when referencing within a document.
      - Use **“not greater than”** or **“not less than”** when stating limitations.
      - Use **“on Figure 1”** and **“in Table 2”** when referencing figure and table information.

http://webcache.googleusercontent.com/search?q=cache:wHi1XsYtRpAJ:www.incose.org/chicagoland/docs/Writing%2520Effective%2520Requirements.pdf+&cd=1&hl=en&ct=clnk

Use case / User Story

Use case

- Use case is a list of steps, typically defining interactions between a role (known in Unified Modeling Language (UML) as an "actor") and a system, to achieve a goal.

User story

- User story is one or more sentences in the everyday or business language of the end user or user of a system that captures what a user does or needs to do as part of his or her job function

http://en.wikipedia.org/wiki/Use_case http://en.wikipedia.org/wiki/User_story

Different between Use case/User Story

User Story	Use Case
• User story is about need. • Represent in simple short sentences. • Use everyday language.	• Use case is about the behavior you'll build into the software to meet those needs. • Set of actions performed by a system, which yields an observable result. • More complex.

ftp://ftp.software.ibm.com/software/rational/web/whitepapers/RAW14023-USEN-00.pdf
http://www.stellman-greene.com/2009/05/03/requirements-101-user-stories-vs-use-cases/

Characteristic of Good Use Case

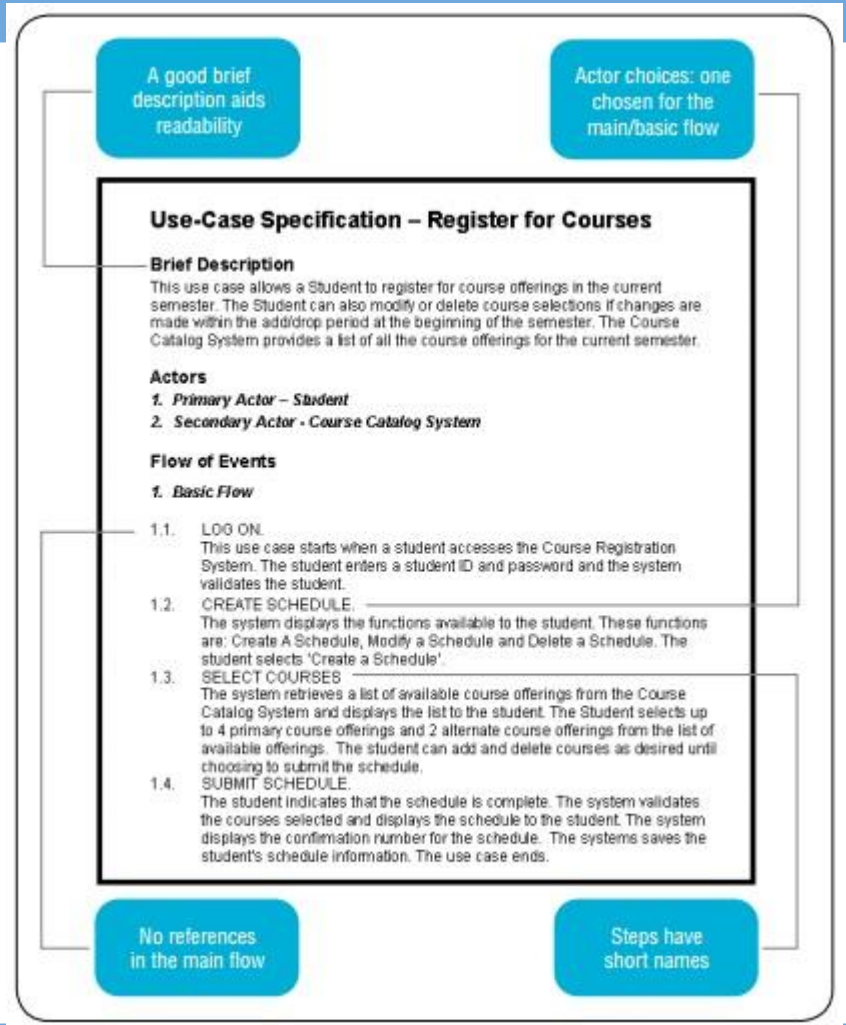
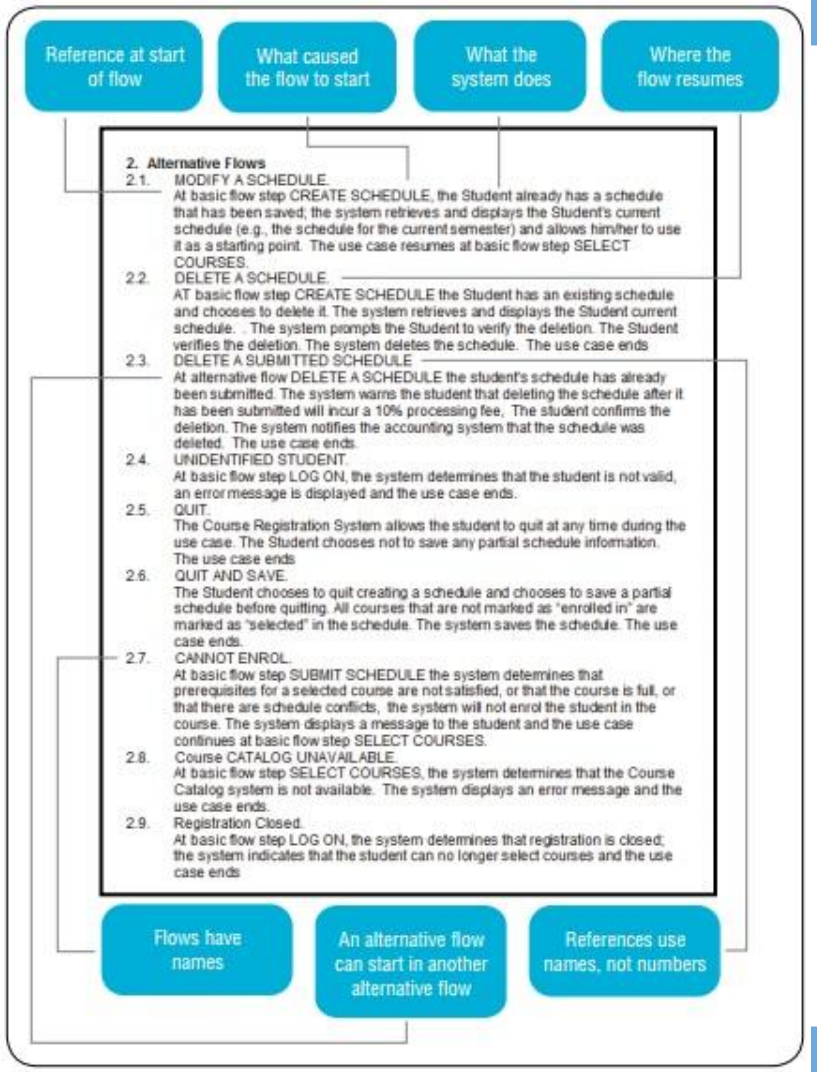


Figure 1: A well-written use case that tells how a student registers for courses.

ftp://ftp.software.ibm.com/software/rational/web/whitepapers/RAW14023-USEN-00.pdf

Characteristic of Good Use Case



ftp://ftp.software.ibm.com/software/rational/web/whitepapers/RAW14023-USEN-00.pdf

Good - Bad use case

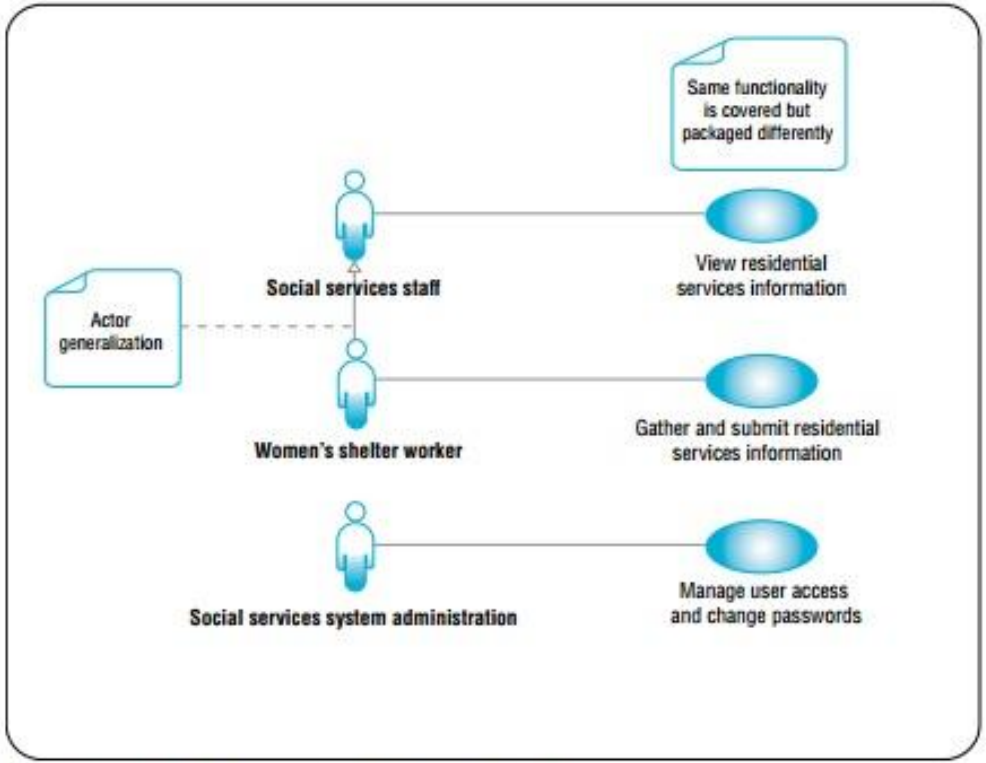


Figure 4: A well-written use case clearly indicates paths and value.

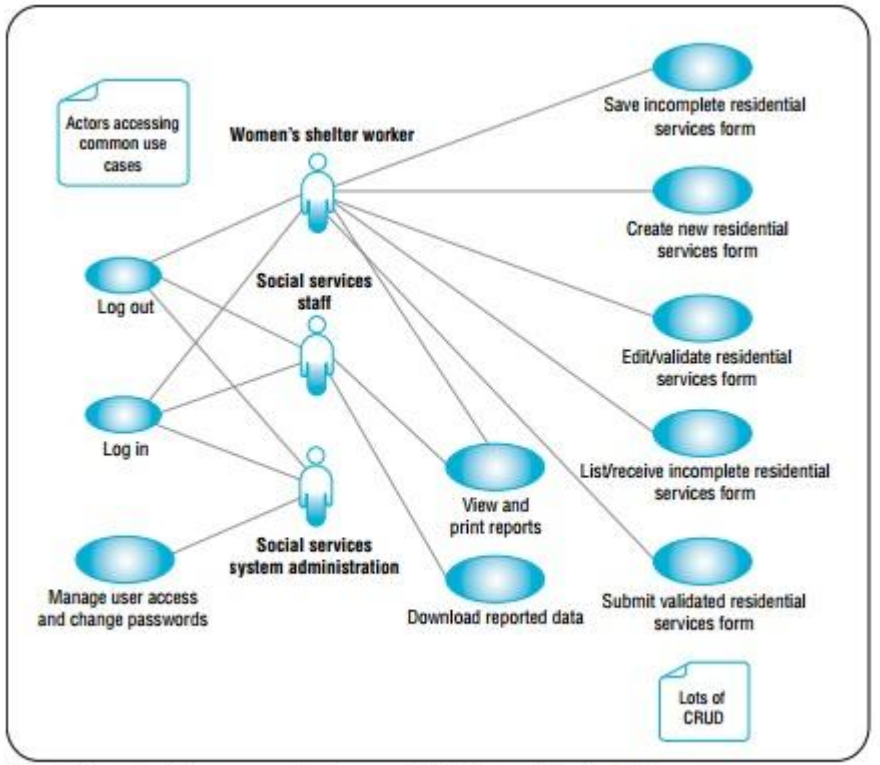


Figure 3: A poorly written use case has too many CRUD-type activities included.

ftp://ftp.software.ibm.com/software/rational/web/whitepapers/RAW14023-USEN-00.pdf

Create User Story

Simple template:

- As a {*role*} , I want {*goal*} so that {*reason*}
- In order to {*receive benefit*} as a {*role*}, I want {*goal/desire*}
- As {*who*} {*when*} {*where*}, I {*what*} because {*why*}

Example:

- As a user, I can backup my entire hard drive.

This can be derive into more specific stories.

- As a power user, I can specify files or folders to backup based on file size, date created and date modified.
- As a user, I can indicate folders not to backup so that my backup drive isn't filled up with things I don't need saved.

In Summary

- When writing effective requirements, remember the following basic concepts
  - **Make sure each requirement is necessary, verifiable, and achievable.**
  - **Write clearly, simply, concisely and unambiguously.**
  - **Make sure each requirement is unique and traceable.**
  - **Use only one “shall” per statement.**
  - **Specify “what’s required," not "how to do it".**
  - **Don't specify a design constraint unless it’s necessary to do so.**
  - **Avoid “buzz word” or project-speak.**
  - **Keep the language active and positive vs. passive and negative.**
  - **Be consistent with your choice of phrasing throughout.**
  - **Don't assume the reader will know what you meant even if that’s not what you wrote.**

http://webcache.googleusercontent.com/search?q=cache:wHi1XsYtRpAJ:www.incose.org/chicagoland/docs/Writing%2520Effective%2520Requirements.pdf+&cd=1&hl=en&ct=clnk

Self Study

- Writing an effective requirement
  - http://www.incose.org/chicagoland/docs/Writing%20Effective%20Requirements.pdf
- Good requirements writing
  - https://www.ibm.com/developerworks/mydeveloperworks/blogs/requirementsmanagement/entry/good-requirements-writing?lang=en
- Tips for writing good use cases.
  - ftp://ftp.software.ibm.com/software/rational/web/whitepapers/RAW14023-USEN-00.pdf

Thank you

References

- https://mike.cpe.ku.ac.th/~uwe/219343/KU-Bangkok_SWTest_2013-14_00_GeneralInfo_StudentPresentations_V1.0.pdf
- http://en.wikipedia.org/wiki/Requirements_analysis
- http://en.wikipedia.org/wiki/Use_case
- http://en.wikipedia.org/wiki/User_story
- https://www.ibm.com/developerworks/mydeveloperworks/blogs/requirementsmanagement/entry/good-requirements-writing?lang=en
- ftp://ftp.software.ibm.com/software/rational/web/whitepapers/RAW14023-USEN-00.pdf
- http://webcache.googleusercontent.com/search?q=cache:wHi1XsYtRpAJ:www.incose.org/chicagoland/docs/Writing%2520Effective%2520Requirements.pdf+&cd=1&hl=en&ct=clnk