

Requirements Engineering

Brooke Lampe

Requirements Engineering

Introduction

Stakeholders

Clients

- Party for whom you build software

Software Owners

- Party who owns the software

Users

- Party who will use the software

Government

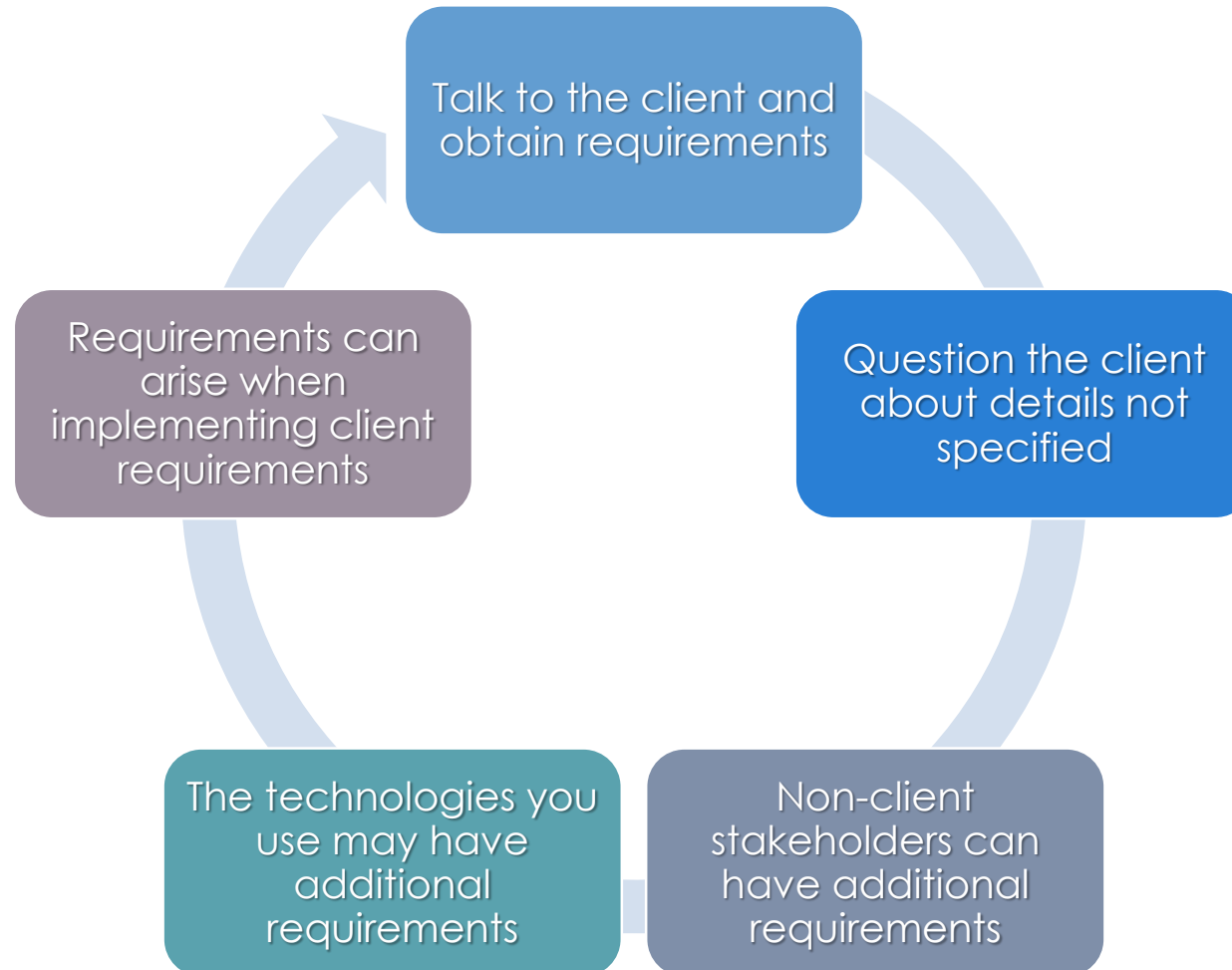
- Party who will regulate the software

Developer

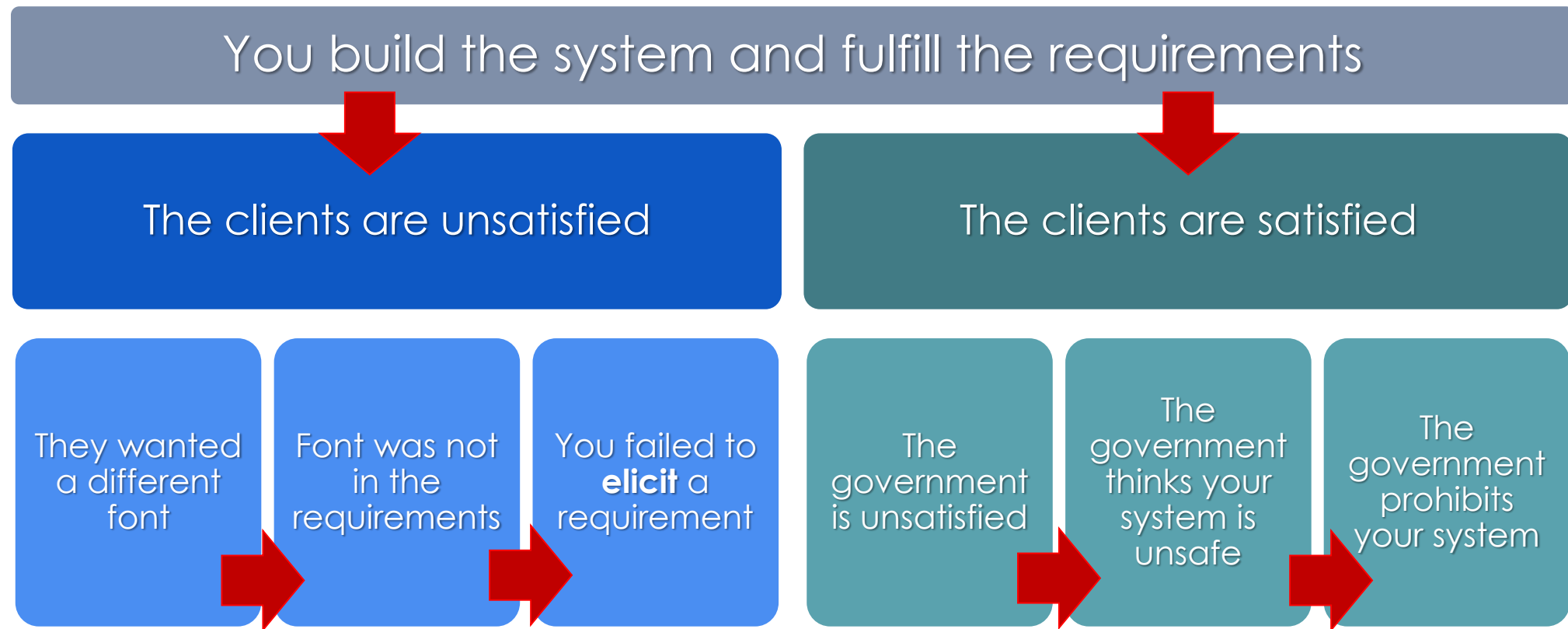
- Party who will write the software

Note: This is not an exhaustive list.

Requirements Elicitation



Requirements Elicitation



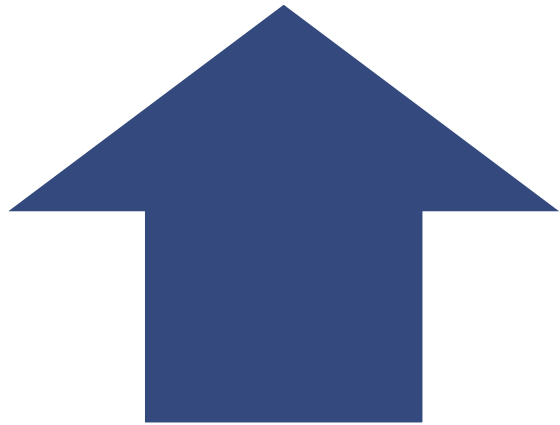
Requirements Negotiation



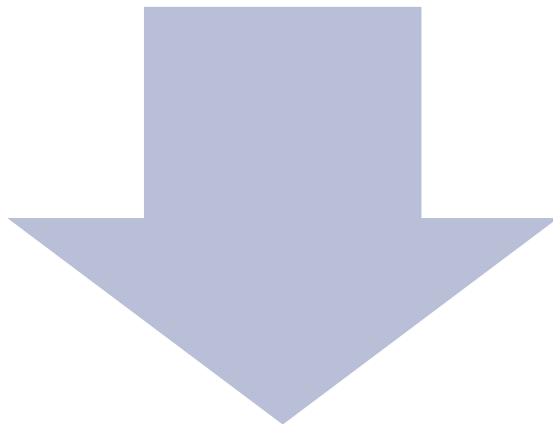
When requirements conflict, we decide which requirements are more important

Explain limitations in terms of time to build software versus features that can be implemented at that time

EXAMPLE: Requirements Negotiation



The users want no
advertisements



The shareholders want
lots of advertisements
(profit)

Requirements Classification

Functional

- **What it does**
- **The system should process transactions**

Product

- **Does the requirement affect the software?**

Traceable

- **Can we tie the requirement to a particular artifact or part of an artifact?**

Non-functional

- **How it does it**
- **The system should complete transactions within 24 hours**

Process

- **Does the requirement affect how we build the software?**

Emergent

- **Does the requirement relate to multiple artifacts?**

Requirements Classification

System

Is this a requirement for the system the software is part of?

Software

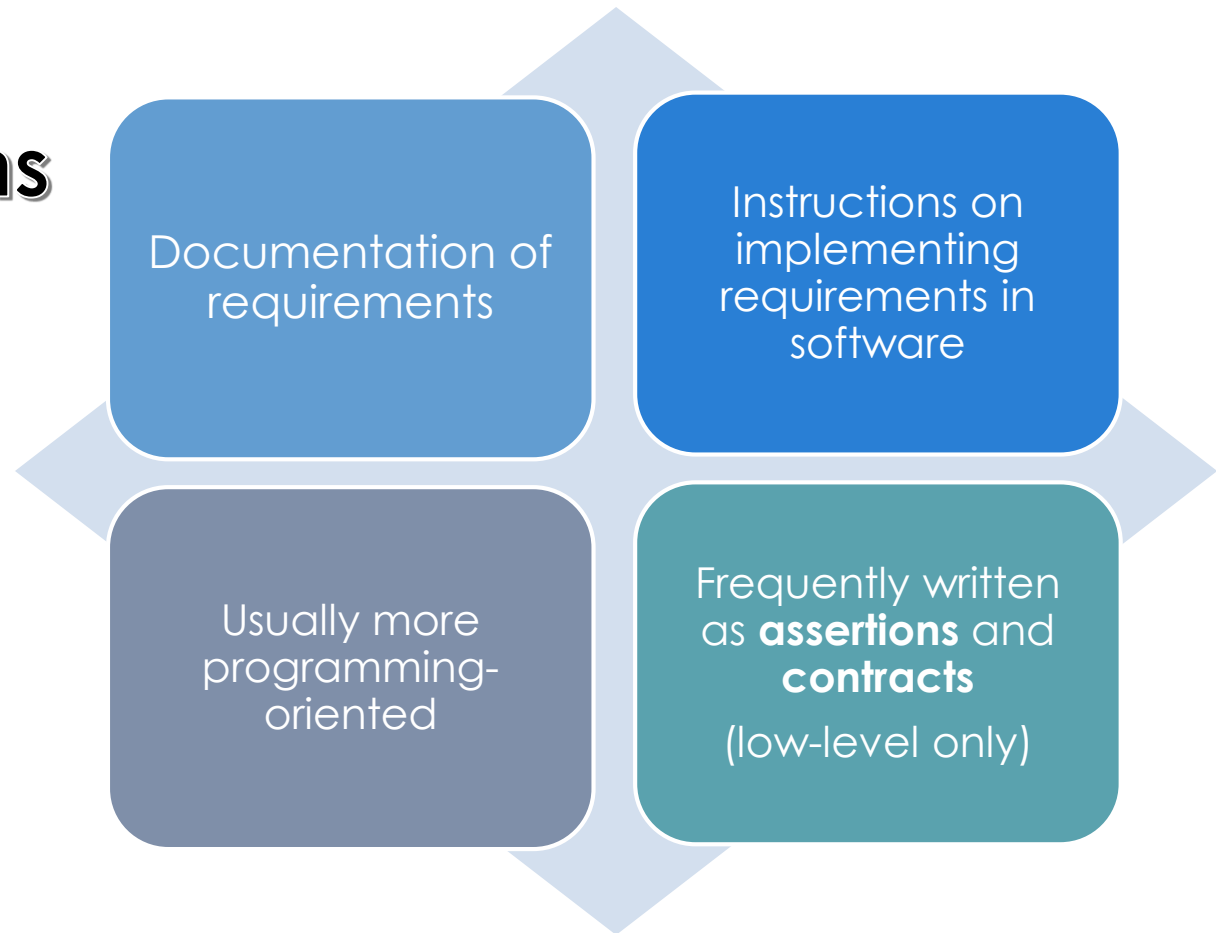
Is this a requirement for the software?

Component

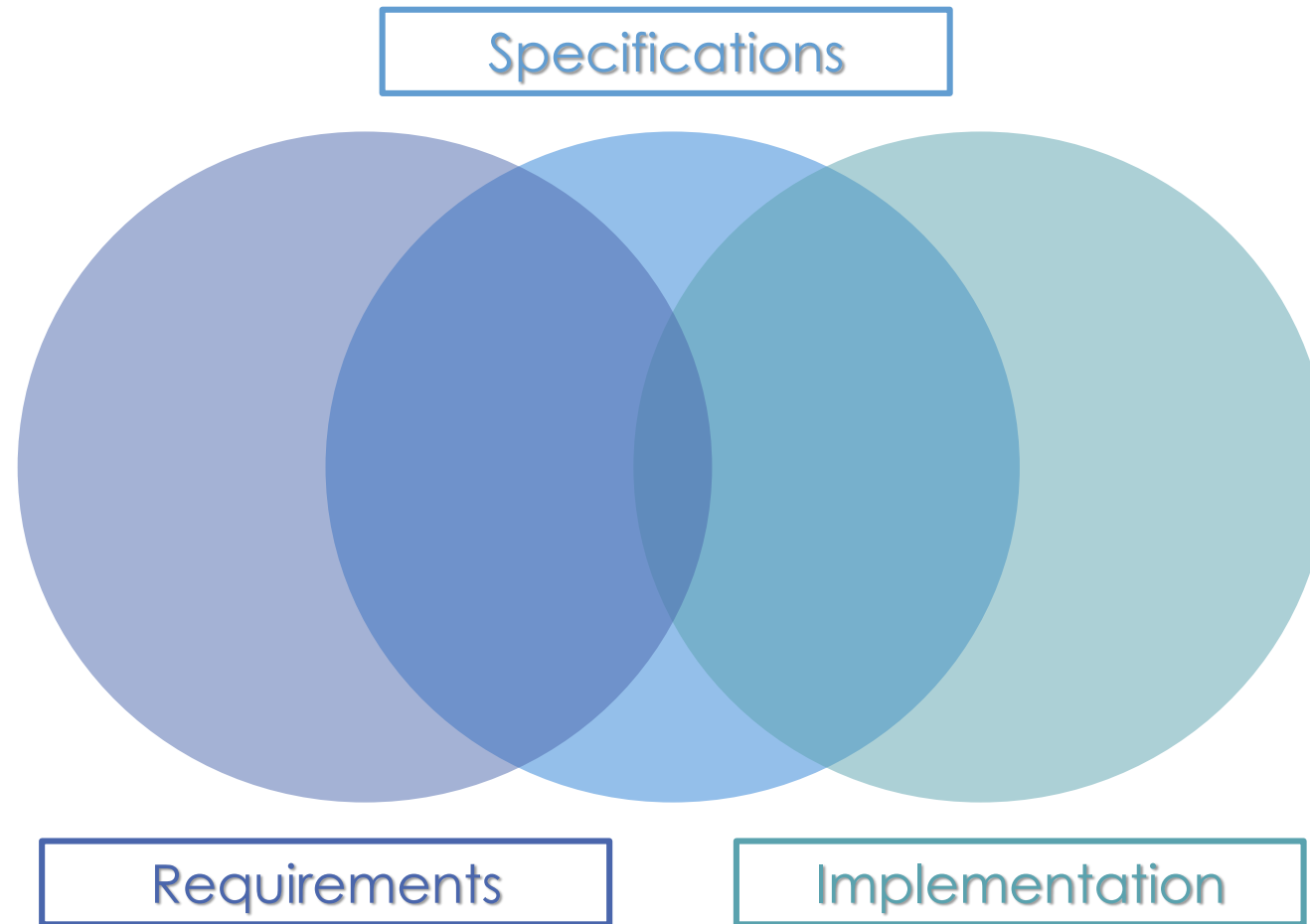
Is this a requirement for a specific part of the software?

Requirements Specifications

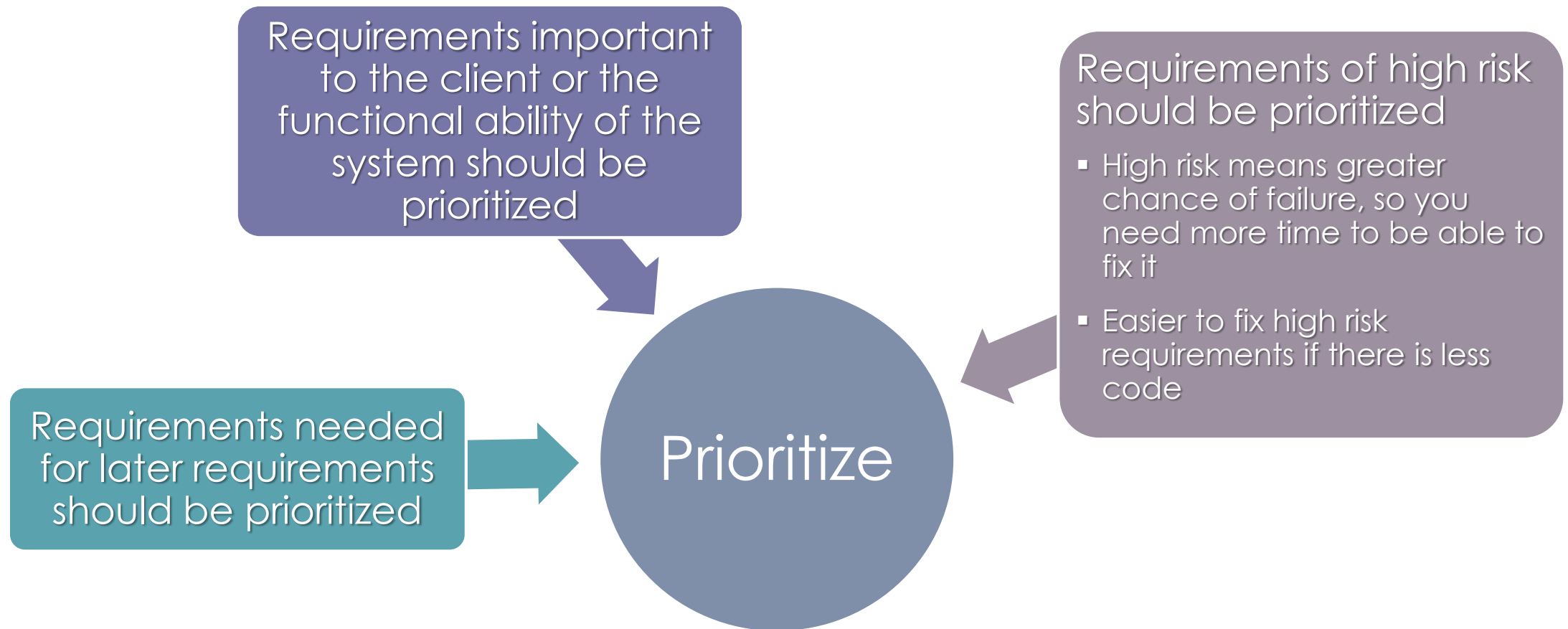
**Specifications
are...**




Requirements Specifications



Requirements Prioritization



Requirements Validation



**Confirming that you are
doing what the client
wants...**

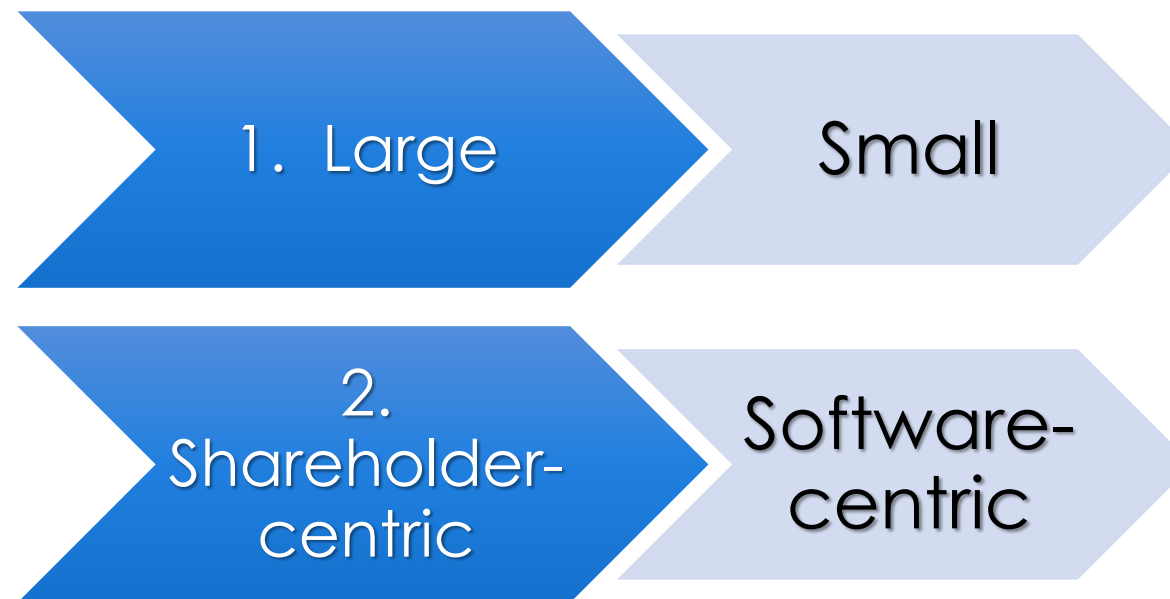
**...As you are
doing it**

Requirements Engineering

Requirements Derivation

Requirements Derivation

Requirements that are derived from the client's requirements



EXAMPLE 1: Requirements Derivation

1. The system shall track a user's time of inactivity.

What can we derive from this requirement?

EXAMPLE 1: Requirements Derivation

1. The system shall track a user's time of inactivity.
 - a. The system shall consider time without activity as time in which no elements of the system are interacted with by the user.
 - i. The system shall be able to track time in minutes.
 - ii. The system shall reset the tracked time when the user interacts with an element of the system.

EXAMPLE 2: Requirements Derivation

1. On a successful login, the system shall redirect from the Login Screen to the Accounts Screen.
 - a. The system shall be able to redirect.
 - i. The system shall be able to automatically follow a link.

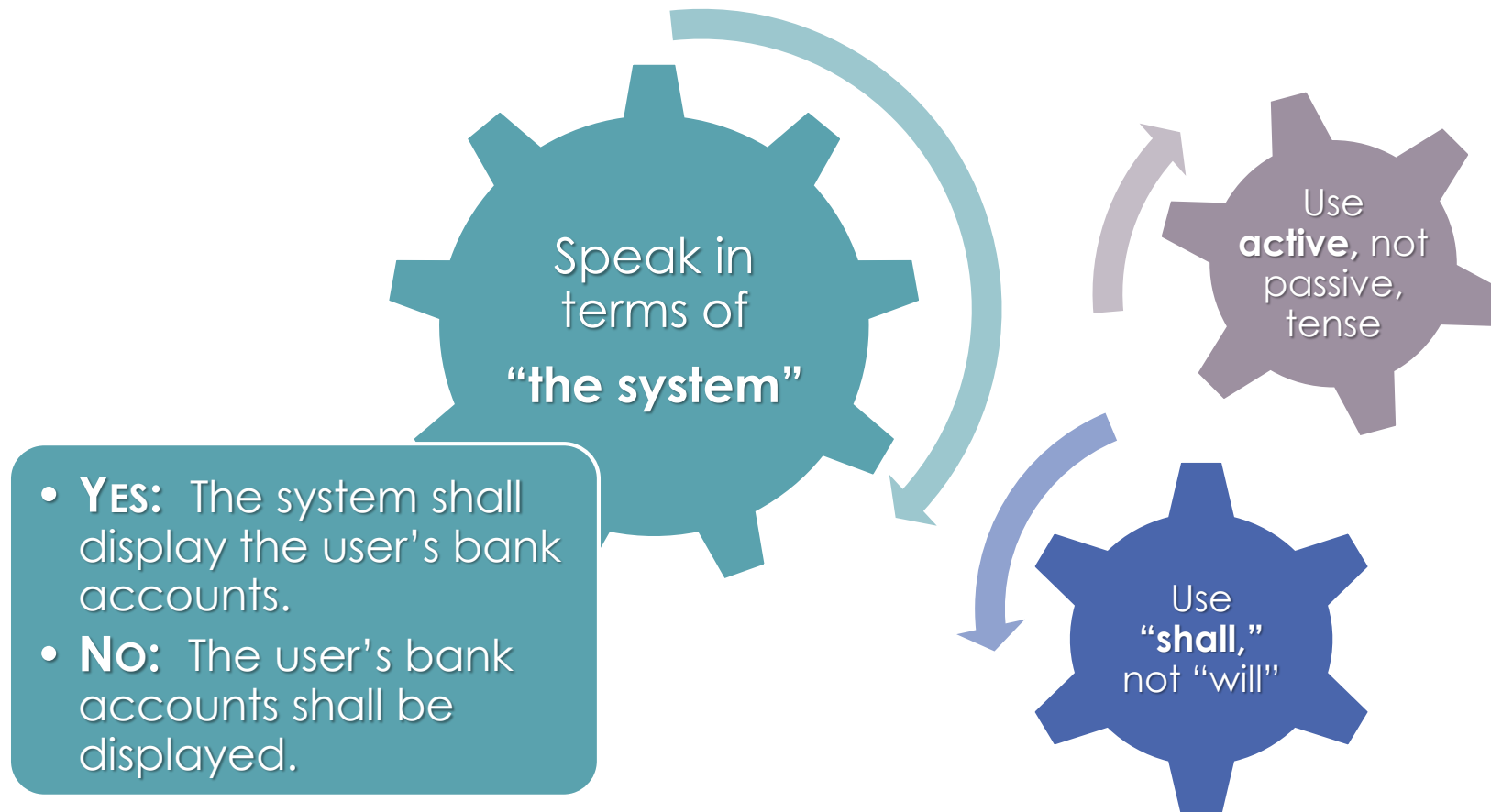
What is wrong about this derivation?

EXAMPLE 2: Requirements Derivation

1. On a successful login, the system shall redirect from the Login Screen to the Accounts Screen.
 - a. The system shall be able to redirect.
 - i. The system shall be able to automatically follow a link.

This derivation is excessive.

Requirements Derivation TIPS



Requirements Engineering

Requirements Analysis

Importance of Requirements Analysis

Elicitation

If questions are not asked, some requirements may not be discovered

Negotiation

Two clients who have completely different views will create inconsistent requirements

Failed negotiation or forgetting negotiation is a huge problem

Derivation

Duplication because something is derived from more than one requirement

Insufficient derivation and incomplete requirements

Incorrect derivation

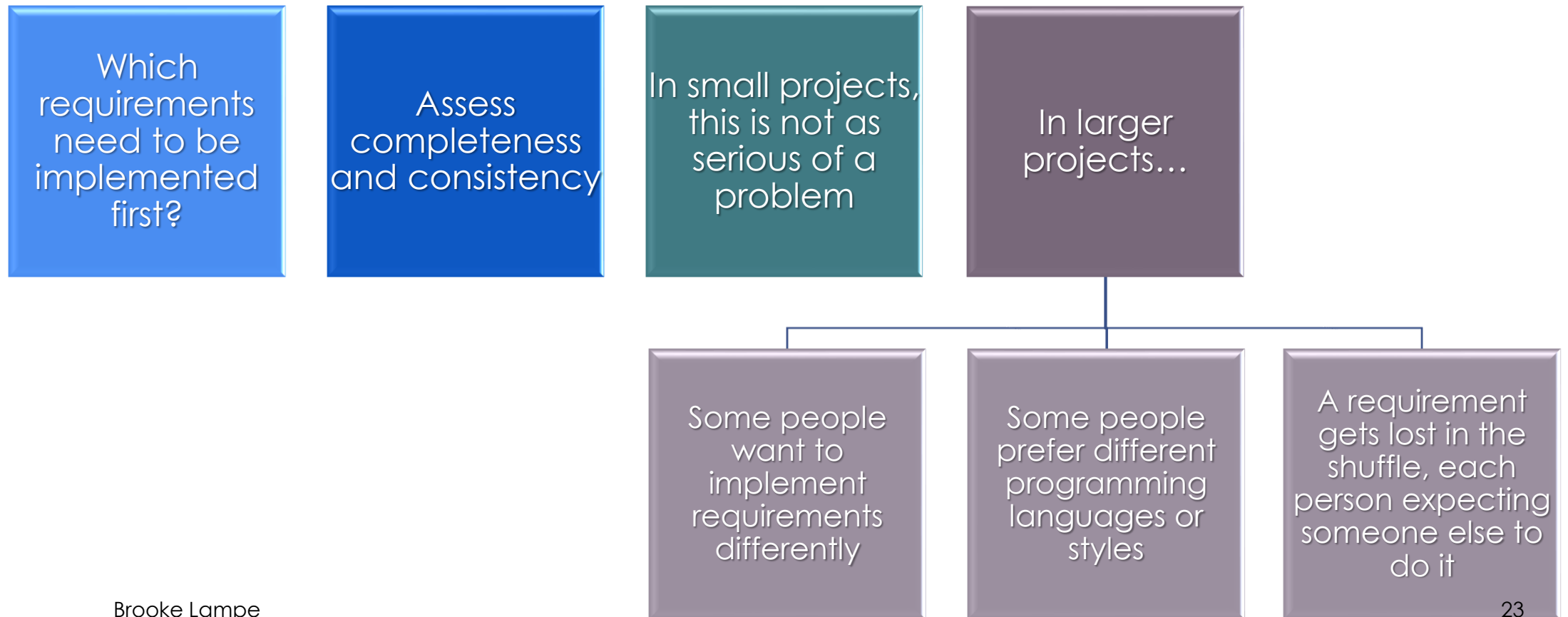
- Derived requirements do not match the top-level requirement they came from

Specification

Errors in requirements

- Forgot to write them down (*lapse*)
- Written down incorrectly (*slip*)

Requirements Analysis



Completeness & Consistency



Completeness

- All upper-level requirements identified
- All parts of an upper level requirement are supported by derived requirements



Consistency

- Requirements cannot be in conflict with each other
- Requirements can all be fulfilled simultaneously

EXAMPLE 3: Requirements **In**completeness

- The following requirements belong to the specification of a banking system:
 1. The system shall require approval for transfers of money in excess of a thousand dollars to the account of a different user.
 2. The system shall not require approval for transfers of money less than a thousand dollars.

Try to find an incompleteness in the requirements.

EXAMPLE 3: Requirements **In**completeness

- The following requirements belong to the specification of a banking system:
 1. The system shall require approval for transfers of money **in excess** of a thousand dollars to the account of a different user.
 2. The system shall not require approval for transfers of money **less than** a thousand dollars.

What if the amount transferred is exactly \$1000?

EXAMPLE 4: Requirements **In**consistency

- The following requirements belong to the specification of a banking system:
 1. All transfers of money shall be approved by a controller
 - a. Transfers of money in excess of a thousand dollars shall be approved by a controller
 - b. Transfers of money less than a thousand dollars shall not be approved by a controller

Try to find an inconsistency in the requirements.

EXAMPLE 4: Requirements **In**consistency

- The following requirements belong to the specification of a banking system:
 1. All transfers of money **shall be approved** by a controller
 - a. Transfers of money in excess of a thousand dollars shall be approved by a controller
 - b. Transfers of money less than or equal to a thousand dollars **shall not be approved** by a controller

Requirement (1.) conflicts with requirement (b.).

EXAMPLE 4: Requirements **In**consistency

- The following requirements belong to the specification of a banking system:
 1. All transfers of money **shall be approved** by a controller
 - a. Transfers of money in excess of a thousand dollars shall be approved by a controller
 - b. Transfers of money less than or equal to a thousand dollars **shall not be approved** by a controller

The upper level requirement says that all transfers of money need to be approved by a controller, while the supporting requirement says that transfers of money less than \$1000 do not need to be approved.

EXAMPLE 5: Requirements **In**completeness

1. The system shall allow a locked account to be unlocked by a system administrator at a user's request.
2. The system shall only allow five invalid login attempts per username per day.
 - a. The system shall record the username(s) of the invalid login attempts.
 - b. The system shall lock the account of a username with five invalid login attempts.
3. The system shall only allow invalid five login attempts per IP address per day.
 - a. The system shall record the IP address(es) of the invalid login attempts.
 - b. The system shall lock the IP address of a computer with five invalid login attempts.

Try to find an incompleteness in the requirements.

EXAMPLE 5: Requirements **In**completeness

1. The system shall allow a **locked account** to be unlocked by a system administrator at a user's request.
2. The system shall only allow five invalid login attempts per username per day.
 - a. The system shall record the username(s) of the invalid login attempts.
 - b. The system shall **lock the account** of a username with five invalid login attempts.
3. The system shall only allow invalid five login attempts per IP address per day.
 - a. The system shall record the IP address(es) of the invalid login attempts.
 - b. The system shall **lock the IP address** of a computer with five invalid login attempts.

What is a locked account? What is a locked IP address?
What does it mean to be locked?

EXAMPLE 6: Requirements **In**consistency

1. The system shall prompt the user to enter a username and a password.
2. When a valid username is entered, the system shall show the security image associated with that image.
3. The system shall allow the user to log in when the user enters a valid username and the correct password for the specified username.
4. In a failed attempt, the system shall indicate that the credentials entered were incorrect.
5. In a failed attempt, the system shall not indicate which of the credentials, the username or the password, was incorrect.

Try to find an inconsistency in the requirements.

EXAMPLE 6: Requirements **In**consistency

1. The system shall prompt the user to enter a username and a password.
2. When a valid username is entered, the system shall show the security image associated with that image.
3. The system shall allow the user to log in when the user enters a valid username and the correct password for the specified username.
4. In a failed attempt, the system shall indicate that the credentials entered were incorrect.
5. In a failed attempt, the system shall not indicate which of the credentials, the username or the password, was incorrect.

Requirement (2.) conflicts with requirement (5.).

EXAMPLE 6: Requirements **In**consistency

1. The system shall prompt the user to enter a username and a password.
2. When a valid username is entered, the system shall show the security image associated with that image.
3. The system shall allow the user to log in when the user enters a valid username and the correct password for the specified username.
4. In a failed attempt, the system shall indicate that the credentials entered were incorrect.
5. In a failed attempt, the system shall not indicate which of the credentials, the username or the password, was incorrect.

Requirement (2.) says the system shall show a valid username's security image. If the system showed the security image, the user knows the username was correct, which means, in a failed attempt, that the password was wrong. This deduction violates requirement (5.).

Analysis Techniques

Ad hoc

Get a team together, read through the documentation, and see if something jumps out

Checklist-based

Write lists of problems to look for in the requirements (checklists)

Go through the requirements, go through the checklists, and compare

Scenario-based

Design scenarios (similar to test cases)

- Walk through the behavior of the system in that scenario, following the requirements

Is there ever a time in which we have to do two things at the same time?

According to research, this is the best low-cost option

Scenario-based Requirements Analysis

Prioritized

Pick the most important scenarios and do them first

Antagonistic

Process:

- Pick a scenario
- Check the requirements
- Figure out which requirements are strong and weak
- Design a scenario to hit the weak requirements

Over time, the weak points are isolated

Coverage-driven

Try to hit all of the requirements with scenarios

Check off each requirement when hit

Similar to testing

Proof-based

Process:

- Take all of the requirements
- Write them mathematically
- Use a computer to apply a reasoning algorithm to look for inconsistencies

If you have the time, this is the best high-cost option

Safety-critical systems (airlines) do this