Software Requirements

IN AN HOUR

Password: institute

Today’s Example

Collecting knee gait data to an iPhone

- A project example from last year kindly provided by Dr. Bercich

I’ve expanded it to be a little bigger to provide examples for this workshop

Knowing what to implement

Software Requirements are our ultimate goal, but it’s difficult to start writing requirements at that level

Before we can get to full Software Requirements, we must:

- Determine our stakeholders
- Elicit the basic problem the software needs to solve/fix
  - The “Needs”
- Compile high level ideas of what the software must do
  - The “Features”
- Then we can complete Software Requirements
  - “Use Cases” and “Shall Statement”

Why, What, and How

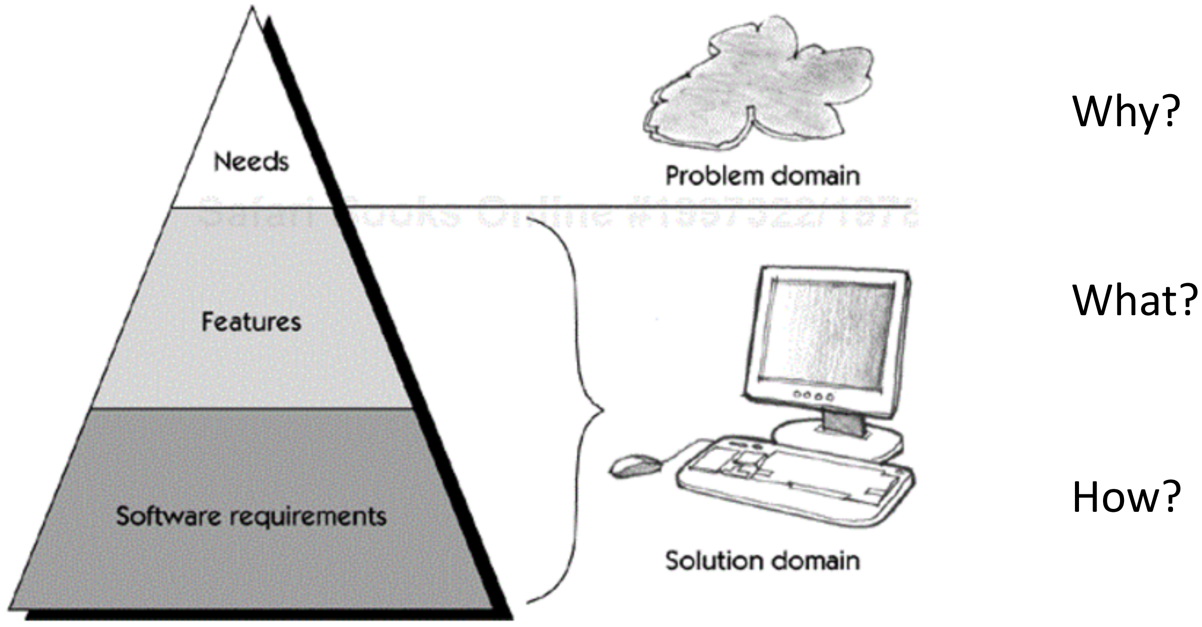


Figure 2-1. Overview of the problem domain and the solution domain

Why?

What?

How?

Image from “Managing Software Requirements, A Use Case Approach”, Second Edition
Leffingwell, et al, page 21

Needs

Why the project is needed.

Pain points for the stakeholders

Regulatory needs

Does not contain the word “need”

Knee Gait Example:

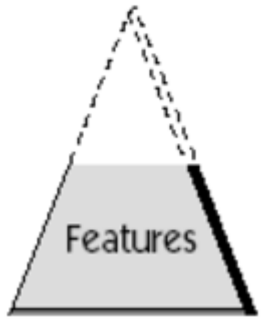
- N1: It is difficult for our physical therapists to fully understand a user’s knee gait
- N2: Without knee gait information, we cannot assign the proper treatment plan to our patients
- N3: Often little space for a large computer in a hospital or physical therapy room.
- N4: Must be HIPAA Compliant

Features

What the software will do when the project is complete.

Think of them like advertising statements

- What would you tell someone you have only 5 seconds with to interest them in your project?



Examples:

- F1: Knee gait parameters are gathered by an iPhone.
- F2: Knee gait parameters are displayed on an iPhone.

- Many more about the device, hardware, etc., but we’re going to focus on the software.

Software Requirements

How the software will behave or be coded.

Two basic types:

- Use Cases
- Shall Statements

Use Cases:

- Conversation between User and System

Shall Statements

- Useful for
  - Non-Functional Requirements (later)
  - Technical, Cross-Cutting Concerns
  - Regulated Environments

Examples:

- Later

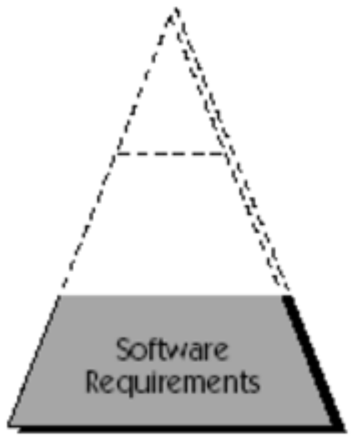


Image from “Managing Software Requirements, A Use Case Approach, Second Edition”
Leffingwell, et al, page 21

Traceability: the key realization

We say we have <u>traceability</u> when we can **prove**:

- Every need has been implemented by one or more features.
- Every feature satisfies one or more needs.

- Every feature has been covered by one or more software requirements.
- Every software requirement covers one or more feature.

Without traceability, we can’t show when we’re done

Example:

- N2: Without knee gait information, we cannot assign the proper treatment plan to our patients
- Is satisfied by F2: Knee gait parameters are displayed on an iPhone.

Use Cases

IN DETAIL

Use Case Template

A. Name
B. Brief description
C. Actors
D. Basic flow
E. Alternate flows
F. Pre-conditions
G. Post-conditions
H. Other stakeholders
I. System/sub-system
J. Special requirements

View Results Use Case 1/4

A. Name: View Knee Gait Results

B. Brief description: User views the results of the knee gait test.

C. Actors: Physical Therapist

View Results Use Case 2/4

D. **Basic flow:**

1. User selects to view knee gait results
2. System displays available patients with results
3. User selects desired patient
4. System displays times of recent result entries
5. User selects time of desired results entry to view
6. System displays the following knee gait data from the selected time:
    1. Accelerometer’s X value
    2. Accelerometer’s Y value
    3. Accelerometer’s Z value
    4. Gyroscope’s R value

View Results Use Case 3/4

E. **Alternate flows**

3. User exits view results process
   a) System returns user to the main menu

5. User exits view results process
   a) System returns user to the main menu

F. **Pre-conditions**

- User is on the main menu screen

G. **Post-conditions**

- User is viewing knee gait results
  - OR
- User is on the main menu screen

View Results Use Case 4/4

H. **Other Stakeholders**
   - Patient

I. **System / Sub System**
   - N/A

J. **Special requirements**
   - Defaults to showing for the most recent 15 results for a given patient

Use Cases – Why?

What's the point of all of this?

- To determine the information passed between a user and the system
- To provide a vehicle for discussing these user/system interactions with client

What we do not do?

- Define how that interaction will happen
  - Avoid words like "clicks" or "button"
  - Instead: "User selects desired patient"
- Use words like "information" or "details"
  - These are not specific enough, e.g., *Displays accelerometer data* can be made more specific
  - Instead: "System displays the following knee gait data from the selected time:

    Accelerometer's X value

    Accelerometer's Y value

    Accelerometer's Z value"

Prototypes

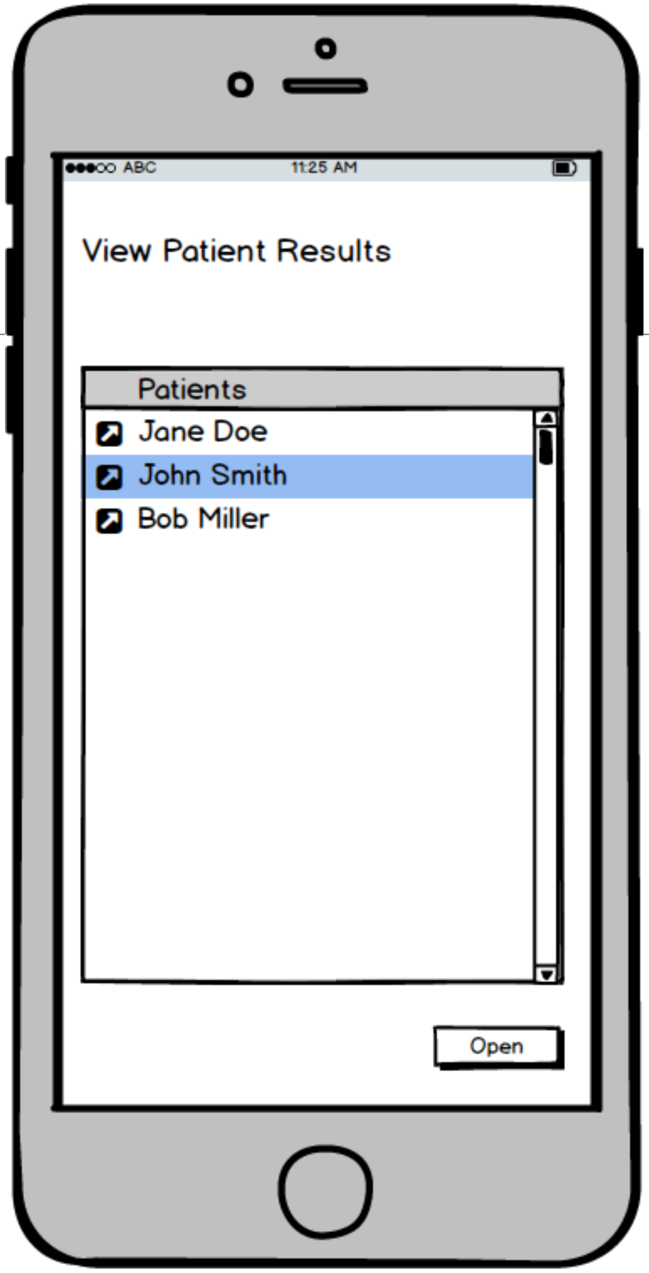
Once you have an idea, prototype it!

- On paper is quick
- With layout software is requires more work
- In implementation software often requires lots of work

Do not make it fancy. Why?

- The fancier it is, the more the client thinks it’s done!
- The client is less likely to suggest changes
- If you make the prototype look fancy, it will take a significant amount of time. If numerous changes are made by client (or you) then that time was not well spent

Prototype Example



Non-Functional Requirements

The way in which the software accomplishes a task

Covers requirements that aren’t about what it does, but more about “how” it does it.

Quality attributes to consider:

- Usability
- Security
- Modifiability
- Scalability
- Performance

Technical or Cross-cutting concerns

- HIPAA compliance or Security
- Communication framework requirements between gyroscope and iPhone

Shall Statements

Non Functional Requirements can’t be written as use cases

- Exist across many use cases

We use Shall Statements:

- Text-based representation of a requirement
- Contains the word “shall” to indicate what the software must do.
- IEEE standards exist, but use is variable among clients

Shall Statements

For example, a performance requirement:

- Usually stated as “Charts must display quickly” by the client.
- Shows up in features as “Patient knee gait data displayed quickly”

DO NOT PUT ADVERBS IN YOUR SOFTWARE REQUIREMENTS

The following definition is from Merriam-Webster

adverb - a word belonging to one of the major form classes in any of numerous languages, typically serving as a modifier of a verb, an adjective, another adverb, a preposition, a phrase, a clause, or a sentence, expressing some relation of manner or quality, place, time, degree, number, cause, opposition, affirmation, or denial, and in English also serving to connect and to express comment on clause content

When the following conditions are true:

- The software is under normal operating conditions
- The user has selected to display a knee gait chart for a selected data point

the software shall display the requested chart within 3 seconds of the user’s selection.

Conclusion

Concrete Concepts:

- Needs
- Features
- Software Requirements
  - Use Cases
  - Shall Statements
- Prototypes
- Non-Functional Requirements

Overall Concept:

- You have to talk to your client, a lot!
  - A lot more than you think you need to!
    - Seriously, even more than you think you need to now!

Questions?

Activity

For your current project:

What needs are you trying to solve?

- Remember, these are current problems, not things the solution “needs” to do.
- Stick to software for your project (if possible) for this activity

What features will you implement to solve these?

- These should be quick advertising statements to sell someone on your software

Check with me with questions/concerns/etc.