Working with the Health First Requirements Document

Overview

It is a good idea to read through the Health First Requirements Document to 1) understand how the Requirements Document is written (so that you can edit it consistently) and 2) to understand more about how the proposed system functions.

The different parts of the document that you can modify include:

1) The List of Requirements: This is a table of features. When you add features, you should extend this table to include the new feature(s). Notice the two additional columns: Type and Priority. For Type, you can add Non-Functional or Functional Requirements. Priorities can include: Required, Highly Desired, Desired.
2) Use Case Overview: This provides a brief introductory description showing how the Use Cases (or database features) fit in together via a diagram and text. For any new features that are added, they should be introduced minimally within the text.

   The Use Case Diagram is a single diagram listing the features of the database system. Each use case is shown as a bubble, and named in <verb noun> or <verb adjective noun> form, such as <Make Appointment> or <Update Patient Information>. Larger teams may modify the use case diagram to include new use cases. Lines connect the use case bubbles to the primary actors who initiate the transactions, and to secondary actors, who participate in the transaction. Secondary actors may be persons who receive email notices, or be systems which interface during the transaction. For example, the health plan may be sent requests or notices during a medical transaction. *You don't need to update this picture, but do understand its role.*
3) Use Case Descriptions: Each Use Case included in the Use Case Diagram should have an extended description explaining how the feature works. This includes a Use Case Description as well as any forms to show usage. The next section describes this more completely.

Working with a Use Case Description

A Use Case table defines who interacts with the use case and how the use case proceeds. The description below defines each table row:

Use Case: Name of Use Case as <Verb Noun> or <Verb Adjective Noun>, such as: Create Patient Information
Use Case ID: Sequentially allocated number
Primary Actors: These are the roles who initiate the transaction. They may include: Nurse, Medical Administrator, Doctor, Registered Dietician
Secondary Actors: These are the roles or systems who interact with the system during the transaction, but do not

initiate it. For example, if the system initiates emails to patients or staff, then these people would be listed here. Alternatively, the transaction may contact the Health Plan system to request or provide information, such as a bill or authorization.

Preconditions: This indicates the conditions at the beginning of the transaction. Ordinarily you can assume that the user is logged in. Commonly used: **"**The user is at the main menu." Do not make assumptions about the application, such as that a patient exists, since your application should handle normal and abnormal conditions.

Flow of Events:

1. The user case begins when the user selects <some form>.
2. The user (or The system) does … (Always starting the sentence with "The user" or "The system" makes sure that it is clear who does what.
3. While the system finds a matching record (This is an example of a sort of loop. Nested entries will have additional numbering: 3.1, 3.2, 3.3, etc., such as:)
    3.1 The system displays an error message: "<whatever>", and requests the user revise the information.

Note: Continue with numbered "The user …" and "The system …" until the entire transaction within the database system is described. You may also nest simple "if" statements, similar to the while statement above. All statements should be sufficiently non-technical that a business user can understand it. The Requirements level describes how the product works from a users' perspective.

Alternate Flows:

You may list error type conditions here, or if they are short, include them in the Flow of Events, and list 'none' here. If there are error conditions, Alternate Flow sections are added as shown below.

Postconditions: (This describes what is true after the use case is completed. For example:)

1. The new record has been saved into the database.
2. A log records the transaction, including who performed what transaction.

Each Alternate Flow you include should be documented with additional tables:

Alternate Flow: Determine Health Plan Eligibility: Plan Eligibility Information Not Available

(Note that the format is:
Alternate Flow: <Name of Use Case>: <Name of Alternate Flow>
Where the Alternate Flow is the same name listed in the Use Case's Alternate Flow section.)

1. The secondary scenario begins after step <n> of the Primary Scenario
2. The system … (e.g., displays an error message: "No connection" or "Connection Timed Out" or "Host not available" etc.)

Postconditions:
(This describes what is true after the alternate flow is completed. E.g.:)
A descriptive error message is displayed.

Also include a form or report with each Use Case, showing the fields on the form, preferably with sample data. This may be drawn with Microsoft Word or cut and pasted from any tool you prefer (as long as it can be read by your instructor.)

How NOT to write a Use Case Description:

Compare these two techniques:

Case A: Correct:

The use case begins when the user selects form x
The system displays form x, including patient notes from visit
The user can select the button 'total bill'
The system calculates and displays the total cost
The user enters the amount paid.

Case B: Incorrect:

The use case begins when the patient approaches the desk to pay a bill
The nurse tells the patient the total for the visit
The patient pays the bill.
The nurse says "Thank you for your visit".

The first case designs the database; the second designs the patient visit. You are designing the database.

Non-Functional Requirements

In addition to listing non-functional requirements in the List of Requirements, it is also possible to add a new section at the end: Non-Functional Requirements. Non-Functional Requirements include things like: programming language, system features, etc. Consider dividing this section into two parts:

1) Non-Functional Requirements Applicable to the System: This includes features you plan on implementing;
2) System Requirements: This includes options that should be set within the system during install or configuration time, that the new product will depend on.