Writing Requirements

CSCE 740 - Lecture 5 - 09/09/2015

Key Points

- A **requirement** is a singular documented physical or functional need that a particular product must be able to perform.
- Each requirement should be accompanied by a **specification** detailing how the requirement should be realized.
- Use templates to structure and clarify specifications.
- Requirement must still be well-written.
  - Precise, avoid amalgamation, make distinction between functional/non-functional
- The structure of the requirements document is of critical importance.

Today's Goals

- Learn how to write "good" requirements
  - Clear and testable
  - Not contradictory
  - Complete
  - Exit criteria
- Using checklists to avoid forgetting items

Easy Requirements Guidelines

- Avoid requirements “fusion”
  - One requirement per requirement entry in document.
    - One property the software must uphold.
- Be precise
  - No vague or incomplete requirements.
  - Define all outcomes of a condition.
- Stated in the positive
  - State what the software will do, not what it will not.
  - You can never prove a negative.

Easy Requirements Guidelines

- Be rigorous in defining test cases
  - If you cannot define how to test whether a requirement is satisfied, you have a poor requirement.
- Attach a person to each requirement
  - Assigning responsibility leads to better work, less feature inflation.

Requirement 1

The system shall validate and accept credit cards and cashier's checks. High priority.

Problem: Requirements Fusion

- validate and accept
  - If validation fails, can we still "accept"?
- credit cards and cashier's checks
  - Can you pay with a combination? What if credit card validation fails and the cashier's check is accepted?
- "high priority"
  - Are both payment methods high priority?

Requirement 2

Charge numbers should be validated online against the master corporate charge number list, if possible.

Problem: Vague Requirement

- How is it validated?
- What is the “master corporate charge number list”? Where is it stored?
- “if possible”? What would make it impossible?

Requirement 3

The software shall not support optical character recognition for converting scanned recipes to text.

Problem: Stated in the Negative

- How do you test whether a feature is not supported?
- There are an infinite number of things the software will not do - why state this one?

Requirement 4

If a failure occurs (either internal or external), an easy to interpret alarm must be raised quickly.

Problem: Untestable

- What does “raised quickly” mean?
- What does “easy to interpret” mean? You can’t test ease of interpretation - subjective quality.
- First problem can be fixed, second cannot.

The Properties of a Good Requirement

Each Requirement Must Be...

- Correct
  - The requirement is free from faults.
- Precise, unambiguous, and clear
  - Each item is exact and not vague.
  - There is a single interpretation.
  - The meaning of each item is understood.
  - The requirement is easy to read.
- Complete
  - The requirement covers all aspects of the property being asserted.
  - All failure scenarios covered - no question of what an outcome should be.

Each Requirement Must Be...

- Consistent
  - No statement contradicts another statement within the requirement or its specification.

“If the user selects the reset password option, an e-mail shall be dispatched with a unique, one-use reset link.”

“If the user selects the reset password option, they shall be prompted to enter a new password along with their e-mail address.”

Each Requirement Must Be...

- Relevant
  - Each item is pertinent to the problem and its solution

"The software shall not require Adobe Acrobat."

- Testable
  - During development and acceptance testing, it will be possible to determine whether the item is satisfied.

"An alarm shall be raised quickly."

"An alarm shall be raised within 10 seconds."

Each Requirement Must Be...

- Traceable
  - Each requirement should be linked to all related requirements, and back to its source.

“In order to withdraw funds, the card must be validated (Req 2.6) and the PIN must be entered correctly (Req 2.7).”

- Feasible
  - Each item can be implemented with the available techniques, tools, resources, and personnel, and within the specified cost and schedule constraints.

Each Requirement Must Be...

- Free of unwarranted design detail
  - The requirements specifications are statements that must be satisfied by the problem solution, but should not unnecessarily constrain the solution.

“The system shall store user information including name, DOB, address and SSN.”

“The system shall store user information in the User class including name (string), DOB (Date), address (string), and SSN (integer).”

Each Requirement Must Be...

- Prioritized
  - Each requirement must be classified according to its importance.
  - Essential for risk mitigation and development planning.

“The system must support credit card payment.”

“The system must support cash payment.”

Which is more important?

The Properties of a Good Requirements Document

The SRS (as a document) Must Be...

- Complete
  - All necessary requirements have been included. Do not forget abnormal and boundary cases.
  - Completeness can be **internal** or **external**:
    - Internal Completeness: The SRS specifies a complete system, leaving out no functionality or outcomes of calculations and actions.
    - External Completeness: The SRS specifies all functionality (and outcomes) that the customer has asked for.

Internal Completeness

- **Internal Completeness:** The SRS specifies a complete system, leaving out no functionality or outcomes of a piece of functionality.
  - If we have a requirement about what to do when a button is pressed, we need a requirement about what to so when it is released.
  - If we have a requirement about what to do when a button is pressed for more than 2 seconds, we need one for when it is released within 2 seconds.

External Completeness

- **External Completeness:** The SRS specifies all functionality (and constraints) that the customer has asked for.
  - If we specify all functionality related to button A, but the customer expects an additional button B, then we can be internally complete, but not externally complete.
  - External completeness is very difficult to achieve:
    - Need to know the customer's needs.
    - Customers change their minds, are vague.

The SRS (as a document) Must Be...

- Consistent
  - No item conflicts with another item in the document.
- Single Voice
  - Consistent level of detail and quality.
- Manageable and Modifiable
  - Things will change! Be able to accommodate requirements evolution.

Example Requirements

Example Requirement 1

The product shall provide status messages regarding background processing at regular intervals not less than every 60 seconds.

Problems:

- Regular intervals? “not less” than 60 seconds? What is the upper limit?
- What events need to trigger status messages?
- What are the status messages?
  - (Should be covered in separate requirements, and linked here.)

Example 1 Rewritten

1. The product shall provide status messages regarding background processing at intervals of 60, plus or minus 10, seconds.
   1.1. If background processing is progressing normally, the percentage of the background task processing that has been completed shall be displayed.
   1.2. A message shall be displayed when the background task is complete.
   1.3. An error message shall be displayed if the background task has not progressed for 10 seconds or failed.
   1.4. Status messages are logged to the log file maintained on the local file system.

Example Requirement 2

The product shall switch between displaying and hiding non-printing characters instantaneously.

Problems:

- Instantaneously is not testable - subjective.
- When is it switching? What causes this switch? Is it random?

Example 2 Rewritten

The text entry box shall switch between displaying and hiding non-printing characters within 200ms of mouse release of the display button in the quick function bar (Req 3.4).

Example Requirement 3

Charge numbers should be validated online against the master corporate charge number list, if possible. The system shall validate the charge number entered against the online master corporate charge number list. If the charge number is not found on the list, an error message shall be displayed and the order shall not be accepted.

Example Requirement 3 (Take 2)

1. The system shall validate the charge card number entered against the online master corporate charge card number list.
   1.1. If the charge card number is not found on the list, an error message shall be displayed and the order shall not be accepted.

Example Requirement 3 Rewritten

1. The system shall validate the charge card number entered against the online master corporate charge card number list.
   1.1. If the charge card number is not found on the list, an error message shall be displayed and the order shall not be accepted.
   1.2. If the charge card number is found on the list, a success message shall be displayed and the order shall be accepted.
   1.3. If the online master corporate charge number list is not available, an error message shall be displayed and the order shall not be accepted.

Example Requirement 4

1. The system shall activate pilot ejection through a red trigger on the backside of the right handle of the control stick.
   1.1. The ejection shall be activated upon button release only if the trigger is held down for more than five seconds.
   1.2. The ejection shall not be activated unless the safety mode has been disengaged (Req 2).

Example Requirement 4 Rewritten

1. The system shall activate pilot ejection through a red trigger on the backside of the right handle of the control stick.
   1.1. The ejection shall be activated upon button release only if the trigger is held down for more than five seconds.
      1.1.1. If the trigger is released prior to five seconds, the ejection shall not be activated, and the pilot display shall display a feedback message indicating that ejection was cancelled.
   1.2. The ejection shall not be activated unless the safety mode has been disengaged (Req 2).

Incomplete Requirements

Example Requirement 4

The system shall respond to all user requests within 2 seconds.

Problems:

- Too overarching - need to address different user requests one at a time, each might have different needs.
- Very likely to lead to inconsistent requirements.

Requirements Rationale

It is important to provide rationale with requirements.

Rationale: Survey ABC-345 indicates that withdrawals are highly desirable. The success of the product hinges on successful withdrawal of funds.

Rationale offers context:

- The **issues** that are being addressed.
- The **alternatives** that were considered.
- The **decisions** that were made to resolve the issues.
- The **criteria** that were used to guide decisions.
- The **debate** developers went through to reach a decision.

Requirements Rationale

It is important to provide rationale with requirements.

- Helps developers understand the application domain and why the requirement is stated in its current form.
- Very important when requirements are changed.
  - Reduces the chance that changes will have an undesired effect.

Why is Rationale Important?

Loading a boat on a car rack

1. The boat must have handles to help a person lift it.
2. The car rack must be padded so the boat can easily slide into the rack.
3. The boat must be lighter than 100 pounds.

Rationale: One person must be able to load the boat on the car rack.

Use Checklists to Avoid Forgetting Requirements

General Requirement Checklist

Are your requirements:

- Complete
- Consistent
- Correct
- Precise, unambiguous, and clear
- Relevant
- Testable
- Understandable
- Expressed in the user's language
- Traceable
- Feasible
- Prioritized
- Classified for stability
- Free of unwarranted design detail

Requirements Definition and Writing Style Checklist

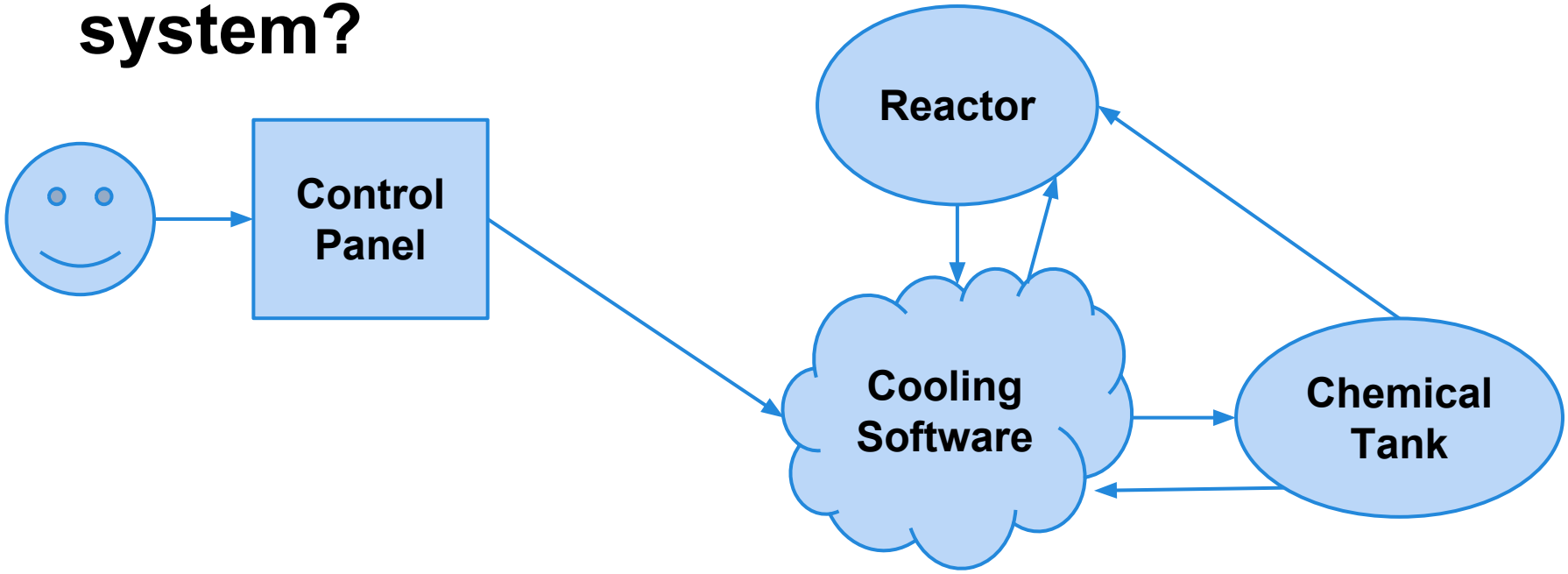
1. Have you varied the stress pattern in a sentence to reveal alternative meanings?
2. Could you commit to implementing this requirement within a week?
3. If a term is defined elsewhere, try substituting the term for its definition.
4. When a structure if described in words, try to sketch a picture of the structure.
5. When a picture describes a structure, try to redraw the picture in a form that emphasizes different aspects.
6. When there is an equation, try expressing the meaning of the equation in words.
7. When a calculation is specified or implied in words, try expressing it in an equation.
8. When a calculation is specified, work through at least two concrete examples by hand and give them as examples in the document.

Requirement Definition Checklist (2)

9. Look for statements that imply certainty and ask for proof.
10. Look for words that are there to persuade the reader, make sure the connected statements are actually backed up.
11. Watch for vague words and clarify those statements.
12. Watch for non-committal words.
13. Ensure lists are complete. If “etc” is used, make sure you know what “etc” means.
14. In attempting to clarify lists, we sometimes state a rule. Be sure that rules do not contain unstated assumptions.
15. Look for requirements without examples (or too few/too similar examples)
16. Avoid vague verbs.
17. Avoid passive voice. Passive voice does not name an actor.
18. Don’t make comparisons without clearly stating what is being referred.
19. Pronouns are often clear to the writer, but not the reader.

Domain-Specific Checklists

What problems and test scenarios can we anticipate in the automated cooling system?



Checklist for Embedded Systems

1. Is the software's response to out-of-range values specified for every input?
2. Is the software's response to not receiving an expected input specified?
   a. Are timeouts provided?
   b. Does the software specify the latency of the timeout?
3. If input arrives when it shouldn't, is a response specified?
4. On a given input, will the software always follow the same path through the source code?
5. Is each input bound in time?
   a. Does the specification include the earliest time at which it will be accepted and the latest time at which it will be considered valid?
6. Is a minimum and maximum arrival rate specified for each input?
   a. What if input arrives too often?
   b. Is there a capacity limit on interrupts?

Checklist for Embedded Systems (2)

7. If interrupts are masked or disabled, can events be lost?
8. Can software output be produced faster than it can be used by the receiving system?
   a. Is overload behavior specified?
9. Can all of the outputs from the sensors be used by the software?
10. Can input received before startup, while offline, or after shutdown influence the software's startup behavior?
    a. Are the values of any counters/timers/signals retained following shutdown? Is the earliest or most recent value retained?
11. In cases where performance degradation is the chosen error response, is the degradation predictable?
12. Are there sufficient delays incorporates into error-recovery responses?

Generality of Checklists

Domain-specific checklists focus on common pitfalls of one domain, but hold important lessons for other problems.

Use checklists to set expectations, but not to limit analysis and requirements refinement.

Checklists are Effective

On two NASA spacecraft projects, 192 critical errors were found during integration and testing.

- 142 of those were found and addressed after using a simple safety checklist.
- Most were problems with unexpected input.
  - Unexpected values, and more importantly, unexpected timing (recall the embedded system checklist).

When Are We Done?

When can we stop writing and revising requirements?

- Are the stakeholders happy?
  - Remember that the stakeholders are more diverse and numerous than you might think: client/customer, sales department, engineers, testers, etc.
- Have the documents passed all inspections and checklists?
- Have all TBD items been closed out?

The Use of TBD

- Any SRS using the term “to be determined” is not complete.

- TBD is often needed, however:
  - You need to include a description of the condition causing the TBD (why is cannot be resolved now)
  - You need a description of what must be done to eliminate the TBD, who is responsible, and a deadline on completion.

We Have Learned

- Use a standard document structure and forms for individual requirements.
- Make sure that requirements are consistent, complete, and testable.
- Use checklists to make sure that you don’t forget requirements.
- Keep an eye towards the future - things will change. Provide rationale and traceability.
- Make sure all stakeholders agree on the requirements document.

Next Time

- How to elicit requirements.
  - Understanding your stakeholders.
  - Formulating use cases.

- Reading: Sommerville, chapter 4
- Keep working on Assignment 1