Requirements Best Practices: A brief Glimpse

1. Basic Definition
   1. A requirement is a criterion that must be satisfied by a successful project completion
      1. Often some impact on the user
         1. E.g. some problem that has been solved
   2. Requirements gathering: attempt to discover what product is desired by people
   3. Quality requirements elicitation is extremely important and often oddly hard
2. Importance of requirements
   1. Unstable requirements are #1 or 2 reasons for runaway project
   2. To deliver it successfully, need correct understanding of what customers what
   3. Fixing in requirements phase far cheaper
3. Good requirements focus and simplify
   1. Cost of rebuilding system from scratch just 1-2% of original system cost
      1. Cf research, start-up that set own requirements
   2. Experiment with same requirements, different team goal
4. Requirements: A System view
   1. Requirements process quality
      1. Non-redundant development
      2. Maintenance effective
      3. Customer matching
      4. Development effectiveness
      5. Early fault elimination
   2. In short
      1. Benefit full iron triangle of time, cost, and quality
5. Benefits of good requirements
   1. Clearer expectations 🡺 better reception by customers
   2. Elimination of needless development
   3. Reduction in project risk
   4. Actual faults caught earlier
   5. Faster development
   6. Improved maintenance
   7. Lessened chance of cancellation
   8. Simpler formulation of testing strategy
   9. Robust design
   10. Documenting changed needs
   11. Re-engineering/converting code to new platform
6. Requirements statement problems
   1. Ambiguity (and implicit assumptions)
   2. Incompleteness
   3. Conflict
   4. Different origins and pedigree
   5. Different priorities
   6. Perceived stakeholder urgency to realize
   7. Origin with developers
   8. Observational error
   9. Recall error
   10. Unnecessarily specific
7. Derived Requirements
   1. Most requirements are not customer requirements
   2. Typically customer requirements lead to many more implied requirements
      1. Most of these relate to how
         1. Because must run on portable system, need to have small memory footprint, low power consumption, run under Android
         2. Because need to run in parallel on GPU to meet performance requirements, needs to be written in CPU C libraries
         3. …
   3. Missing these can be disastrous
8. Distinction Between
   1. Functional requirements: requirements dictating behavior of software
      1. Reference use cases
      2. Contain additional information beyond use cases
      3. Often themselves require dialogue about what is wanted/feasible
   2. Non-functional requirements: other non-behavioral constraints on software
9. Key Functional requirements GAP
   1. Client can’t readily
      1. Clearly envision solution
      2. Communicate required domain understanding
      3. Be aware of what is feasible
      4. Logical steps that must be taken in solution
      5. Understand technology implications of steps
   2. Technical team can’t readily
      1. Clearly understand domain-specific need
      2. Appreciate gap between what is now and what is sought
      3. Clearly envision solution
      4. Readily parse domain language
10. Functional Requirements
    1. User’s Perspective
       1. User type
       2. Result
       3. Object
       4. Qualifier
       5. Use cases valuable for many functional requirements
    2. System’s perspective
       1. Conditions
       2. Result
       3. Qualifier
11. Use Cases
    1. Tell story, focus on user goal
    2. Typically come before functional requirements
    3. Many components
       1. Pre
       2. Post
       3. Normal
       4. Alternate flow
       5. Exception
       6. Relevant business rule
    4. Can be understood by many user
    5. Work well for cases where much complexity in user interaction
    6. Can prod to consider things not considered in use cases
12. Use Case shortcomings: Poor For
    1. Non-user-facing functionality
    2. Complex series of conditions (better represented in table/tree)
    3. Sole description of functionality
    4. Giving dev. Integrated picture
    5. Non-primary user-facing task
13. Non-Functional requirements Examples
    1. Performance
    2. Footprint
    3. Platform limitations
    4. Reliability
    5. Availability
    6. Portability
    7. Flexibility for future development
    8. Security components
    9. Reusability
    10. Robustness to error
    11. Scalability
    12. Usability
14. Indirect Requirements
    1. Can have maintainability requirements
       1. Limits on function call nesting
       2. Requiring a certain level of commenting
       3. Time to modify report to
       4. Ease of replacing components
    2. Performance requirements
       1. Real time guarantees
       2. Response time
       3. Throughput
       4. Load time
       5. Time to perform an algorithm on a certain size input
       6. End to end time for transaction
15. Tips
    1. Repeat understanding of requirement to stakeholder
    2. Ask stakeholder to repeat requirement
    3. Recognize requirements docs as useful, living documents
    4. Put a priority on requirements traceability
       1. Design
       2. Reviews
       3. Test
       4. Code
    5. Prioritize requirements
    6. Structuring acceptance tests around requirements
    7. Focus on process – not documents
    8. Think about hidden requirements
    9. Start with a statement of the problem
       1. Often in vision document
    10. Use requirements to guide acceptance test
    11. Split up compound statements of Boolean conditions into multiple cases
        1. Case use tree/flowchart
        2. Otherwise can’t tell if exhaustive
    12. Try to avoid saying what won’t do
    13. Use tables for many cases
    14. Avoid ambiguous word
        1. Multiple meanings
        2. Culturally/geogr. Dependent meanings
        3. Unclear magnitude
        4. Might may …
    15. Create multiple views of complex requirements
    16. Use a requirement change process
16. Documents and test
    1. Development
       1. Requirements
       2. High-level design
       3. Detailed design
       4. Coding
    2. Testing
       1. Acceptance
       2. System
       3. Integration
       4. Unit
17. Sources of errors
    1. Missed ambiguity
       1. Interviewer may have jumped to conclusion on but doesn’t realize that made assumption about unmentioned issue
    2. Missing requirement
       1. Will not get mention of requirement under totally different conditions
          1. User don’t care about
          2. User care strongly about and assume that technical staff are aware of requirement
          3. User forget to think about issue
    3. Observational error (misheard statement)
    4. Recall error (miscalled statement)
    5. Completely neglect tacit requirements for entire areas
       1. Performance
       2. Operability
       3. Scalability
       4. Maintainability
       5. Security
       6. Integration with existing system
       7. Cutover for production/deployment
          1. Conversion of existing data
       8. Whole types of users
          1. Administrators
18. Documents
    1. Requirement-Oriented Deliverables include documents
    2. Do not get tied up in orthopraxy of document rules – focus on obtaining useful information to allow design to proceed and decrease risk
    3. Evolved requirements processes differ in division of material between documents
       1. Product request
       2. Marketing requirements documents (MRD)
       3. Vision/Scope
       4. System requirements Specification (SRS)
       5. First two documents contain a rough early understanding of project requirements
          1. Sometimes this material goes into SRS
19. Product Request
    1. Brief (<1) statement of
       1. Scope of problem
       2. Very general statements of what criteria a system would meet to satisfy this
    2. Circulated to stakeholders solicited for requirements meeting
20. Vision Statement Contents
    1. Problems statement
       1. Project background
       2. Stakeholders
       3. Customers
       4. Users
       5. Risks
       6. Assumptions
    2. Vision of the solution
       1. Vision statement
       2. Possible: organizational vale delivered
       3. List of feature (essential/desired)
       4. Scope of phased release
       5. Possible: work division across team
       6. Rough schedule
       7. Possible: what other teams are depending on this
       8. Features that will not be developed
21. Building vision/scope requirements statement
    1. Convert problems into “feature statements” or “scenarios” that product should support to address problem
       1. This is something that the customer can do because of the project
       2. Key here and in requirements document is to avoid describing any particular design to make feature possible
22. Ambiguity Tests on problem statement
    1. Ask for restating of problem statements or requirement by parties at requirements meeting
       1. Will tend to forget or reinterpret unclear points
       2. Then ask people if impression changed after seeing the statement again
    2. Emphasizing different words in problem statement
    3. Listing definition of key words