Current Requirements Practice Self-Assessment

This assessment contains 20 questions that you can use to calibrate your team's current requirements engineering practices and to identify areas to reinforce. Select the response for each question that most closely describes the way your team currently deals with that requirements issue. If you want to quantify the self-assessment, give yourself 0 points for each (a) response, 1 point for each (b), 3 points for each (c), and 5 points for each (d) response [except for question 16, where both (c) and (d) are worth 5 points]. The maximum possible score is 100 points. Generally speaking, the higher score, the more mature—and likely more effective—your requirements practices are. Each question refers you to the chapter or chapters *Software Requirements, 3rd Ed.* by Karl Wiegers and Joy Beatty that address the topic of the question.

Instead of just trying to achieve a high score, use this self-assessment to spot opportunities to apply new practices that might benefit your organization. Some questions might not pertain to the kind of software your organization develops. Also, situations are different; not every project needs the most rigorous approaches. Recognize, though, that informal approaches to requirements increase the likelihood that your team will end up doing excessive rework. Most organizations will benefit from following the practices represented by the "c" and "d" responses.

The people you select to complete the assessment could influence the results. Watch out for respondents who, rather than describing what's really going on in the organization, might bias their responses based on politics, on what they wish was being done, or on what they think the "correct" answers should be. Asking multiple people to complete the self-assessment independently will help remove some of that bias and provide a more realistic representation of your current practices than asking just one person. Multiple responders might also reveal different understandings of how certain practices are being performed at present. You can use the associated spreadsheet tool from http://aka.ms/SoftwareReq3E/files to accumulate multiple sets of responses and view the distribution.

1. How are the project's business requirements defined, communicated, and used? [Chapter 5]
   a. We sometimes write a high-level product description early on, but we don't refer back to it.
   b. The person who conceives the product knows the business requirements and discusses them verbally with the development team.
   c. We record business requirements in a vision and scope, project charter, or similar document according to a standard template. All project stakeholders have access to this document.
   d. We actively use the documented business requirements on our project, evaluating proposed product features and requirement changes to see whether they lie within the documented scope, and adjusting scope as needed based on business objectives.
2. How are the user communities for the product identified and characterized? [Chapter 6]
   a. The developers think they know who our users will be.
   b. Marketing or the project sponsor believes that they know who the users are.
   c. Target user groups or market segments are identified by management or marketing from some combination of market research, our existing user base, and input from other stakeholders.

d. The project stakeholders identify distinct user classes, whose characteristics are summarized in the software requirements specification.

3. How do you elicit customer input on the requirements? [Chapter 7]

   a. The developers are confident that they already know what to build.

   b. Typical users are surveyed with questionnaires or interviewed in focus groups.

   c. We meet with people, sometimes one on one and sometimes in groups, and they tell us what they want.

   d. A variety of elicitation techniques are used, including interviews and workshops with user class representatives, document analysis, and system interface analysis.

4. How well trained and how experienced are your business analysts? [Chapter 4]

   a. They are developers or former users who have little experience and no specific training in software requirements engineering.

   b. Developers, experienced users, or project managers who have had some previous exposure to requirements engineering perform the BA role.

   c. The BAs have had several days of training and considerable experience in collaborating with users.

   d. We have professional business analysts or requirements engineers who are trained and proficient in interviewing techniques, the facilitation of group sessions, technical writing, and modeling. They understand both the application domain and the software development process.

5. How are the high-level system requirements allocated to the software portions of the product? [Chapters 19 and 26]

   a. Software is expected to overcome any shortcomings in the hardware.

   b. Software and hardware engineers discuss which subsystems should perform which functions.

   c. A system engineer or an architect analyzes the system requirements and decides which ones will be implemented in each software subsystem.

   d. Knowledgeable team members collaborate to allocate portions of the system requirements to software subsystems and components and to trace them into specific software requirements. Component interfaces are explicitly defined and documented.

6. To what extent are requirements reused on your projects? [Chapter18]

   a. We do not reuse requirements.

   b. A business analyst who is familiar with previous projects sometimes knows of requirements that can be reused on a new project, so she copies and pastes them into the new specification.

   c. A business analyst can search through the previous projects stored in our requirements management tool for requirements that are relevant to his new project. He can reuse specific versions of those requirements by using the functions built into the tool..

   d. We have established a repository of potentially reusable requirements, which have been

adapted and improved from previous projects. BAs routinely check this repository for requirements that might be usable on their current projects. We use trace links to pull in child requirements, dependent requirements, design elements, and tests when possible when we are reusing a requirement.

7. What approaches are used when working with stakeholders to identify the specific software requirements? [Chapters 7, 8, 12, and 13]
    a. We begin with a general understanding, write some code, show the software to some users, and modify the code until they're happy.
    b. Management or marketing provides a product concept, and the developers write the requirements. Customer stakeholders tell the development team if they've missed anything.
    c. Marketing or customer representatives tell the development team what features and functions the product should contain. Sometimes marketing tells the development team when the product direction changes.
    d. We hold structured requirements elicitation interviews or workshops with representatives from the different user classes for the product. We employ use cases or user stories to understand the users' goals, and we create analysis models to help ensure we identify all the functional requirements. We flesh out the requirements incrementally and iteratively, giving the customers numerous opportunities to improve them.
8. How are the software requirements documented? [Chapters 10, 11, 12, and 30]
    a. We piece together oral history, email and voice mail messages, interview notes, and meeting notes.
    b. We write unstructured narrative textual documents, or we create simple requirements lists, or we draw some diagrams.
    c. We write requirements in structured natural language according to a standard template. Sometimes we augment these requirements with visual analysis models that use standard notations.
    d. We create requirements and visual analysis models and store them all in a requirements management tool. Several attributes are stored along with each requirement.
9. How are nonfunctional requirements, such as software quality attributes, elicited and documented? [Chapter 14]
    a. What are "software quality attributes"?
    b. We do beta testing to get feedback about how the users like the product.
    c. We document certain attributes, such as performance, usability, and security requirements.
    d. We work with customers to identify the important quality attributes for each product, which we then document in a precise and verifiable way.
10. How are the individual functional requirements labeled? [Chapter 10]
    a. We write paragraphs of narrative text or short user stories; specific requirements are not explicitly identified.

b. We use bulleted or numbered lists.

c. We use a hierarchical numbering scheme, such as "3.1.2.4."

d. Each discrete requirement has a unique, meaningful label that is not disrupted when other requirements are added, moved, or deleted.

11. How are priorities for the requirements established? [Chapter 16]

    a. All of the requirements are important, so we don't need to prioritize them.

    b. The customers tell us which requirements are most important to them. If the customers don't tell us or don't agree, the developers decide.

    c. Each requirement is labeled as high, medium, or low priority by customer consensus.

    d. To help us make priority decisions, we use an analytical process to rate the value, the cost, and the technical risk of each requirement, or we use a similar structured prioritization technique.

12. What techniques are used to prepare a partial solution and verify a mutual understanding of the problem? [Chapter 15]

    a. We just build the system and then fix it if we need to.

    b. We build some simple prototypes and ask users for feedback. Sometimes we're pressured to deliver prototype code.

    c. We create prototypes for both user interface mock-ups and technical proofs of concept when appropriate.

    d. Our project plans include tasks to create electronic or paper throwaway prototypes to help us refine the requirements. Sometimes we build evolutionary prototypes. We use evaluation scripts to obtain customer feedback on our prototypes.

13. How are the requirements validated? [Chapter 17]

    a. We think our requirements are pretty good when we first write them.

    b. We pass the specified requirements around to people to get their feedback.

    c. The BA and some stakeholders hold informal reviews when they have time.

    d. We inspect our requirements documents and models, with participants that include customers, developers, and testers. We write tests against the requirements and use them to validate the requirements and models.

14. How are different versions of the requirements documents distinguished? [Chapters 27 and 30]

    a. The document shows the auto-generated date that the document was printed.

    b. We use a sequence number—like 1.0, 1.1, and so on—for each document version.

    c. We have a manual identification scheme that distinguishes draft versions from baselined versions and major revisions from minor revisions.

    d. The requirements documents are stored under version control in a document management system, or requirements are stored in a requirements management tool that maintains a revision history for each requirement.

15. How are software requirements traced back to their origin? [Chapter 29]
    a. They aren't.
    b. We know where many of the requirements came from but don't document the knowledge.
    c. Each requirement has an identified origin.
    d. We have full two-way tracing between business requirements, system requirements, user requirements, functional requirements, and nonfunctional requirements.
16. How are requirements used as the basis for developing project plans? [Chapter 19]
    a. The delivery date is set before we begin requirements development. We can't change either the project schedule or the scope. Sometimes we go through a rapid descoping phase to drop features just before the delivery date.
    b. The first iteration of the project plan addresses the schedule needed to gather requirements. The rest of the project plan is developed after we have a preliminary understanding of the requirements. We can't change the plan much thereafter, however.
    c. We start with just enough information about requirements to prioritize them, then estimate the effort needed to implement the top-priority requirements. We develop our requirements and our software incrementally, planning the requirements for each iteration based on their priority and size. If we need to accommodate more requirements than our plan allowed, we add more iterations.
    d. We base the schedules and plans on the estimated effort needed to implement the required functionality, starting with the highest-priority requirements. These plans are updated as the requirements change. If we must drop features or adjust resources to meet schedule commitments, we do so as early as possible. We plan to deliver multiple releases to accommodate requirements changes and growth. [Note: (c) and (d) are equally good responses for this question.]
17. How are the requirements used as a basis for design? [Chapter 19]
    a. When we have written requirements, we refer to them during development.
    b. The requirements documents describe the solution we intend to implement.
    c. Each functional requirement is traced to a design element.
    d. Developers inspect the requirements to make sure they can be used as the basis for design. We have full two-way traceability between individual functional requirements and design elements.
18. How are the requirements used as the basis for testing? [Chapter 19]
    a. The testers test the software based on how they think it should function.
    b. The testers test what the developers said they implemented.
    c. We write system tests against the user requirements and functional requirements.
    d. Testers inspect the requirements to make sure they are verifiable and to begin their test planning. We trace system tests to specific functional requirements. System testing progress is measured in part by requirements coverage.

19. How is a software requirements baseline defined and managed for each project? [Chapters 2 and 27]

    a. We don't have to think about baselines because we are on an agile project.

    b. The customers and managers sign off on the requirements, but the development team still gets a lot of changes and complaints.

    c. We define an initial requirements baseline, but we don't always keep it current as changes are made over time.

    d. The requirements are stored in a requirements management tool when an initial baseline is defined. The requirements repository is updated as requirements changes are approved. We maintain a change history for each requirement after it's baselined. On an agile project, the team agrees on a requirements baseline for each iteration.

20. How are changes to the requirements managed? [Chapter 28]

    a. The requirements change whenever someone has a new idea or realizes that he forgot something.

    b. We discourage change by freezing the requirements after the requirements phase is complete, but informal change agreements are still made.

    c. We use a defined format for submitting change requests and a central submission point. The project manager decides which changes to incorporate.

    d. Changes are made according to our documented change control process. We use a tool to collect, store, and communicate change requests. The impact of each change is evaluated before the change control board decides whether to approve it.