syncroness
innovative product development

NOVEMBER 15, 2013

Syncroness, Inc.
Corporate Office
10875 Dover St., Unit 200
Westminster, CO 80021
Phone: (303) 429-5005
Fax: (303) 429-5025

WRITING EFFECTIVE REQUIREMENTS – REDUCING RISK AND ENSURING POSITIVE OUTCOMES THROUGH CLEAR & CONCISE SPECIFICATIONS

AUTHORED BY JERED DEAN & KYLE HARRISON, SYNCRONESS INC

THE PROBLEM

Many times, the success (or failure) of a product development effort is defined before any engineering or technical execution takes place. Ill-defined, under-defined and unclear requirements can lead to mis-guided implementation, disagreement of application and, worst of all, delays in product releases. Especially when working with tight deadlines and tighter budgets, knowing how to write good requirements can ensure positive outcomes and common understanding of scope and complexity where there otherwise may be none.

EXECUTIVE SUMMARY

As a provider of engineering solutions on a contract basis, we understand the importance of having a well-defined vision for new product development and its impact on the success of individual projects. Similarly, internal product development efforts (sans outsourcing) must also apply a good set of practices surrounding the development of product requirements. This paper describes the Syncroness method for writing good product requirements as well as how to capture, document, track and manage them throughout the project lifecycle. Using real project inputs and outputs, we explore both good and bad examples of product requirements.

WHY CAPTURE REQUIREMENTS?

In every product development effort, requirements - both acknowledged and implied -drive tasks (and cost) that must be executed and addressed prior to finalizing and releasing the product. Simply discussing requirements at a high level or half-heartedly documenting an outlined view of general requirements is not only a costly bad-practice, but may be against industry regulation. Particularly when working with an outsourced firm, requirements will not only define the project scope, but also serves as the initial understanding of what it will take to complete the project, how you will define success and how you know when the project is complete.

CAPTURING REQUIREMENTS

PRODUCT REQUIREMENTS SPECIFICATION (PRS)

A good requirements document is a succinct, complete summary of known design inputs. Each requirement included in the Product Requirement Specification (PRS) adds additional cost to the project (regardless of internal or external execution). Due to the cost of additional specifications, the goal should be to create the fewest requirements as possible while accurately capturing the end customer's needs. This can be particularly challenging as a service provider where we aim to exceed expectations and the natural tendency for engineers to gravitate towards new, cutting edge and exciting technologies and implementations - you must focus on the end goal.

The PRS itself consists of crucial information regarding the requirements and its relative state. Below is a list of the most common elements of the PRS which are used to manage, track and implement the requirements.

ID - A numerical identification number for the requirement, to be used throughout the project for reference.

Epic/Component - The subsystem or feature that a requirement is part of (database, reporting, user interface etc).

Specification - The requirement statement.

State - For less formal tracking; generally either Proposed or Firm.

Verification Activity - The ID number of the verification activity that will test the requirement.

Source - The originating document from which the requirement was derived.

INITIAL REQUIREMENTS

An obvious question is "where do I start"? A good place to begin mining requirements is to request existing documentation - preliminary requirements/specifications, request-for-quote, VOC/Marketing documents and regulatory design documents. In the absence of these documents, or as follow-on, it is best practice to discuss, illustrate and capture typical use cases and Functional Flow Block Diagrams for the product in question. Simply walking through the intended use of the product, from simple tasks as turning the device on, can generate a multitude of requirements. For example, take a medical device intended to monitor cardiac rehabilitation patients in a clinical setting. The client states that the end user "turns the device on and begins monitoring the patients". Had we not discussed in detail what that means and how it is done, we never would have determined that the workstation is located in a different room and the clinician needs to have wireless access to the patient database to process that day's activities.

WHERE DO THEY GO?

Soon after the project kickoff, the Project Manager or Systems Engineer should determine the best vehicle for capturing, tracking and managing requirements. For most regulated industries, requirements will be captured in a formal document which is stored in the project Design History File (DHF) and approved by pre-determined parties prior to release. For other industries or projects not bound by high-regulation, a simple excel file or Microsoft SharePoint site is adequate for managing requirements. The vehicle for capturing, maintaining and managing requirements is discussed in later sections.

TYPES OF REQUIREMENTS

Not all requirements are created equally. In fact, there are various different types of requirements to be aware of as they impact implementation, verification and, ultimately, project cost. The following requirement types are typical among complex, technology-driven products.

Functional Requirements
These requirements specify what the device does, focusing on the operational capabilities of the device and processing of inputs and the resulting outputs.

> Example: The device shall output test results for duration, average force and serial ID number in a .csx format.

Performance Requirements
Quantitative in nature, these requirements specify how much or how well the device must perform, addressing issues such as speed, strength, response times, accuracy, limits of operation, etc. This includes a quantitative characterization of the use environment, including, for example, temperature, humidity, shock, vibration, and electromagnetic compatibility. Requirements concerning device reliability and safety also fit into this category.

> Example: The device shall remain fully operational in temperatures ranging from 0 degrees Celsius to 50 degrees Celsius.

Interface Requirements
Interface requirements specify characteristics of the device which are critical to compatibility with external systems; specifically, those characteristics which are mandated by external systems and outside the control of the design team. One interface which is important in every case is the user and/or patient interfaces.

> Example: The device shall be compatible with 6-lead ECG cable part number xxx-xxx.

DEVELOPING REQUIREMENTS

WRITING GOOD REQUIREMENTS

A succinct list of requirements captured in the PRS is of limited use if the individual statements themselves are poorly worded. Writing good requirements requires concise, simple language, and precise English. The goal of a requirements statement is to be clear and unambiguous to anyone who reads it.

The majority of requirements should be written in complete, declarative sentences with an imperative voice. Requirements should be stated in complete sentences and end in a period. A declarative sentence simply states a fact or idea without requiring an answer or action on the part of the reader. Declarative sentences should not question or elicit an emotional response. The imperative voice is commonly used in engineering writing, but is especially important in specification writing. The imperative voice expresses a command and focuses attention on the verb more than the subject. A couple of good and bad specification writing are provided below:

Bad 1: If the user pushed a button, 2cc of fluid "A" shall be added to the cup by the system.

Bad 2: In proper operation, the user will request and receive 2cc of fluid "A" in the cup

Good: The system shall fill the cup with 2cc of fluid "A" upon receiving a user request.

Requirements should be broken down into single, testable statements. If a requirement cannot be verified with testing or some other type of verification activity (analysis, demonstration etc) then it is a meaningless requirement and may bring potential problems downstream during verification testing. Avoid combining multiple requirements into one statement wherever possible. The body of the requirement statement should not document the source or rationale for a requirement. Some examples are:

Bad: External hardware to be tamper resistant. Hardware interior to units is not restricted to tamper resistant variety.

Good: External hardware shall be #4 pinned-Torx head style. Interior hardware shall be #2 Phillips head style.

Bad: One component of the solution has a significant Ti02 component. This component is known to cause corrosion and build up on ball valves and filters.

Good: The system shall not use ball valves or filters.

Avoid defining implementation with requirement. A requirement should define how the system functions, how well it performs its intended purpose, and the budgetary and physical constraints place on the design team. A requirement should not force the design team to solve the design challenge with a specific technology or approach unless specifically stated by the customer. While some customers do request inclusion of particular parts or hardware, the specification writer must be careful not to create unintended constraints.

Bad: Front of base station will incorporate at least one touch sensor to verify that car is present for charging.

Good: The base station shall verify that a car is present before charging.

Bad: Design shall have some sort of tamper proof seal incorporated into the design. The intent is to allow the manufacturer to know if the guts of the unit have been tampered with.

Good: The system must provide a method to visually indicate if an individual has accessed the internal mechanism of the unit.

Verb tense is important. All requirements are not created equally. The following list is an excerpt from the INCOSE Systems Engineering Handbook and describes the common use of the forms of the verb "to be" as they apply to specifications:

- **Shall** - "Shall" requirements are demands upon the designer and the resulting product.
- **Will** - Statements containing "Will" identify future happening. It is used to convey an item of information, explicitly not to be interpreted as a requirement.
- **Must** - "Must" is not a requirement, but is considered to be a strong desire by the customer, possibly a goal.
- **Other forms** - "To be,""is to be,""are to be,""should," and "should be" are indefinite forms of the verb and they should be minimized when developing requirements. There are not requirements, but should be considered to be capabilities desired by the customer.

Keep language simple and avoid playing with synonyms. Typically, good writing is characterized by using a variety of sentence constructs, words and phrases. When writing a specification, use the simplest, most common word and phrases. In addition, once you have used a word, avoid using synonyms in subsequent specifications. Synonyms never have exactly the same meaning and can cause unintended confusion.

There are a few other common mistakes that need to be avoided. Below is a list of some of the more common errors made when writing specifications:

- Avoid superlatives ("best,""most," etc) and ambiguous qualifiers ("significant, ""minimal,""approximately,""real-time," etc)
- Avoid comparative specifications ("better than,""faster than," etc)
- Do not add loop holes to a specification ("if possible,""as appropriate," etc)

MANAGING REQUIREMENTS

EVOLVING/CHANGING REQUIREMENTS

It is not uncommon that requirements are in need of further discussion, disposition or research prior to solidification. While not ideal, requirements may evolve during the project and, therefore, need to properly managed and documented. This is particularly common in new product development efforts where the technology being developed is new and not heavily based on an existing product. When possible, requirements should be firm prior to moving into design and development.

In the event a requirement evolves, needs to be changed or, in some cases, a new requirement needs to be added, it is imperative that the team review the updated requirement and its impact to the project from a technical, budgetary (including Cost of Goods Sold (COGs)) and schedule perspective. Regardless of whether there is a direct cost impact, the change should be captured in a formal engineering change order to provide traceability and agreement that the requirement has changed or the project scope has been updated. Formally tracking the changes makes it clear to the design team, project sponsor and stakeholders that the project has be redefined, even if only in a minor way.

Following disposition of the updated requirement, the PRS should also be updated and released as the latest revision in addition to impacted verification and traceability documentation.

TRACEABILITY AND VERIFICATION

Along with providing guidance to the design team, requirements also serve as the defining component for traceability and verification, especially in regulated environments.

As discussed previously, each requirement is mapped back to a specifically defined verification activity that will serve as evidence that the requirement has been satisfied. This is both for the formal verification execution at the end of the project as well as a convenient way for the design team to test changes 'ad-hoc' as they are made - substantiating a higher quality product. In some cases, the verification activity will be a component of the PRS while in others, it may be a stand-alone verification plan. Regardless, the requirement and verification activity references shall remain consistent for traceability. An example is provided below.

Requirement 3.9.7: Crosshead Travel
The crosshead travel of the device shall be maximized at 500mm.

Verification Matrix:

Requirement	Test Method	Test Level	Protocol
3.9.7 Crosshead Travel	Test	System	4.2.2

The above example shows the requirement statement as seen in the specification section of the PRS as well as the verification matrix from the verification section of the PRS. In this case, the team has indicated that a test procedure will be conducted at the system level to verify the requirement 3.9.7 and that procedure can be found in protocol 4.2.2

This example provides auditable evidence of traceability. The management of the requirements tells a story to interested parties and answers questions like "where did this requirement come from,""how do you know the requirement has been satisfied," etc.

FREQUENTLY ASKED QUESTIONS

What if I know a requirement is coming for a particular area, but don't know what the actual requirement is yet?

At times, the customer may not be prepared to provide exact requirements for the PRS (pending VOC, need input etc). The best practice is to add a place holder item in the PRS document. Give the specification an appropriate title and put "TBD" in the description field. By consistently using "TBD" in the description field, you allow yourself to easily search and sort later. Set the status to "unknown"

Is this requirement essential?

Per the Code of Federal Regulations Title 21, "those design outputs that are essential for the proper functioning of the device"...shall be identified. In general, any requirement that deals with human safety is essential. In practice the customer should decide if the requirement is essential. Use of the "Essential" category in the PRS is only required under the full control process. Check the project Model Control Checklist to see if you are required to use the column at all.

What do I do if I've been provided a poorly written specification?

At times, clients have been hesitant in requesting that Syncroness execute a requirements phase in a project, believing they have a good grasp on the requirements. They may even have them documented.

However, as we've discussed, requirements must be written properly to avoid costly future confusion. The best response is to modify or clarify them, then review them with the customer to make sure that the updates have maintained the original intent. Do not change customer supplied requirements without customer review and approval. Reserve the "Firm" status for only those requirements that are unambiguous, quantitative, verifiable, and customer approved.

ABOUT SYNCRONESS

Syncroness is a contract engineering firm located in Westminster Colorado. With specialties in mechanical, electrical, software and firmware engineering, we focus on supporting our clients through our three core services: New Product Development, Sustaining Engineering and Production Equipment Design. Flanked by professional project managers and systems engineers, the company concentrates on serving highly-regulated industries including medical device, aerospace and defense.

UNAMBIGUOUS REQUIREMENTS LEAD TO HAPPY OUTCOMES

While at times it may feel tedious, over-detailed or even a duplication of efforts, we have found that be routinely reviewing and capturing effective requirements, we are able to establish much better relationships with our clients and teams. Design teams appreciate a focused approach that provides freedom in creativity (a luxury not often experienced when working on fixed-price/schedule projects) and customers appreciate the predictable results and smooth-running projects.

CONCLUSIONS

Through the methods and best practices discussed in this paper, it is concluded that product developers and design teams can benefit from writing clear, concise and effective requirements. Moreover, identifying, documenting and establishing those requirements early in the project will ensure common understanding of scope and cost of the effort while proper change management will minimize future disagreements and conflict. By writing quantifiable and declarative requirements and tracing them to verification activities, the team will know when it is done designing the right product and have appropriate evidence to justify confidence in quality and the release of the product to the field. Even for less-formal or less-regulated product development efforts, writing effective requirements is a best practice that should be applied, especially if dealing with outside contractors, vendors and suppliers.

Want more information? This white paper is an overview of the best practices identified by the International Council on Systems Engineering (INCOSE) as well as Syncroness, Inc. For more information, see the Systems Engineering Handbook, Technical Process section available at www.incose.org or contact Syncroness directly at sales@syncroness.com.