Requirements Notes

Note: Most of these notes were taken from the IEEE Xplore Textbook and the citation is stated below.

"Citation Format," *2009 42nd Hawaii International Conference on System Sciences*, 2009, pp. 1-1, DOI: 10.1109/HICSS.2009.109.

General Requirements Statment Notes

Requirement statements are typically stored in a document as a proposal to an authority, meaning it states a business problem or opportunity and seek funding or approval to conduct project identification stage activities. It is important to remember that requirements help communicate the customer's needs and problems through stakeholders establishing a consensus of needs to solve the customer's problem. Requirement statements carry this overall goal so it is important to have quality, concise, in-scope, requirement statements that the client and development team are both satisfied with. Below gives a more in-depth analysis of the parts of requirements that individually make a whole and how they feed into achieving the overarching goal. Below is a great diagram that breaks into all of the sectors of software requirements showing that there is an immense amount of direction and detail for requirements to go in, all to paint a quality picture of what a client can expect from their product.

Characteristics of a set of requirements (Source: 29148-2018 5.2.6)

The link above takes you to the cited source of the book that is used to define this section. It is important to note there are certain characteristics that are considered for the set of stakeholder, system, and system element requirements. These sets of requirements provide a consistent solution that meets the stakeholder intentions and constraints. Below is a list of characteristics requirements should follow to keep consistency:

- **Complete** - The set of requirements stands alone such that it sufficiently describes the necessary capabilities, characteristics, constraints, or quality factors to meet entity needs without needing further information. Ensure that your requirements have none of the to be defined, specified, or resolved clauses as this will lead to the possible timeframe and acceptance issues. When trying to improve the completeness of requirements you can adopt the following practices:
  - include all requirements types relevant to the system under consideration.
  - account for requirements in all stages of the life cycle.
  - involve all stakeholders in the requirements elicitation, capture, and analysis activity.
- **Consistent** - The set of requirements contains individual requirements that are unique, do not conflict with or overlap with other requirements in the set, and the units and measurement systems are homogeneous. The terminology used within the set of requirements is consistent, i.e. the same term is used throughout the set to mean the same thing.
- **Feasible** - The complete set of requirements can be realized within entity constraints (e.g., cost, schedule, technical) with acceptable risk.
- **Comprehensible** - The set of requirements is written such that it is clear as to what is expected by the entity and its relation to the system of which it is a part.
- **Validatable** - the requirements are able to be validated. It is practicable that satisfaction of the requirement set will lead to the achievement of the entity's needs within constraints like cost, legal, schedule, or any sort of compliance.

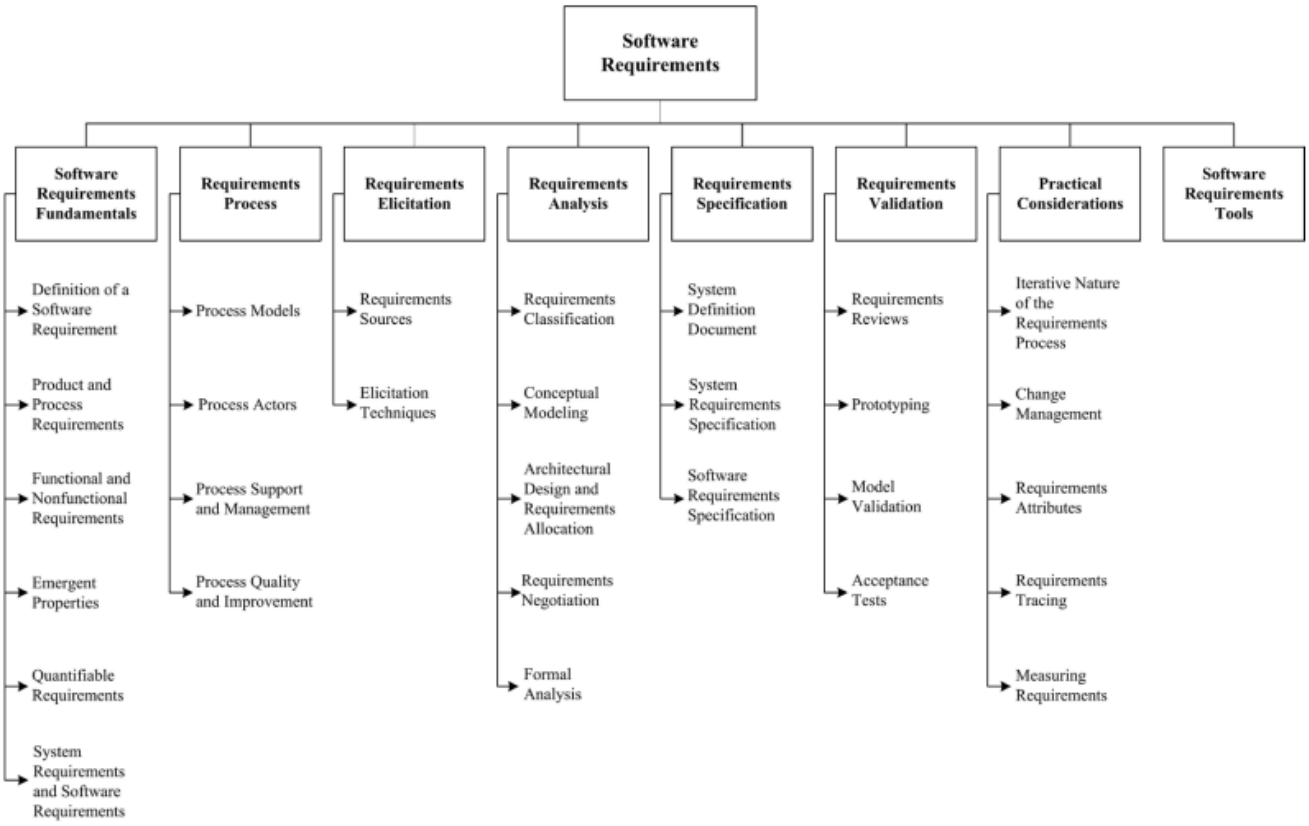
It is highly important to be careful checking the requirements set for these characteristics. it is also is critical to avoid changing requirements and requirement growth during the life cycle that will impact the cost, schedule, or quality of the system. These concepts will allow you to beat out that as a possibility during your elicitation.

Set of Requirements

Key	Summary	Description	T	Linked Issues	P	Labels
CCM-74	The user interface shall use the same font across all text to ensure it stays uniform	This keeps the software more uniform during the UI design phase				
CCM-73	The system shall have a set of error messages if an action fails or cannot be completed	This helps with debugging from a management perspective although it requires some technical knowledge.				
CCM-72	The system shall not use any audio queues as it will be used outdoors	This is a design function to ensure that audio is never used within the software design.				
CCM-71	Employees will use Member ID cards as a proof of identification	This acts as a means of security instead of people memorizing member ID numbers.				
CCM-70	A photo of the member pops up during check in for an employee to objectively validate their person	While this is not a highly accurate failsafe it gives the employee a general idea if a member is who they claim to be as a layer of security.				
CCM-69	Member numbers shall be six digets long with a number between zero and nine for each digit.	This keeps the membership numbers uniform.				

CCM-68	Immediate family to the member shall have a sub-member ID with a dash and letter following the ID number (123456-A)	This gives employees the ability to differentiate the main membership holder from their immediate family.				
CCM-67	The interface shall share the same color schematics to keep the software uniform	Keeping the system uniform helps play into a simple feeling UI				
CCM-66	The system shall require a username and passowrd for admin functionality to be accessed	This allows a restriction from the functionality that the company would not want any employee to use				
CCM-65	The system shall display dates in the format mm/dd/yyyy	This interface requirement will keep dates consistent when employees and management use the software.				
CCM-59	The system shall remove unavailable dates from the reservation list	This is a mandatory function to deter scheduling collisions within the system.				functional
CCM-58	The system shall give an estimation of the capacity for the club website	While it does not have to be exact, this is a requirement the BOT was interested in potentially deploying.				functional
CCM-57	The system shall retrieve member data in less than a minute	This nonfunctional requirement is important as members will be checked in at high rates during the season.				non-functional
CCM-56	The system shall provide available reservation dates	This is a mandatory function as this digitizes the reservation system.				functional
CCM-55	The system shall provide all members checked in during operation hours	This is a log used for the club in case of any legal records are needed.				functional
CCM-54	The system shall search members by ID, First name, or Last name	This is a mandatory function, as it is the entire reason for the existence of the system.				functional
CCM-53	The system shall store a physical copy on the device	This is a non-functional requirement as the location of the club is by the ocean leading to potential poor weather function.				non-functional
CCM-52	The system shall handle one hundred thousand individual member tuples	This is a minimum that the database must handle without any collisions of data.				non-functional
CCM-51	The system shall provide statistics of peak member check-in hours	This will give management an idea of the busier hours during season/nonseason to help with scheduling employees.		CCM-31		functional
CCM-50	The system shall send a conformation email to the member when a reservation is booked	The conformation email will help keep members in check with their reservation and act as proof that the reservation was confirmed by the employee.		CCM-30		functional

20 issues



Product and Process Requirements

A product requirement is a need or constraint on the software to be developed, whereas a process requirement is a constraint on the development of software. An example of a product requirement could be “The software shall verify that a student meets all prerequisites before they can register for a course”, whereas a process requirement example could be “The software shall be developed using a RUP process”. It is important to differentiate between the two as some software requirements will generate implicit process requirements, and this could potentially lead to confusion. Being able to correctly know the difference and apply product and process requirements will allow for your requirements documentation to have more depth, in turn generating better product scope.

Functional & Nonfunctional Requirements

Functional requirements describe the functions that the software is to execute, whereas non-functional requirements are used to act as constraints on the solution. While this sounds confusing at first nonfunctional requirements are sometimes known as constraints or quality requirements. They can be further classified according to whether they are performance requirements, maintainability requirements, safety requirements, and more. Functional requirements on the other hand are sometimes known as capabilities or features. A functional requirement can also be described as one for which a finite set of test steps can be written to validate its behavior. It is important to differentiate functional from nonfunctional when conducting your Business Analysis, as you do not want to have any confusion.

Requirements Construct (IEEE Chapter 5.2.4)

A requirement can be written and should include a subject and verb coupled with other information to properly express the information. It is important that any keywords used in a requirement are predefined. Some great rules to keep quality requirements are as follows below:

- Requirements are binding and use the word ‘shall’
- The best verbs to use when stating requirements are ‘are’, ‘is’, and ‘was’, it is important to avoid the term ‘must’ as this leads to potential misinterpretation
- when expressing a fact use the verb ‘will’ to stipulate context or limitation of use
- for expressing preferences or goals use the word ‘should’ as this is not entirely binding since it has the potential subject to change
- Suggestions or allowances should follow after the term ‘may’

- Using positive statements is important, so avoid using terminology such as ‘shall not'
- Use the active voice and avoid terms like ‘shall be able to’

Below is a great tool to use as a reference when following these guidelines:

[Condition] [Subject] [Action] [Object] [Constraint of Action]

EXAMPLE: When signal x is received **[Condition]**, the system **[Subject]** shall set **[Action]** the signal x received bit **[Object]** within 2 seconds **[Constraint of Action].**

Or

[Condition] [Subject] [Action] [Object] [Constraint of Action]

EXAMPLE: At sea state 1 **[Condition]**, the Radar System **[Subject]** shall detect **[Action]** targets **[Object]** at ranges out to 100 nautical miles **[Constraint of Action].**

Or

[Subject] [Action] [Constraint of Action]

EXAMPLE: The Invoice System **[Subject]** shall display pending customer invoices **[Action]** in ascending order of invoice due date **[Constraint of Action].**

Another great way for requirements to arise is from the constraints that require the system implementation to be specific. Things like physical size limitations, budget, the technology to be used, user capabilities, and more. As these things are factored it helps the business analyst create a high-quality set of requirements that keep the project in scope.

Characteristics of Individual Requirements (IEEE Chapter 5.2.5)

each stakeholder, system, and system element must possess the following characteristics listed:

- **Necessary** - The requirement defines an essential capability, characteristic, constraint, and/or quality factor. The requirement is currently applicable and has not been made obsolete by the passage of time. Requirements with planned expiration dates or applicability dates are clearly identified.
- **Appropriate** -The specific intent and amount of detail of the requirement are appropriate to the level of the entity to which it refers (level of abstraction appropriate to the level of entity).
- **Unambiguous** -The requirement is stated in such a way so that it can be interpreted in only one way. The requirement is stated simply and is easy to understand.
- **Complete** - The requirement sufficiently describes the necessary capability, characteristic, constraint, or quality factor to meet the entity's need without needing other information to understand the requirement.
- **Singular** - The requirement states a single capability, characteristic, constraint, or quality factor.
- **Feasible** - The requirement can be realized within system constraints (e.g., cost, schedule, technical) with acceptable risk.
- **Verifiable** - The requirement is structured and worded such that its realization can be proven to the customer’s satisfaction at the level the requirements exist. Verifiability is enhanced when the requirement is measurable.
- **Correct** - The requirement is an accurate representation of the entity need from which it was transformed.
- **Conforming** - The individual items conform to an approved standard template and style for writing requirements, when applicable.

Requirements Language Criteria (IEEE Chapter 5.2.7)

Requirements should state 'what' is needed, not 'how'. Vague and general terms shall be avoided, as they result in requirements that are often difficult or even impossible to verify or may allow for multiple interpretations. Below is a compiled list of ambiguous terms that should be avoided when creating requirements to prevent confusion from developers or clients.

- **Superlatives** - ‘best' or 'most'
- **Subjective Language** - ‘user friendly’ or ‘cost effective’
- **Vague Pronouns** - ‘it’, ‘this’, or ‘they’
- **Ambiguous Terms** - ‘almost always’ or ‘minimal’
- **Non-verifiable Terms** - ‘provide support’, ‘not limited to’, or ‘at a minimum’
- **Comparative Phrases** - ‘better than’ or ‘higher quality
- **Loopholes** - ‘as appropriate’, ‘if possible’, or ‘as applicable’
- **Totality Terms** - ‘all’, ‘always’, or ‘never’

Requirements Attributes (IEEE Chapter 5.2.8)

To support requirements analysis, well-formed requirements should include descriptive attributes defined to assist in identifying relevant requirements and to help in understanding and managing the requirements. Below are some examples of great requirement attributes that can be used to achieve this goal.

- **Version Number** -This is to make sure that the correct version of the requirement is being implemented as well as to provide an indication of the volatility of the requirement.
- **Owner** - The person or element of the organization that maintains the requirement, who has the right to say something about this requirement, approves changes to the requirement, and reports the status of the requirement.
- **Stakeholder Priority** -The priority of each requirement should be identified. This may be established through a consensus process among potential stakeholders. A great way to do this is with a value system between 1 - 5.
- **Risk** - A risk value assigned to each requirement based on risk factors. Requirements that are at risk include requirements that fail to have the set of characteristics that well-formed requirements should have.
- **Rationale** - The rationale for establishing each requirement should be captured. The rationale provides the reason that the requirement is needed and points to any supporting analysis, trade study, modeling, simulation, or other substantive objective evidence.
- **Difficulty** - The assumed difficulty for each requirement should be noted. This aids with cost modeling
- **Type** - Requirements vary in intent and in the kinds of properties they represent. The use of a type attribute aids in identifying relevant requirements and categorizing requirements into groups for analysis and allocation.

In conjunction with the use of requirement attributes, quality requirements should also include requirements types to identify in order to help with the management of these requirements. Below is a list of some examples within requirement types

- **Functional/Performance** - Functional requirements describe the system or system element functions or tasks to be performed by the system.
- **Interface** - Interface requirements are the definition of how the system is required to interact with external systems, or how system elements within the system, including human elements, interact with each other.
- **Process Requirements** - These are stakeholder, usually acquirer or user, requirements imposed through the contract or statement of work. Process requirements include compliance with national, state, or local laws, including environmental laws, administrative requirements, acquirer/supplier relationship requirements, and specific work directives.
- **Quality** - Include a number of the 'ilities' in requirements to include, for example, the user's needs. Usability/Quality-in-Use requirements are developed in conjunction with and form part of, the overall requirements specification of a system.
- **Human Factor** - State required characteristics for the outcomes of interaction with human users (and other stakeholders affected by use) in terms of safety, performance, effectiveness, efficiency, reliability, maintainability, health, well-being, and satisfaction.

Sample Requirements Statement

The Commons Club Membership System shall be able to display checked-in members in ascending order of time.

Characteristics of this individual Requirement

- **Conforming**: This requirement uses syntax templates from 29148-2018 5.2.4 Figure 1 — Examples of functional requirements syntax.
- **Singular** - This requirement states a single capability, characteristic, constraint, or quality factor.
- **Necessary** - The requirement defines an essential capability, characteristic, constraint, and/or quality factor.

Requirement Language Criteria

The requirement uses the verb shall, and lacks the use of any superlatives, vague pronouns, ambiguous terms, or any sort of language of nature that has been quantified by the section provided above. There are no vague or general terms making this requirement hard to verify or highly interpretative.

Requirements attributes

- **Identification**: Member.checkinlist would be a good way to identify this requirement as it produces the list of all current members who are currently checked in.
- **Stakeholder Priority**: This would be stakeholder priority 5 as this works as a verification once the check-in is completed. It is not as important as the check-in itself but it is nearly just as important since the company would want to know that a member has been verified before allowing them to enter into the commons club.
- **Type**: This is an example of a product requirement as it requires the product to have the specific feature where a last of members can be viewed after check in.