Defining, Documenting, and Managing Requirements

1 Introduction

1.1 Requirements Definition

For software products, the IEEE (Institute of Electrical and Electronic Engineers) *Software Engineering Glossary* [IEEE83] provides the following definition of requirements:

1. *A condition or capability needed by a user to solve a problem or achieve an objective*

2. *A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document. The set of all requirements forms the basis for subsequent development of the system or system component.*

The set of R*equirements document* provides a precise and consistent statement of all client/user requirements, a description of all client/user expectations, and a list of constraints that define a bounded Solution Space for the product.

1.2 Goals of Requirements Specification

- ☐ Requirements establish a common understanding between client/user and the project team of functionality, characteristics, and behaviors of the product.
- ☐ Requirements are the basis for development of project plans and activities, product and software design, and product maintenance.
- ☐ Well-documented and traceable requirements ensure that plans, activities, products are consistent with the original user requirements.
- ☐ User Acceptance Criteria establish constraints for each requirement attribute and determine if the completed product and software design satisfies the documented requirements

1.3 Requirements Management - Definition

- ☐ Process that creates and maintains an agreement between the client and project team on what will be built
- ☐ Process to identify and allocate requirements from the system level to the component level (e.g., software, hardware, documentation, training)
- ☐ Process of verifying and analyzing requirements
- ☐ Process to manage change and additions to requirements
- ☐ Process to monitor status of requirements

1.4 Types of Requirements

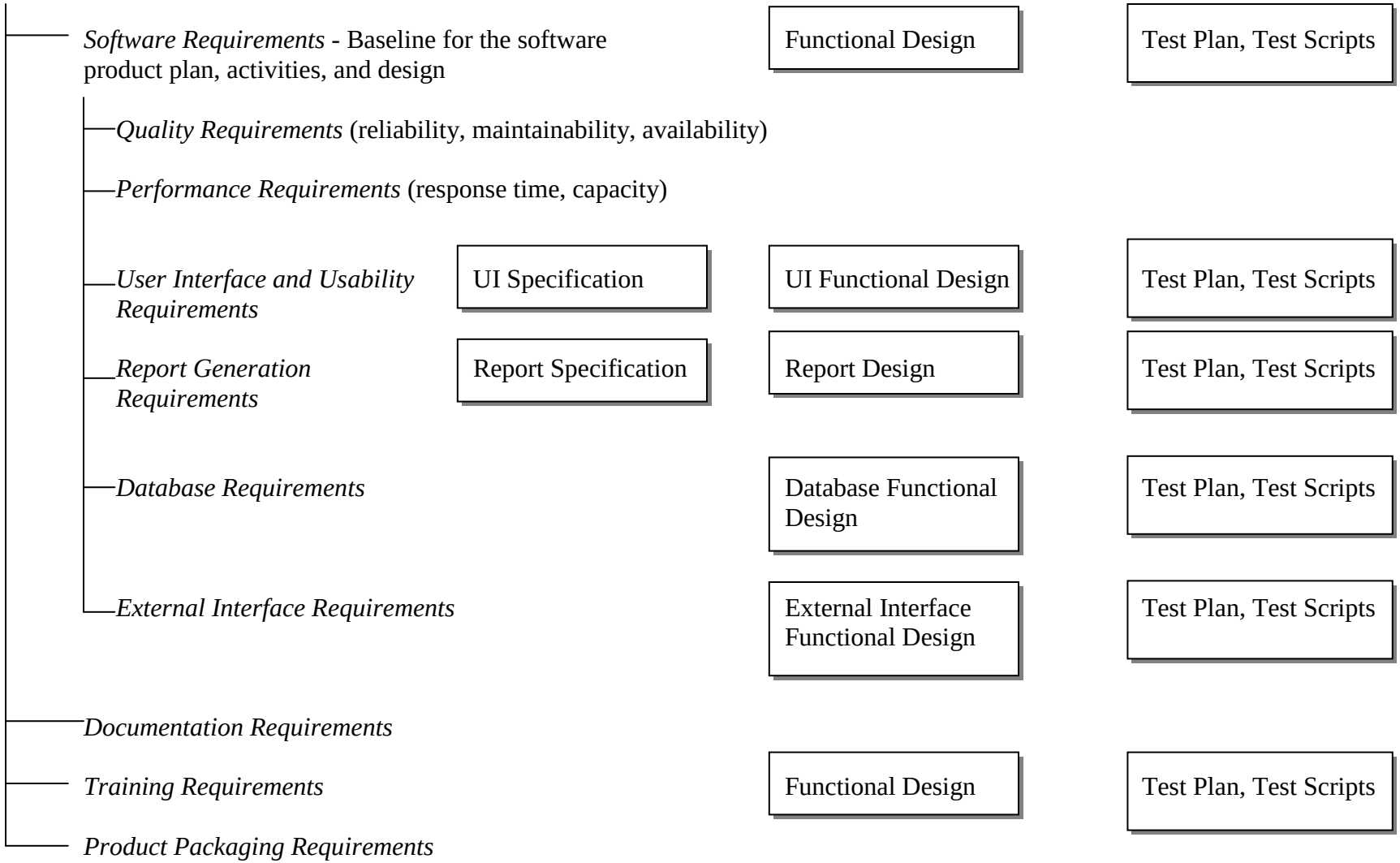
Below are the three types of requirements that should be captured, documented, and tracked:

1. **Functional Requirements**
   - ☐ Functions/services performed/expected by the end-user, support person, operator
   - ☐ Reports generated (information reported, frequency, volume, delivery media)
   - ☐ Error handling and messages
   - ☐ Do not address "how to" implementation details

2. **Non-functional Requirements**
   - ☐ Expectations of the overall product characteristics and qualities that must be achieved by the design
   - ☐ Constraints under which the system must operate
   - ☐ System performance (capacity, resources, reliability, availability, accuracy, response time, security)
   - ☐ Usability objectives
   - ☐ Software characteristics (efficiency, flexibility, maintainability, portability)
   - ☐ Operational characteristics (recovery, audit capability, built-in support)
   - ☐ Standards which must be met
   - ☐ Interface requirements (internal/external devices or modules, data/file transfer)
   - ☐ Internationalization

3. **Non-technical Requirements**
   - ☐ Project management expectations and constraints
   - ☐ Deliverables and delivery dates
   - ☐ Staffing plan, resources
   - ☐ Product schedule, budget/costs,
   - ☐ Activities and milestones associated with the product

System Requirements - Requirements for hardware, software, documentation, training



1.5 Requirements Management – CMM[1]

The CMM Key Process Areas (KPAs) for Level 2 compliance (Repeatable) are as follows:

1. Requirements Management
2. Software Project Planning
3. Software Project Tracking and Oversight
4. Software Quality Assurance
5. Software Configuration Management

The SEI/CMM defines Repeatable as: "Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications."

Requirements are defined as an agreement between the client/user and the project team. This agreement is used as the foundation for estimating, planning, performing, and tracking the software project's activities throughout the lifecycle of the software.

System Requirements are *allocated* to software, hardware, documentation, training, etc. Any change to System Requirements ultimately affects the allocated requirements and the Project Plan, project activities, and product deliverables.

1.5.1 Commitment to Perform

CMM Level 2 compliance requires that the organization and project team follow a documented, approved, and agreed upon policy for managing System Requirements allocated to software.

1.5.2 Ability to Perform

Within each software project, responsibility must be established to analyze and allocate System Requirements to hardware, software, etc.

Allocated System Requirements must be documented.

[1] CMM Practices – Level 2: Repeatable, Requirements Management (CMU/SEI-93-TR-25)

Resources to perform requirements management and documentation must be allocated within project team. Resources must be provided to fund a team member with requirements management capability and this individual must be provided with the required tools for documentation and management. Other related tools included Software Configuration Management (SCM), Requirements Traceability, and Test Management.

All members of the project team and related support and development groups must be trained and knowledgeable of their role in requirements management.

1.5.3 Activities Performed

The project must review and analyze requirements allocated to software before they are included as part of the Software Project. Requirements must be complete, unambiguous, precise, comprehensive, consistent, feasible, and testable.

Commitments to perform must be obtained from all groups affected by the Project/Release requirements (e.g., engineering, system test, SQA, SCM, documentation, training).

Requirements allocated to software are used as the foundation for software plans and activities.

Requirements must be managed and controlled. This implies a process for version control (being able to far back to a previous state within the project) and change control. Processes for Software Configuration Management are recommended.

All changes to allocated software requirements are reviewed, approved, and integrated into the Project Plan and affected deliverables. Changes must be negotiated and communicated with any engineering or support group with a commitment to perform.

All changes to allocated software requirements must be evaluated, assessed for risk, documented, planned, communicated, and tracked to completion.

1.5.4 Measurement and Analysis

Metrics are collected and used to determine the status of requirements and requirements management activities:

- ☐ status of each allocated requirements
- ☐ number of original requirements
- ☐ number of changes to requirements submitted, approved, and rejected
- ☐ status of all requirements changes
- ☐ status of planned and scheduled requirements management activities (e.g., formal inspections, estimates, analysis)

1.5.5 Verifying and Implementation

Requirements management activities are periodically reviewed by Senior Management.

Requirements management activities are periodically reviewed with the Project Manager.

Organization responsible for Software Quality Assurance (SQA) routinely reviews and audits the requirements management activities and work products.

1.6 Requirements Management – Roles and Responsibilities

Below are the key roles and responsibilities that should be established to implement Requirements Management:

Product Manager
- ☐ Responsible for the collection of all requirements
- ☐ Responsible for communicating aspects of the requirements with the client/user (e.g., impact of change/addition, technical feasibility)
- ☐ Responsible for all negotiations with the client/user.

Configuration Manager
- ☐ Maintains all client/user approved requirements
- ☐ Responsible for the requirements change control process
- ☐ Responsible for applying approved changes to the requirements documents
- ☐ Responsible for maintaining a requirements modification history

Configuration Control Board (CCB)
- ☐ Analyze and evaluate requirements for feasibility and impact to the project.

Client/User
- ☐ Responsible for defining and approving requirements
- ☐ Responsible for all modifications to requirements

1.7 Requirements Management – Processes and Procedures

Identification of Requirements

☐ Product Manager submits preliminary requirements to the CCB to review feasibility.
☐ Product Manager reviews findings of the CCB with the client/user, negotiates changes as required, and obtains approval to proceed.

Documenting Requirements

☐ Configuration Manager documents and numbers each requirement
☐ Configuration Manager maintains a shared repository of all requirements documents and a revision history of each document

Modification of Requirements (initiated by Client/User)

☐ Client/user submits a change request to the Product Manager.
☐ Product Manager submits change to the CCB for analysis of feasibility and impact on the project.
☐ Product Manager communicates the analysis of the CCB to the client/user prior to any changes being implemented.
☐ Configuration Manager updates the requirements documents as appropriate with the approved changes.
☐ Engineering and other affect project groups implement change.

Modification of Requirements (initiated by Product/project team)

☐ Product Manager submits change to the CCB for analysis of feasibility and impact on the project.
☐ If change is approved by the CCB, the Product Manager communicates the request to the Client/User for approval.
☐ If the Client/User approves the change, the Configuration Manager updates the requirements documents as appropriate with the approved changes.
☐ Engineering and other affect project groups implement change.

Reconciling Requirements with Final Product

☐ Testing group uses the Traceability matrices to verify that all approved requirements have been implemented.
☐ Testing group documents conformity of product/release deliverables with the requirements. This document should be presented to the Client/User with the final deliverable.

1.8 Requirements Management - Checklist of Process Issues

Requirements Management Checklist		
	Process Issue	**Checklist**
1	Is there a process in place to define, update, manage, and maintain requirements?	☐ Who is responsible? ☐ Who ensures that requirements management activities are performed throughout the project? ☐ Is there a standard Requirements Management Tool licensed and available to the projects as needed? ☐
2	Are mechanisms are in place to communicate requirements and requirements management activities to management, project owner, client/users, project team?	☐ Who is responsible?
3	How are requirement statements uniquely identified and named?	☐ Does each requirement have a unique ID#? ☐ Is each requirement statement accurately specified?
4	What format is used to document requirements?	☐ Is there a standard format for requirements documents? ☐ Is there a standard numbering scheme?
5	Who is responsible for the allocation of requirements to software?	☐ How are system requirements allocated to software? ☐ Who is responsible for documenting the allocated requirements?
6	What analysis techniques are used to evaluate and analyze requirements?	☐ How are the analysis results documented, approved, and communicated?
7	What is the approval process for validated requirements?	☐ Who must approve?
8	What is the process for changing requirements documents?	☐ Who can make changes to the requirements? ☐ Who is responsible for assessing the

Requirements Management Checklist		
	Process Issue	**Checklist**
		impact and cost of change to the schedule, product, etc.?
9	Where are requirements documents stored?	☐ Is there a managed and controlled repository?
10	Is Traceability implemented? How is traceability performed?	
11	What software development artifacts are produced for each requirement (analysis and design specifications, code, test plans and procedures, test results)?	☐ Is traceability implemented from each of these artifacts to the original requirements?
12	When are requirements baselined?	☐ What controls are in place to baseline requirements documents? ☐ Who is responsible for baselining requirements? Who controls the baseline?
13	Is there training available to project team members on the Requirements Management process?	☐ Who performs the training? ☐ How are training costs allocated?
14	How are requirements management activities funded within an organization or project?	☐ Who ensures that resources are allocated to each project to implement requirements management? ☐ Are these resources included in the project plan and cost estimates?
15	What is the process that ensures that requirements drive project activities, costs, and schedule?	☐ How is requirements management integrated into the product design and development lifecycle?
16	What is the process for monitoring and tracking the status of allocated requirements?	
17	How are all changes associated with a single requirement summarized, reviewed, and tracked?	☐ Who is responsible for tracking the number of changes to each requirement and the status of each change request?

1.9 Requirements Management References

- ☐ CMM Practices – CMU/SEI-93-TR-25
- ☐ ANSI/IEEE Std 1058.1 – 1987 (standard for software project management)
- ☐ ISO-STD-12207 – Software Development and Documentation
- ☐ MIL-STD-1521B (technical reviews and audits for systems)
- ☐ IEEE Standard 1028-1988 (software reviews and audits)

2 Capturing Requirements

Elicit requirements → Analyze requirements → Specify/document requirements → Validate (confirm accuracy and completeness) requirements

1. Develop a list of Functions
   - ☐ Identify what the product is supposed to accomplish
   - ☐ Express Functions as verbs that represent actions
2. Classify each Function in the list (Evident, Hidden, Frill)[2]. Evident – function is clearly visible to the user; Hidden – function is a background or behind-the-scenes process that is not necessarily visible to the user; Frill – function that the user is unwilling to pay for either in terms of $'s or not being willing to sacrifice another function deemed more important)
   - ☐ Create a list of Evident and Hidden Functions
   - ☐ Create a separate list containing only the Frill Functions (designers may only provide these functions if there is no cost or effort)
3. Based on the classification exercise brainstorm other Functions that are Hidden and add them to the Evident and Hidden Function list.
4. Identify Attributes (e.g., response time) – Attributes are characteristics of functions desired by the client/user
5. Identify Attribute Details (e.g., consistent, fast, slow) for each Attribute.
   - ☐ Attribute Details are the adjectives or adverbs
   - ☐ Response Time = (consistent, fast, slow)
6. Assign Attributes to Functions. An Attribute is a modifier to one or more Functions.
   - ☐ Some attributes may exist for all functions, a sub-set of functions, or for a single function.
   - ☐ There should be a common set of Usability Attributes that are assigned to each Function (see list below).
7. Prioritize Attributes (Must have, Want to have, Ignore)[1]
8. Drop Attributes prioritized as "Ignore". Document the "Must have" and "Want to have" attributes with the associated Functions.
9. Identify User Acceptance Criteria for all "Must have" Attributes (sometimes referred to as Constraints).
   - ☐ The system must satisfy the User Acceptance Criteria (Constraint) in order for the design solution to be acceptable.
   - ☐ User Acceptance Criteria must be measurable, concise, and complete.
   - ☐ The User Acceptance Criteria defines the *Solution Space* for a Function(s).

[2] *Exploring Requirements: Quality Before Design,* Donald C. Gause and Gerald M. Weinberg, 1989, Dorset House Publishing

10. Review all User Acceptance Criteria (Constraints) to ensure that they are not "over constrained" and thus unnecessarily restrict the *Solution Space*. The goal is to widen the *Solution Space* to a point acceptable to the client.
11. Develop a list of Preferences - Preferences are a desirable but optional condition placed on an attribute.
    - ☐ Whereas there may be multiple design solutions within the Solution Space, Preferences enable the designer to select the solution that is most acceptable to the user. Preferences guide the designer through the Solution Space.
    - ☐ The client, not the designer determines which Constraints are Preferences. Cost may influence their decision.
    - ☐ Go back and review the list of Constraints to determine if any may actually be Preferences ("what's it worth?" exercises will help distinguish between Constraints and Preferences)
    - ☐ Ensure that each Preference is measurable.
12. Develop a list of client Expectations – Expectations help to highlight unstated client expectations about items not in the requirements documents.
    - ☐ Revise the list to generalize, restate, and clarify the expectations.
13. Prioritize (limit) the Expectations (Implement now, Defer to later, Absolute impossibility).
    - ☐ Provide a rationale/reason/source for how each Expectation was prioritized to facilitate changes and disagreements later.
    - ☐ Whenever user satisfaction is questioned, go back and review the documented list of Expectations.
14. Measure, test, and review all requirements

2.1 Example Usability Attributes

- ☐ Colorful
- ☐ Ease of learning (learnability)
- ☐ Ease of relearning
- ☐ Easy to Find and Select information and tasks
- ☐ Easy to Install
- ☐ Easy to Maintain
- ☐ Easy to Perform tasks
- ☐ Easy to Understand information and tasks
- ☐ Easy to Use
- ☐ Efficiency
- ☐ Entertaining
- ☐ Flexible/versatile
- ☐ Help available
- ☐ Intuitiveness
- ☐ Long term ease of use
- ☐ Reliable
- ☐ Response time
- ☐ Self-diagnosing
- ☐ Smart design
- ☐ Usefulness
- ☐ User customizable
- ☐ User error rate
- ☐ User Friendly
- ☐ Well documented

3 Evaluating Requirements - Requirements must be..

1. **Complete** – requirements represent everything the product/application must accomplish (include all valid and invalid cases).
   - ☐ All inputs/triggers/conditions that initiate the process and all factors that influence the process are well documented.
   - ☐ Requirements include a definition of all product/application results/responses.
   - ☐ The boundary conditions including limitations, constraints, and modes are specified and well understood.
   - ☐ All data objects and action verbs are clearly defined and understood.
   - ☐ The Requirements Document is properly formatted (pages, tables, and figures are numbered/named/referenced, a glossary or data dictionary of terms and units of measure is provided).
   - ☐ All references are included and all sections are complete – no sections marked TBD.
2. **Correct** – all requirements represent the product/application to be built
3. **Nonambiguous** – requirements do not contain ambiguous words (diagrams may help eliminate ambiguity); terms with multiple meanings are defined in the glossary.
4. **Understandable by the user/client** – requirements are logically organized, worded in the client's domain, terms are defined, and figures/graphics are used to clarify.
5. **Consistent** – requirements, terminology, data flows, and control flows must not conflict
6. **Verifiable** – for each requirement there exists a method (test case and procedure) to observe/demonstrate/measure that the requirement is satisfied by the product/application
7. **Traceable** – the origin of each requirement (behavior and function) must be clear and linked to business/usability objectives, high-level business functions, or user needs.
   - ☐ Traceability must identify the impact of change on the product/application.
   - ☐ Traceability also helps maintain the original project scope.
8. **Annotated** – each requirement should contain a rationale, priority, and an indication of the likelihood of change in the future. A Notes section may be used to annotate requirements.
9. **Modifiable**

4 Techniques for wording and specifying requirements

1. When an artifact or term is explicitly defined, use that definition in the requirement statement rather than the term or artifact.
2. When a structure is described in words, draw a picture of the structure. Create several sketches that emphasize different aspects of the requirement.
3. Express calculations, equations, and formulas both in words and mathematically. Show several examples.
4. Verify (request proof) statements that imply Certainty.
   - ☐ Look for words such as always, never, every, none, all.
   - ☐ Request user artifacts.
5. Verify and clarify Persuasive terms such as obviously, clearly, certainly.
6. Clarify and quantify Vague terms such as some, often, usually, most.
7. Ensure that lists are complete.
   - ☐ Watch for terms: “such as”, “etc.”
   - ☐ Lists should contain examples.
8. Clarify Vague Verbs such as eliminated, rejected, processed, and handled. Identify the processing associated with each verb.
9. Further define statements in Passive voice. Rewrite the requirement in Active voice and identify the actor.
10. Clarify Comparative terms without references such as new, special, and earliest.
11. Clarify Pronouns such as he, she, it, they, our, their, and your. Specify to whom the pronoun refers.

5 Requirements Template

Requirement Template	
ID#	Unique ID#
Name	
Date created	Date requirement documented
Originator	Source of requirement
Type	Functional, non-functional, non-technical
Category	User Need, Preliminary, Requirement, Baselined
Classification	Evident, Hidden, Frill
Priority (optional)	High, Medium, Low
Functional Area	Product, component
Statement	Concise statement of the capability, function, or condition
Description	Paragraph describing the Requirement statement
Rationale	Background on this requirement (why)
Attribute list w/Priority, User Acceptance Criteria	State each attribute, list the priority (Must have, Want to have, Ignore), define the User Acceptance Criteria
Business Rules constraining requirement	Specify all business rules, exceptions, conditions, etc. that constrain this requirement.
Required Inputs	Specify data and conditions that must be present.
Required Processing	Paragraph describing the processing required to satisfy the requirement.
Expected Outputs	Specify outputs, data, and conditions that are produced as a result.
Preferences list	List of Attributes that may be classified as Preferences
Expectations list w/Priority	List of client Expectations w/ the priority rating (Implement now, Defer to later, Absolute impossibility)
Related Requirements	List requirements that are related to this requirement
Associated Software Release	Requirement is associated with Software Release <##>
Associated Systems	Requirement is associated with <system names>
Traceability to Business Need	Reference traceability matrix or ID#

Traceability to User Need	Reference traceability matrix or ID#
Assigned to	Person or group responsible
Comments	

6 Traceability

Definition of Traceability: "A requirements document is traceable if the origin of each of its requirements is clear and if it facilitates the referencing of each requirement in future development or enhancement documentation."[3]

1. Assess completeness of the product/release specification and implementation

   ☐ Identify requirements that are not satisfied by any product/release deliverable

2. Identify features, functionality, and project activities that are out-of-scope

   ☐ Highlight deliverables and/or features that satisfy no requirement(s)

3. Assess the impact of changes and additions to requirements

Below are typical types of traceability tracked for a project. Traceability cross references may be bi-directional and many-to-many.

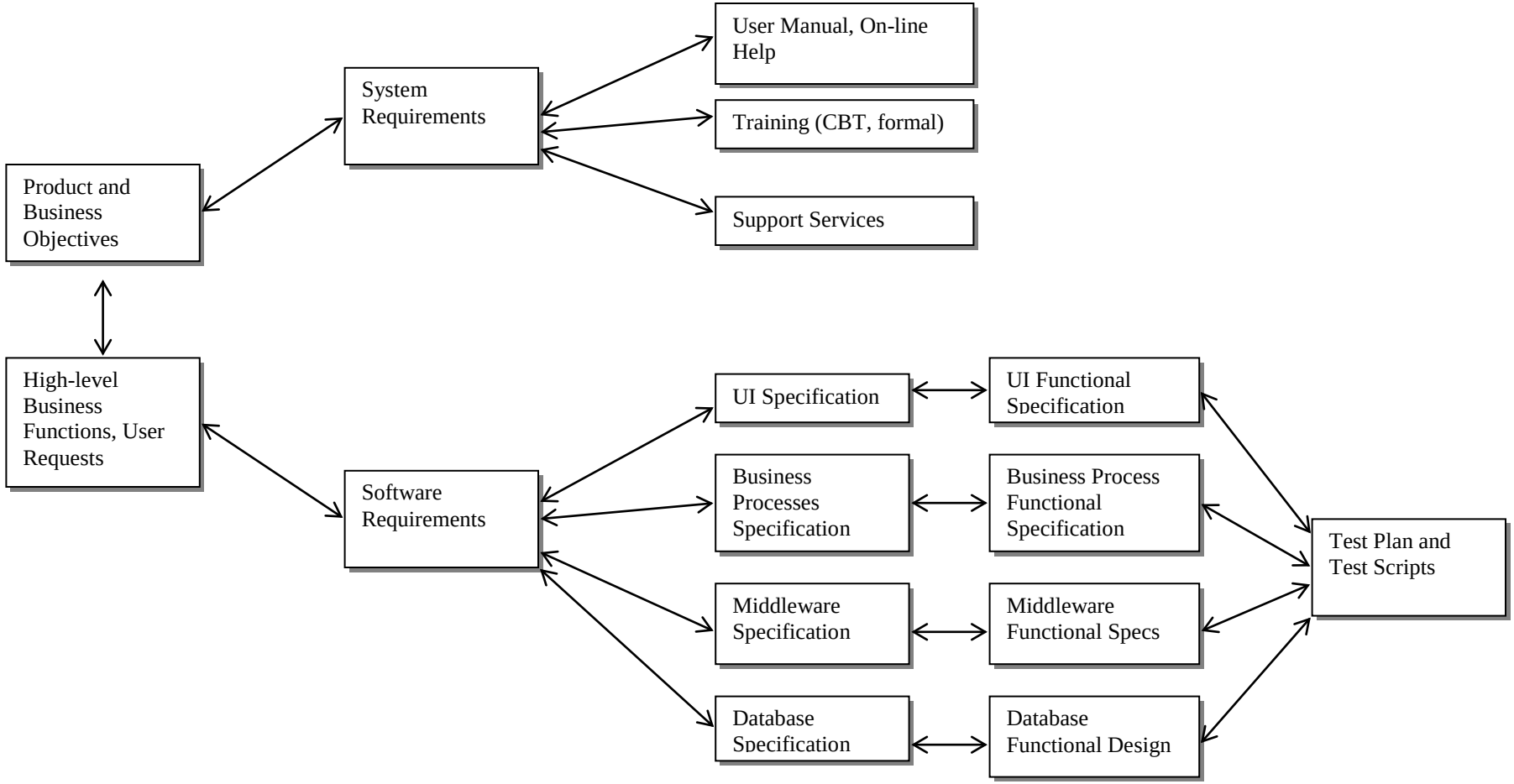
1. Forward from requirements
2. Backward to requirements

6.1 Requirements Products that may be traced:

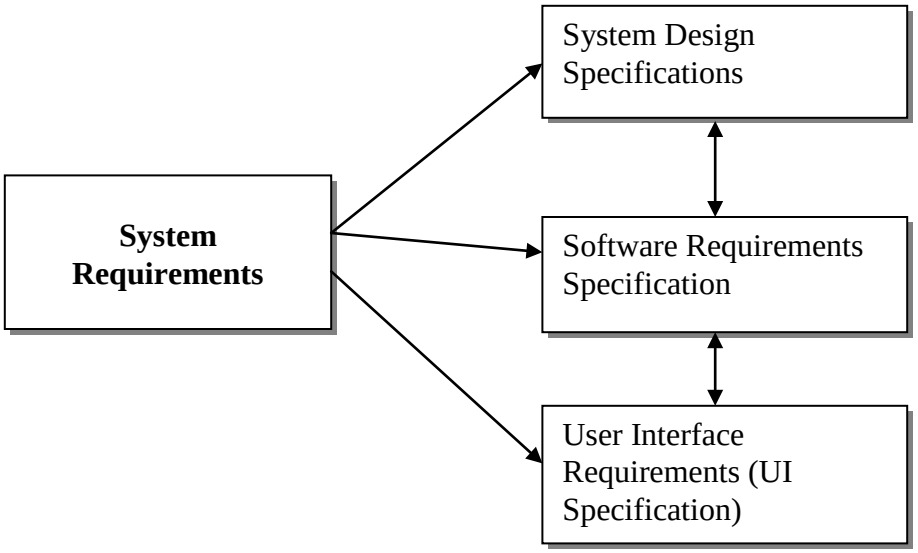
- ☐ Product/release requirements
  - ☐ Documentation Requirements
  - ☐ Training Requirements
- ☐ Software requirements
  - ☐ User Interface requirements
  - ☐ Usability Requirements
- ☐ User Interface Specification
- ☐ Detailed Functional Design
- ☐ Source Code
- ☐ Test Plans, scenarios, and procedures
- ☐ Software Development Plan – controlling document for managing a software product (technical and managerial processes, methods, tools, schedule, budget)

[3] IEEE Std. 830- 1984

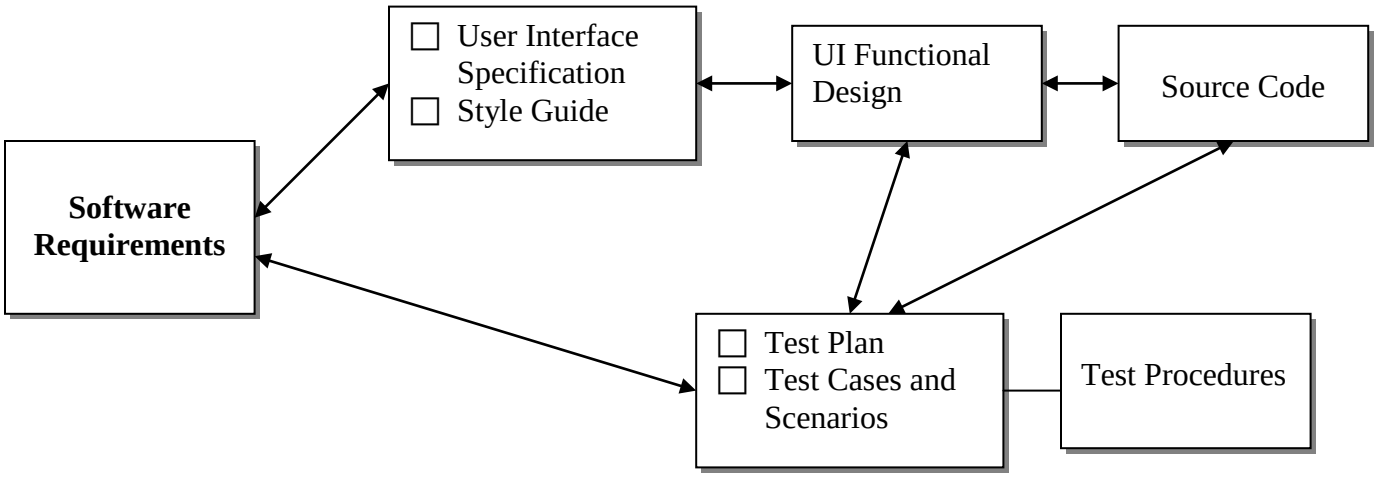
Recommended Traceability between major requirements documents:



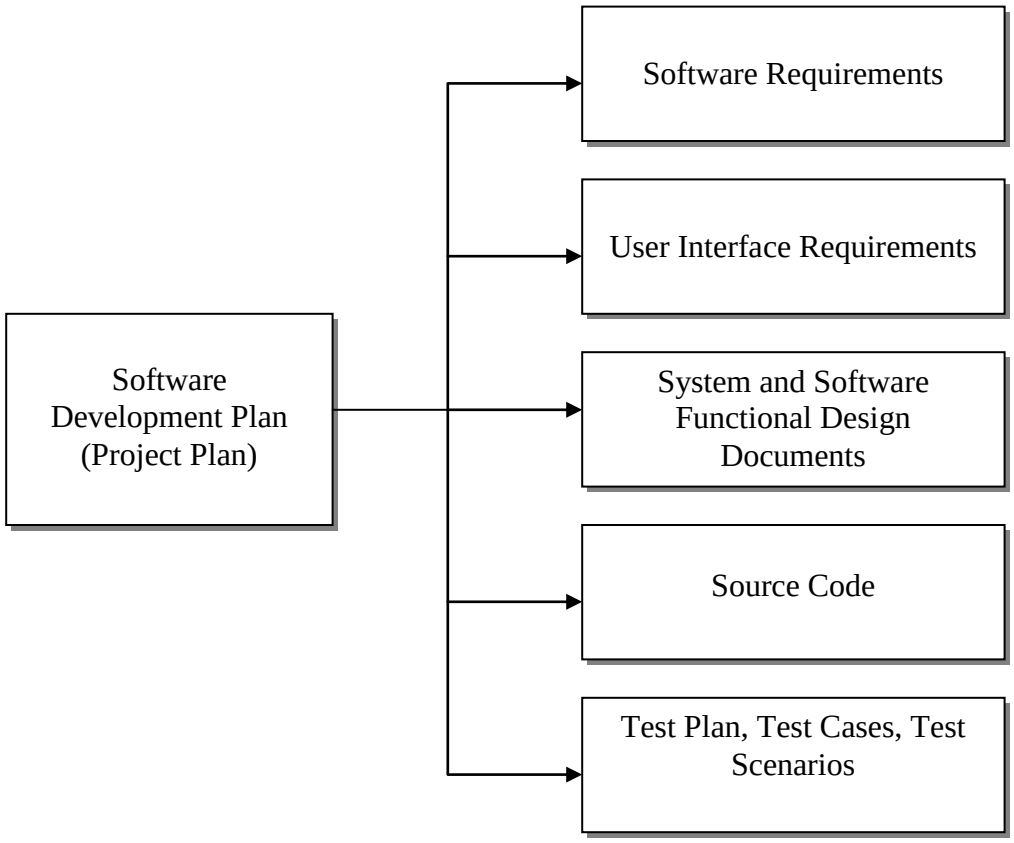
Traceability from the System Specification to System Design Document, Software Requirements, and the User Interface (Requirements) Specification



Traceability from Software Requirements to UI Design Documents, Source Code, and Test Plans



All Requirements Documents and Design Specification Documents must be consistent with the Software Development Plan (sub-set of the overall Project Plan):



6.2 Traceability Matrices

Table 1 : Requirements Traceability Matrix – Software Requirements ← → User Interface Specification

Software Requirements to UI Specification – Traceability Matrix				
Software Requirements ID#	**User Interface Specification ID#**			
	1.1.1.1	**1.1.1.2**	**1.1.1.3**	**1.1.1.4**
2.1.1.1-1		X		X
2.1.1.1-2			X	
2.1.1.2-1			X	
2.1.1.3-1			X	X
2.1.1.4.1				

Table 2: Business Objectives to Software Requirements Traceability Matrix

Business Objectives – Software Requirements Traceability Matrix			
Business Objective	**Software Requirement ID#**	**Software Requirement Name**	**Describe how the requirement supports the business objective**

Table 3: Business Objectives to User Needs Traceability Matrix

Business Objectives – User Needs Traceability Matrix			
Business Objective	**User Need ID#**	**User Need Name**	**Describe how the User Need supports the business objective**

Table 4: High-level Functions to Software Requirements Traceability Matrix

High-level Functions – Software Requirements Traceability Matrix			
High-level Function	**Requirement ID#**	**Requirement Name**	**Describe how the Requirement supports the High-level Function**