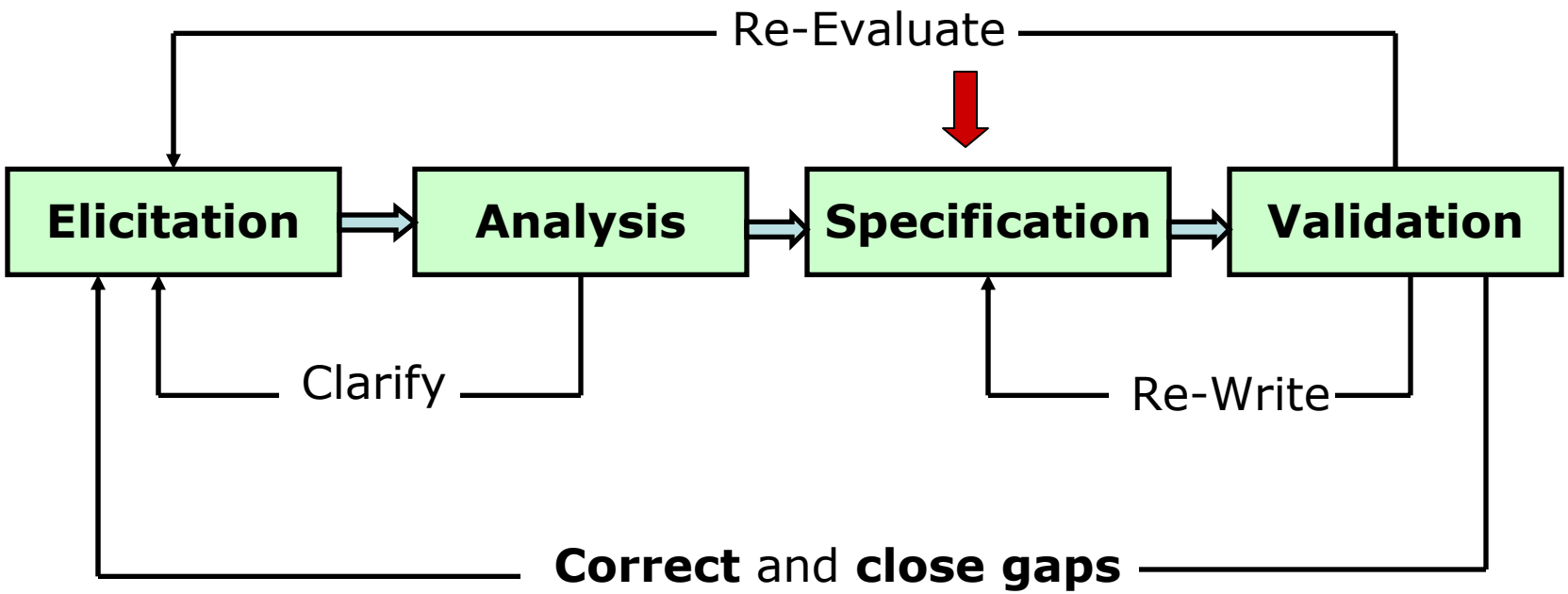
Requirements Engineering

Specification

John Vu
Senior Scientist
Institute for Software Research
Carnegie Mellon University

Requirements Development Process



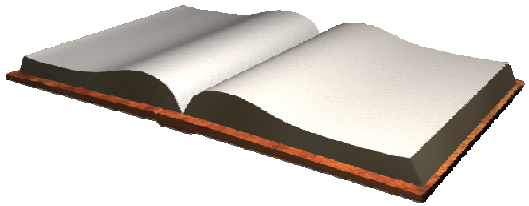
Iterative process - Multiple-steps, Not sequential

Document Requirements

- Requirements engineering is a **communication intensive** activity.
- Communication should start as early as possible since stakeholders may have different ideas of exactly what "requirements" are.
- The most essential practice is to write down the requirements in a structured format as you gather and analyze them.
- The objective of requirements development is to communicate a shared understanding of the new product among all project stakeholders.
- This understanding is typically captured in the form of a textual requirements specification document, written in natural language and augmented by appropriate analysis models.

Requirements Specification

- Requirements specification is the process of elaborating, refining, and organizing requirements **into documentation**.
- The development of a document that clearly and precisely records each of the requirements of the system or product.



* IEEE Standard 610.12 “IEEE Glossary of Software Engineering Terminology”

Requirements Specification

"A specification can be a written document, a graphical model, a formal mathematical model, a collection of usage scenarios, a prototype, or a combination of these."

Roger Pressman

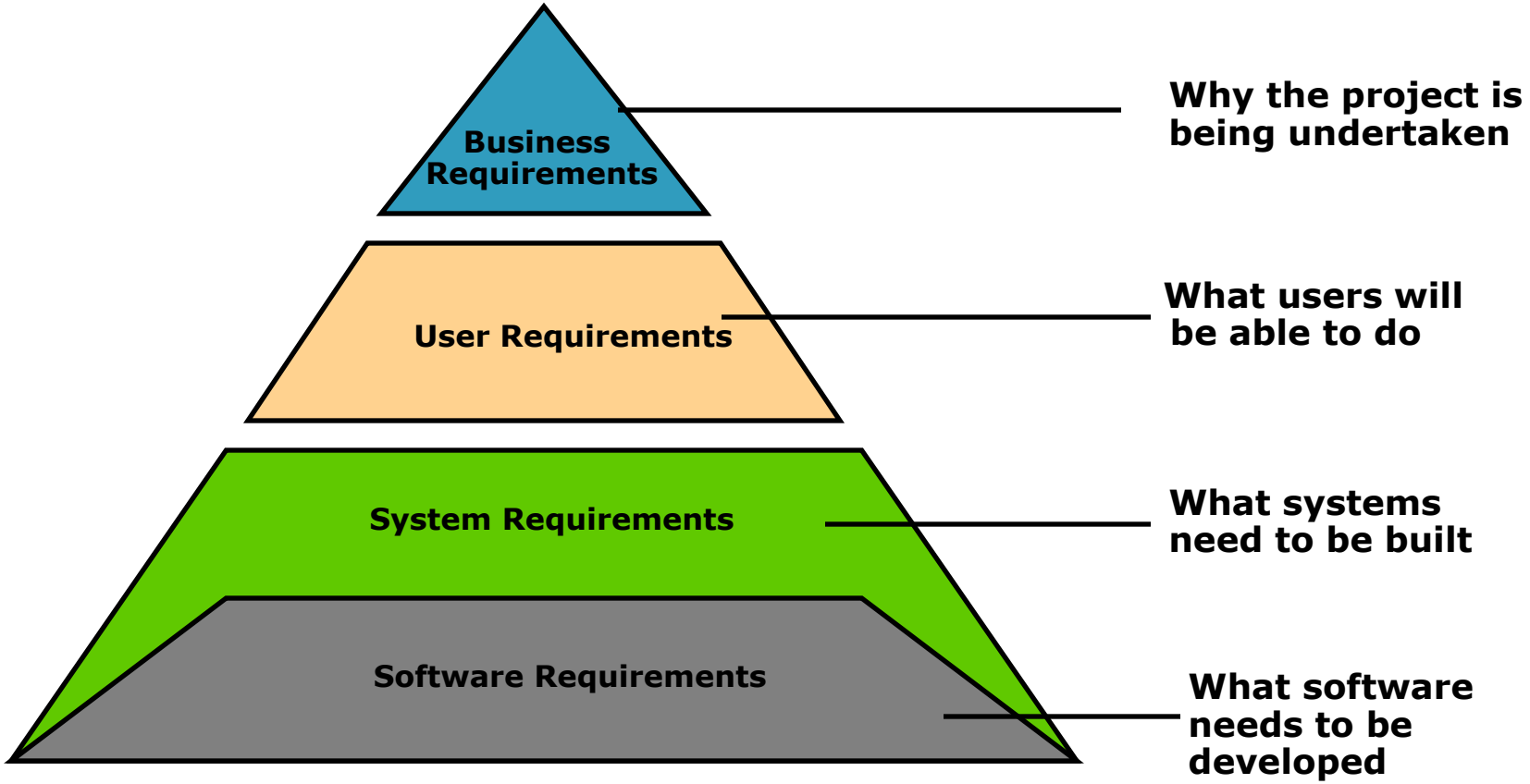
"A requirements specification describes the functions and performance of a computer-based system and the constraints that will govern its development."

Alan Davis

Requirements Specification Is

- A baseline agreement between stakeholders and project teams.
- A description of system behavior under various conditions.
- A document describing the characteristics and behavior of the proposed system from the user's point of view.
- The basis for subsequent planning, design and coding.
- The foundation for system testing, and user documentation.
- A bridge between user needs and the software requirements specification that the project team will use to develop the system.

Where Do Requirements Come From?



Basic Types of Documents

- Business Requirements Document (BRD)
- User's Requirements Document (URD)
- System Requirements Document (SRD)
- Functional Requirements Document (FRD)
- Architecture Design Document (ADD)
- Hardware Requirements Document (HRD)
- Software Requirements Specification (SRS)
- Interfaces Definition Document (IDD)

Documentation

WHY	WHAT	HOW WELL	HOW
Vision Goals Objectives System Goals *Business Requirements*	System Goals *User Requirements* *System Requirements* *Functional Requirements*	User Requirements System Performance System Constraints *Non-Functional Requirements (Quality attributes)*	System Architecture System Constraints *Software Requirements*

Business Requirements (WHY)

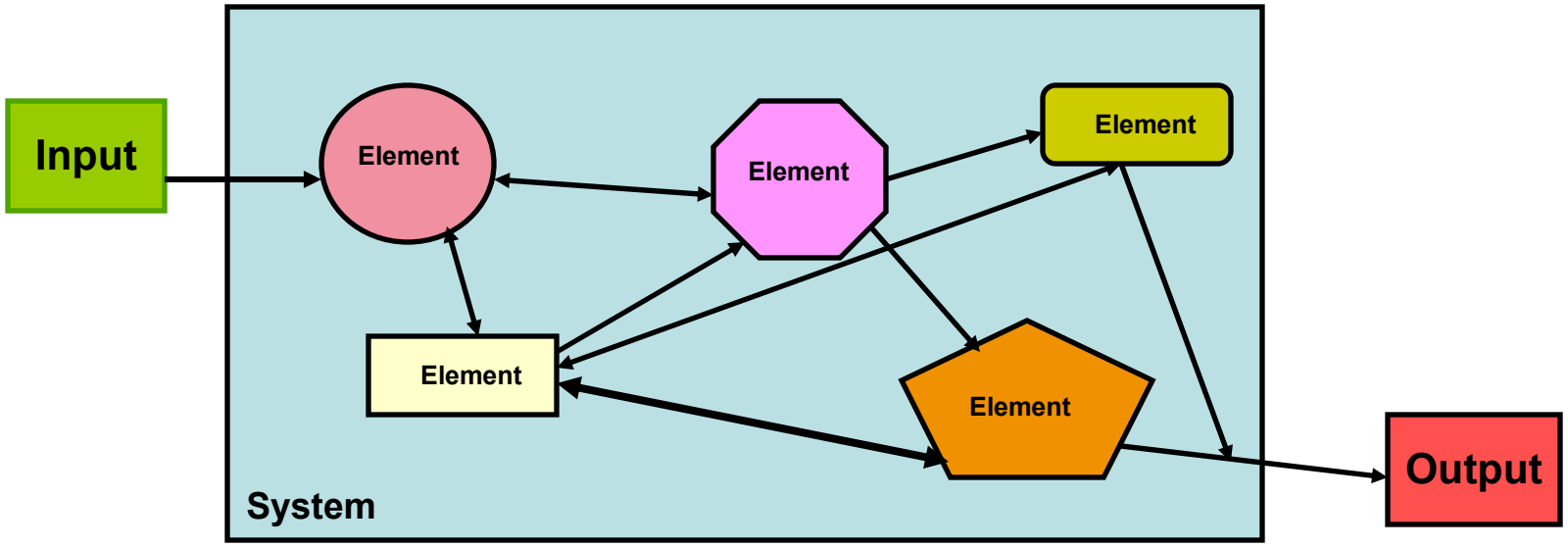
- High level objectives of the organization or customer requests for the system, product, or changes to current system.
- Reason are we doing this.
- Vision, mission, goals and objectives.
- Expectations and needs.
- Business analysis.
- Business case.
- Proposal, charters, roles and responsibilities.
- Sponsors and stakeholders.

System Requirements (What)

- Defines constraints from different stakeholders and environments.
- Owned by Software Engineers who work as requirements engineers, but must be reviewed by stakeholders.
- Stakeholders are responsible for prioritizing system requirements.
- Software Engineers are responsible for other attributes (i.e., Cost, efforts …).
- System requirements can be broken down into hardware requirements and software requirements.

System Definition

A system is a collection of elements related in a way that allows the accomplishment of some common objectives.



What Makes Up A System?

Components of System Requirements:

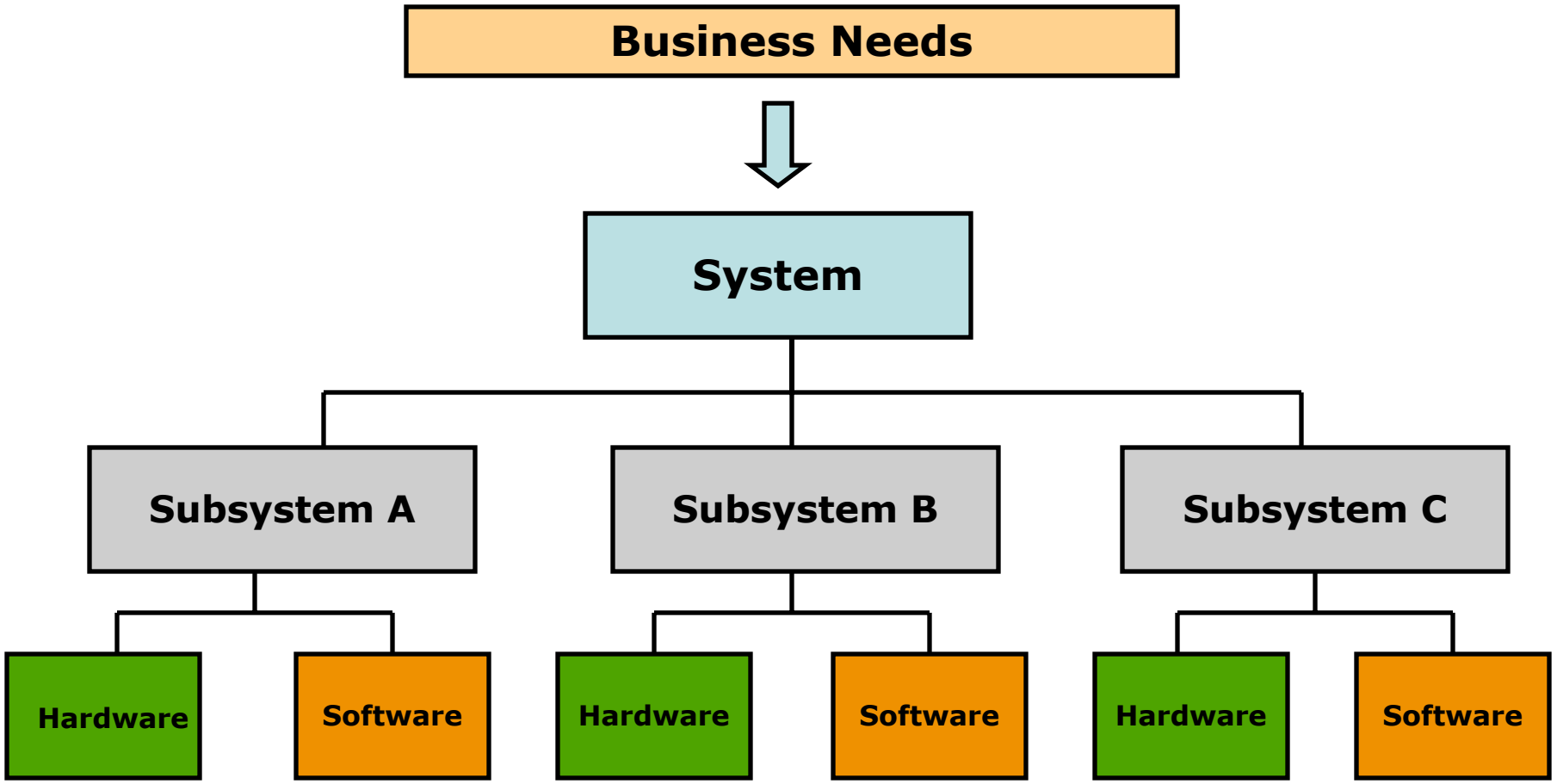
Key requirements here

The Core of the system requirements here

1. Descriptive Elements
2. Functional Elements
3. Performance
4. Interfaces
5. Non-Functional Requirements
6. Traceability to Stakeholders' inputs

Seldom mentioned by Stakeholder Need to explore further in order to "make sense"

System View


Business Needs
System
Subsystem A
Subsystem B
Subsystem C
Hardware
Software
Hardware
Software
Hardware
Software

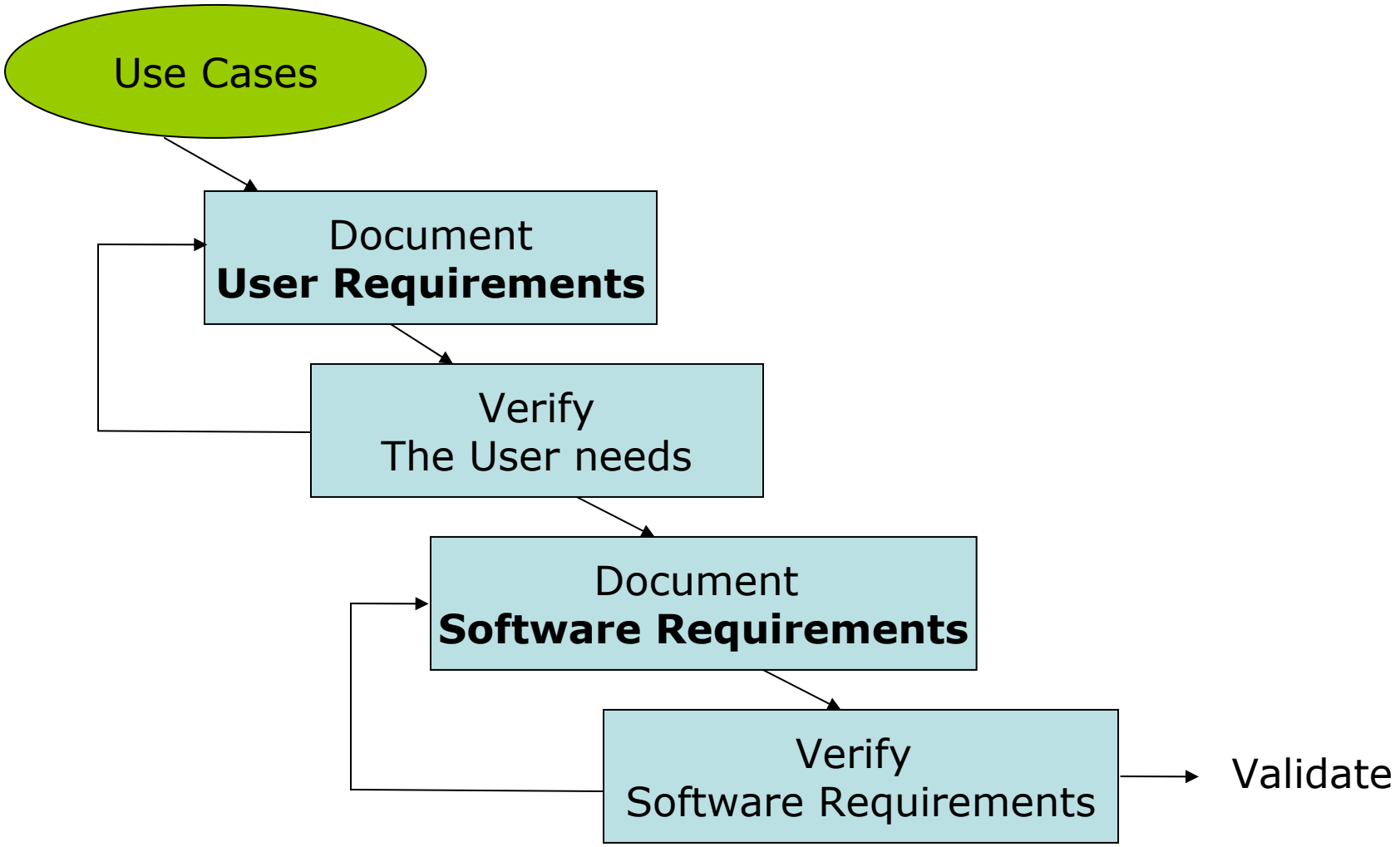
System Requirements Document

- Defines a model of the system to be built – Not the system.
- Defines some mixture of functionality, behavior, performance, and constraints.
- It is **only a model** – Not a complete system definition – Not yet.
- Organized by a logical lay-out to facilitate communication.
- Do not get bogged down on detail implementation.
- Every statement must be verifiable and traced back to stakeholders' requirements.

User Requirements (What)

- Definition:
  - A User requirements document is a record of requirements **written for users** that describes what users can do and why they need it.
- Purpose:
  - To obtain agreement on what the product must do to satisfy user requirements, to consolidate the needs of multiple users.
  - To act as a bridge between defining business needs and software requirements.

Specification Process Flow



Key Concept

When you need to	Then create
Document and verify User requirements	A User Requirements Document (URD)
Document and verify Software Requirements	A Software Requirements Specification (SRS)

Note:
A User requirements document is a record of requirements written for users that describes what users need and why they need it.

A Software requirements document is a record of requirements written for developers and allows them to design, develop and test the software solution.

Why User Requirements?

- To show understanding of stakeholders' requirements and describes the functions and performance of a system, as well as the constraints that will govern its development.
- Why don't we implement software **directly** from user requirements?
  - Distinguish desires, wishes, wants, and needs.
  - Understand different viewpoints of stakeholders.
  - Understand what the customer will pay for?
  - Analyze and organize requirements for verification with stakeholders and validation by testers.

What User Requirements Do?

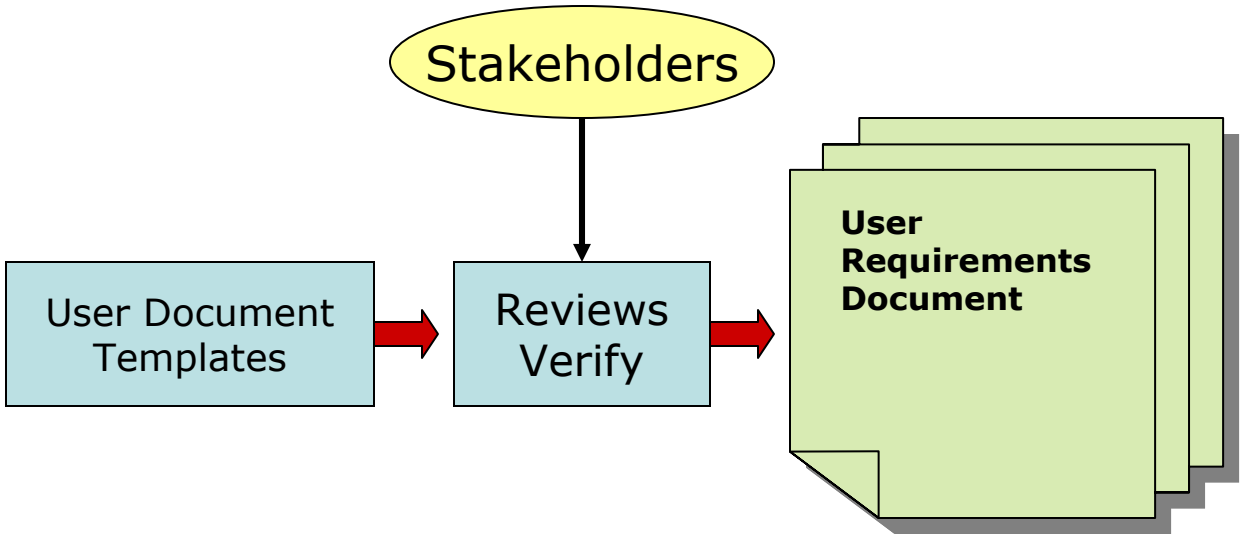
- Formalizes requirements elicited from all users and customers.
- Presents requirements at a high level without too many details.
- Provides background information, target environment, functional needs and quality attributes.
- Obtains agreement on what the product must do to satisfy user requirements.
- Consolidates the needs of multiple users.
- Provides a basis for deriving software requirement specification details.

Purposes

- User Requirements Document is used to:
  - Verify user requirements
  - Create preliminary test cases
  - Define some measurements
  - Negotiate contracts
  - Establish baseline for the project
  - Convey user's needs
  - State constraints
  - Identify quality attributes
- Reminder: Verify that user requirements describe what users need to do with the system to ensure that they are derived from business requirements, i.e., align with business goals, product vision etc.). Also have stakeholders review that it is complete, consistent and of high quality, then revise the document as needed.

User Requirements Process

Product Vision
Project Charter
Interview Notes
Analysis Models
ConOps
Business Rules
Standards
Process
Procedures
Use Cases
Context Diagram



Note: Be sure the language in the document is clear, concise and includes the users' terminology without technical language for ease to understand and verify.

When Writing Requirements - 1

- Write complete sentences with proper grammar and spelling.
- Keep sentences and paragraphs short and direct.
- Use active voice <***the system shall do X***>.
- Use terms consistently and as defined in the glossary.
- Decompose vague top-level requirements into sufficient detail so they are clear and unambiguous.

When Writing Requirements -2

- Use <**shall**> or <**must**> but avoid <should>, <may> or <might>.
- Identify specific actors whenever possible <**the administrator shall do** …>.
- Use lists, tables and graphs to display things visually.
- Emphasize the most important pieces of information (graphics, color, shading, etc.).
- Avoid vague and subjective terms.

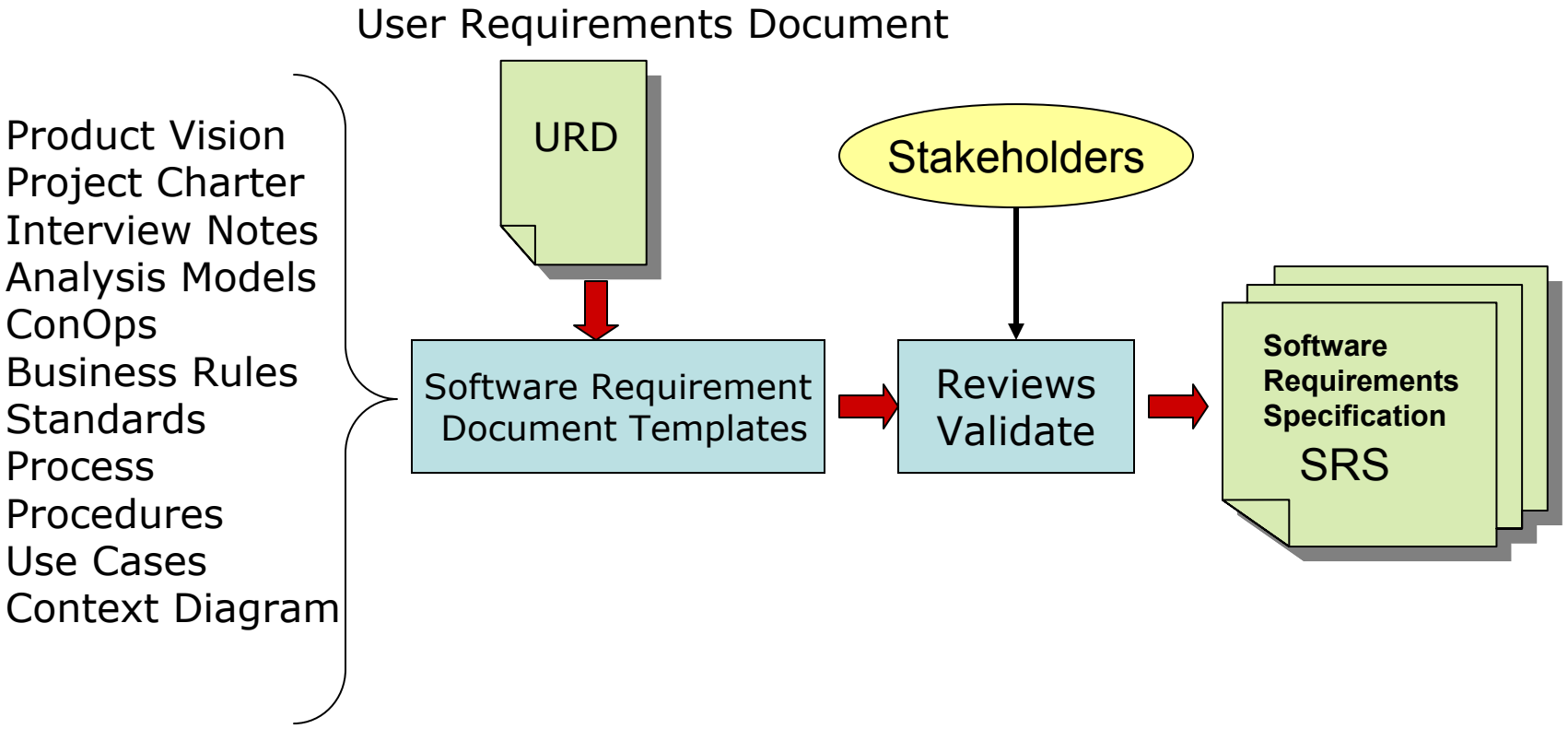
Software Requirements Specification

- The software requirements specification (SRS) is a precise record of requirements that enables software developers to design, develop, and test the software solution.
- The specification contains the entire set of prioritized functional and non-functional (quality attributes) requirements that the software product must satisfy.
- Its purpose is to document functional requirements, quality attributes, constraints and external interfaces for the software solution. It also serves as a contract between customers and software organizations.

What Software Specification Does?

- Transforms analysis models and user requirements information into precise software requirements statements.
- Incorporates analysis models directly into the software requirements specification.
- Provides the details necessary to design and code the product.
- Provides the basis for creating test plans and procedures.
- Serves as a source for creating training manuals, tutorials and documentation.
- Identifies mandatory requirements and the relative importance of the requirements.

Software Requirements Process



Note: Be sure the language in the document is clear, concise and groups all functions in a logical top down manner, organize sets of event-responses based on similar inputs and outputs.

Seven Rules For Specifications

1. Separate functionality from implementation: “What” not “How”.
2. Develop a model of desired behavior of a system that responds to stimuli from the known environment.
3. Establish the context in which software operates by specifying the manner in which other system components interact with the software.
4. Fully define the environment in which the software operates.
5. “Find any solution, then find the best solution.”
6. Ensure the specification is augmentable as system understanding improves.
7. Establish the content and structure of the spec and insure it is amenable to change.

Requirements Specification

- The key parts of a requirements specification are:
  - Definition of terms
  - Constraints on the project
  - Business logic
  - Limitations
  - Environment
- Note:
  - Be sure document conforms to the organization's templates, standards, and naming convention. Establish procedures for documentation change control to monitor any changes to document.
  - Decompose and document the functional requirements within each function.

How To Write Requirements?

- Write short, concise sentences using imperatives to describe the functional requirements.
- For example:
  - "The system shall prompt the user for the password."
  - The system shall display "Wrong password" when the input password does not match the stored password.
  - The system shall prompt user to enter another "password".

Standard Structure:

- [<Restriction>] <subject> <action verb> [<observable result>] [<qualifier>]
- Where:
  - [<restriction>] Identifies the conditions under which the requirement must be satisfied.
  - <subject> Shows who or what is doing the task (an actor).
  - <action verb> Is the task being performed.
  - [<observable result>] Shows the outcome of the action.
  - [<qualifier>] Identifies the quality attributes for the requirement.
- Note: The bracket [] indicate optional components of the sentence.

Examples

- No restriction:
  - "The system shall allow user to select an item in the menu display."
- Restriction:
  - "When password is wrong, the system will not allow user to access the system."
- Restriction, observable result and qualifier:
  - "When the password is correct, the system will allow user to select an item in the menu display within 5 seconds."

Writing Requirements - 1

- Use the active voice so that the subject is the performer of the action that is denoted by the verb.
  - "When user types in password correctly, the system shall allow user to access the system."
- Use phrase that follows an imperative and introduces a lower level requirement if needed.
  - "The system shall display several menus in the following order …"
- Always provide examples immediately following what is to be illustrated to make it clear and concise.
  - "For example …."

Writing Requirements - 2

- Decompose complex statement (sequence or flow) into multiple statements.
  - "The system shall display a menu that contains many choices for the user… from A to Z".
    - "The system shall display menu."
    - "The menu shall have many choices."
    - "The user shall choose between A to Z."
- Note: Careful about multiple requirements that have been aggregated into a single statement such as "and" and "or", as the requirement may suggest that several requirements have been combined. Never use "and/or".

Writing Requirements - 3

- Decompose compound statements (“something with “and”, “or” “also” and with”) into multiple statements.
  - “The system shall display the menu and allow user to select items.”
    - The system shall display the menu.
    - The system shall allow user to select items.

Writing Requirements - 4

- Breakdown exception clause (such as “Not”, “If”, “But”, “Unless”, “Although” and “Except”) into separate distinct requirements statements.
- Use tables or charts to explain complex requirements. Label each table or chart with unique identifiers for easy identification.
- Uniquely label each lower level requirement so that its hierarchical association to its higher level requirement is clear.
  - 1.0 Display Menu (Function name)
  - 1.1 Menu Item job (Use case name)
  - 1.1.1 The system shall allow user to select an item (Use case step)

Requirements Document Issues

- Inappropriate Level
- Poor Definition
- Inconsistent Terminology
- Lack of Structure and Control
- Undefined Ownership
- No Traceability
- Not Testable Requirement Statement
- In complete sentence – mostly “buzzwords”, list of acronyms, or “sound bites”
- No completion criteria

Good Requirements - 1

- Each requirement should be:
  - **Clear** Avoid confusion
  - **Brief** Short and simple
  - **Verifiable** Testable and verifiable
  - **Traceable** Uniquely identified, can be tracked
- Each requirement should be annotated with attributes:
  - **Priority** Emphasize important requirements
  - **Source** Who requires it?
  - **Urgency** When?
  - **Identity** Uniquely identifiable
  - **Comments** Clarification when needed
  - **Value** Benefit justification

Good Requirements - 2

- Each collection of requirements should be:
  - **Complete** Are all requirements expressed?
  - **Balanced** Are all requirements at a consistent level of detail?
  - **Modular** Can system be changed without unforeseen side effects?
  - **Correct** Are user needs correctly expressed?
  - **Consistent** No contradictions exist between requirements.
  - **Realistic** Can we really build it?
  - **Roles/State** Are requirements associated with user roles or system state?
  - **Type inclusive** Functional, Non-Functional, Constraints, guidelines.

Are Your Requirements Readable?

- Because requirements specification documents have a diverse audience, follow these guidelines to make them understandable:
  - Label all sections, subsections and individual requirements consistently.
  - Use white space liberally.
  - Use visual emphasis consistently (**bold**, *italics*, underline).
  - Create a table of contents and an index.
  - Number all figures and tables and provide captions.
  - If possible, use Hyperlinks between sections.
  - Use an appropriate template.

Audiences

- **Users**:
  - Most interested in System or User requirements.
  - Not generally interested in Software requirements.
- **Systems Analysts, Requirements Engineers**:
  - Write various specifications that interrelate.
- **Developers, Programmers**:
  - Have to implement the requirements.
- **Testers**:
  - Determine that the requirements have been met.
- **Project Managers**:
  - Measure and control the analysis and development processes.

Rate This Requirement

Under ideal conditions, the system ought to perform in a manner that significantly satisfies the customer in a majority of favorable scenarios where its use is deemed as appropriate.

Rate This Requirement

"The system shall invoice customers for jobs at month-end."

What is an invoice? What information does it contain? Will the invoice amount ever be adjusted for any reason? Do all customers get an invoice? When is month-end? Do all jobs get invoiced or only certain ones? In what format is the invoice provided?

Answering the above questions can result in additional documentation such as a glossary definition for an invoice, a statement naming the attributes of an invoice, a business rule for when invoices are adjusted, a restriction statement for the function requirements statement, or conditions for printing the invoice.

Key Concept In Specification

1. Unambiguous for intended readers:
   - Must be written from the reader's view.
   - Avoid unnecessary repetition.
   - Avoid ambiguity.
   - Follow a standard organization or template.
2. Clear enough to test. (Testable)
   - e.g., How to satisfy this requirement by test case or test script.
3. No Design mixed in with Requirements.
   - e.g., "What" not "How".

Preparation for Testing

- All specified requirements must be grouped into relevant hierarchical levels of requirements.
- Linkage to related requirements must be explicit, complete.
- Each requirement must be specified so that it is possible to define an unambiguous test, to prove that it is actually implemented.
- A unique test may be specified or outlined immediately but specific tests can be designed in detail later. The key idea is that all requirements must be clear enough to be testable by some means.

Questions To Ask

- Why isn't this requirement quantified?
- What is the degree of risk for this requirement?
- What kind of uncertainty is this requirement and why?
- Are you sure about this? If not, why not?
- Where did you get that requirement statement from?
- How can I check it out?
- How does this idea affect the organization’s business goals?

Questions to Ask

- Did we forget anything critical?
- How do you know it works that way? Did you do it before?
- Have we gotten a complete solution here?
- Are all system objectives satisfied?
- Who is responsible for the failure or success of this system?
- How can we be sure that this requirement specification is working?

Requirements Review

- Software Quality Assurance must review requirements specification to:
  - Identify any standard non-compliances.
  - Ensure it follows organizational templates and guidelines.
  - Ensure it is documented, well written, clear, and complete.
  - Ensure it can be used by 'intended readership'.
  - Check before baselined by configuration management.

Common Requirements Mistakes

- Mixing real requirements with ‘designs’.
- Failing to quantify competitive qualities.
- Failing to derive implied requirements from designs.
- Failing to set requirements at appropriate levels.
- Failing to include source information.
- Failing to include priority information.
- Failing to specify all critical stakeholder requirements.

Common Requirements Mistakes

- Failing to distinguish between wishes and their profit.
- Failing to consider all limited resources as requirements.
- Failure to update requirements evolutionarily.
- Failure to consider time, location and conditions with requirements.
- Failure to plan a time series of quality levels.

Questions & Answers