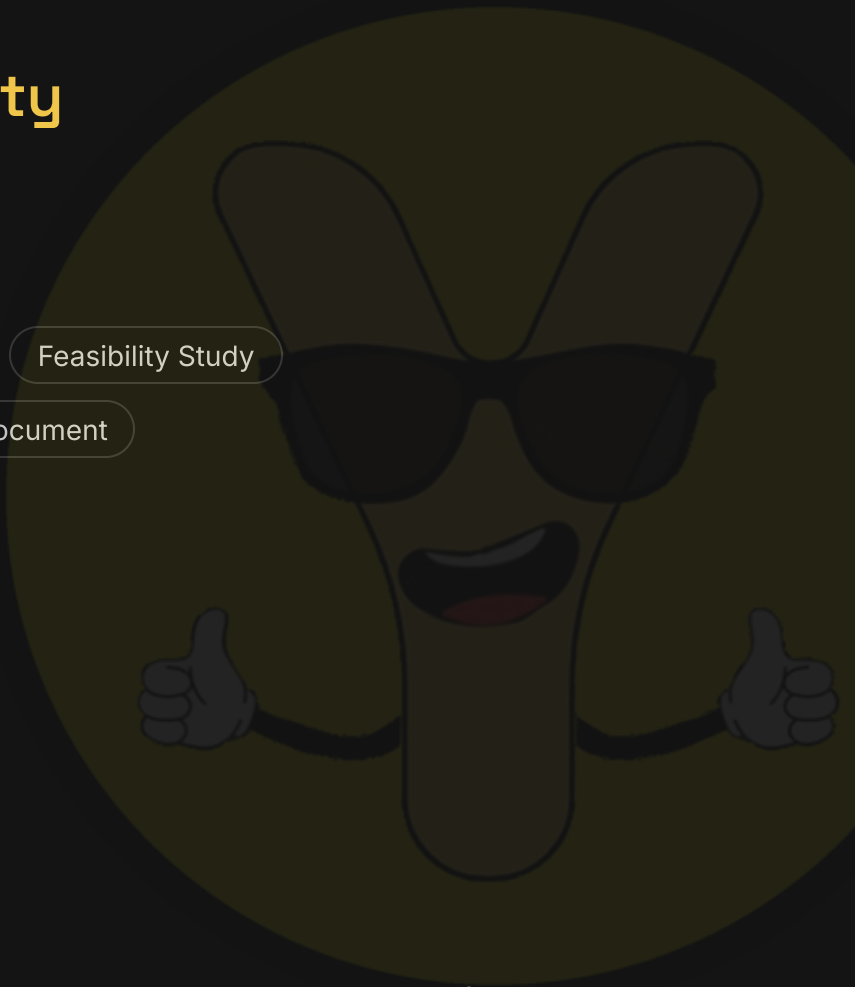
AKTU · 3RD YEAR · SEMESTER 6 · BCS601

Software Engineering

Unit 2 — Software Requirement Specifications & Quality Assurance

- Software Requirement Specifications (SRS)
- Requirement Engineering Process: Elicitation, Analysis, Documentation, Review and Management of User Needs
- Feasibility Study
- Information Modelling
- Data Flow Diagrams
- Entity Relationship Diagrams
- Decision Tables
- SRS Document
- IEEE Standards for SRS
- Software Quality Assurance (SQA)
- Verification and Validation
- SQA Plans



DECODED BY

Chirag Jain

Founder, Ycotes · Gold Medalist in Artificial Intelligence, IIT Madras · taught 10,000+ students

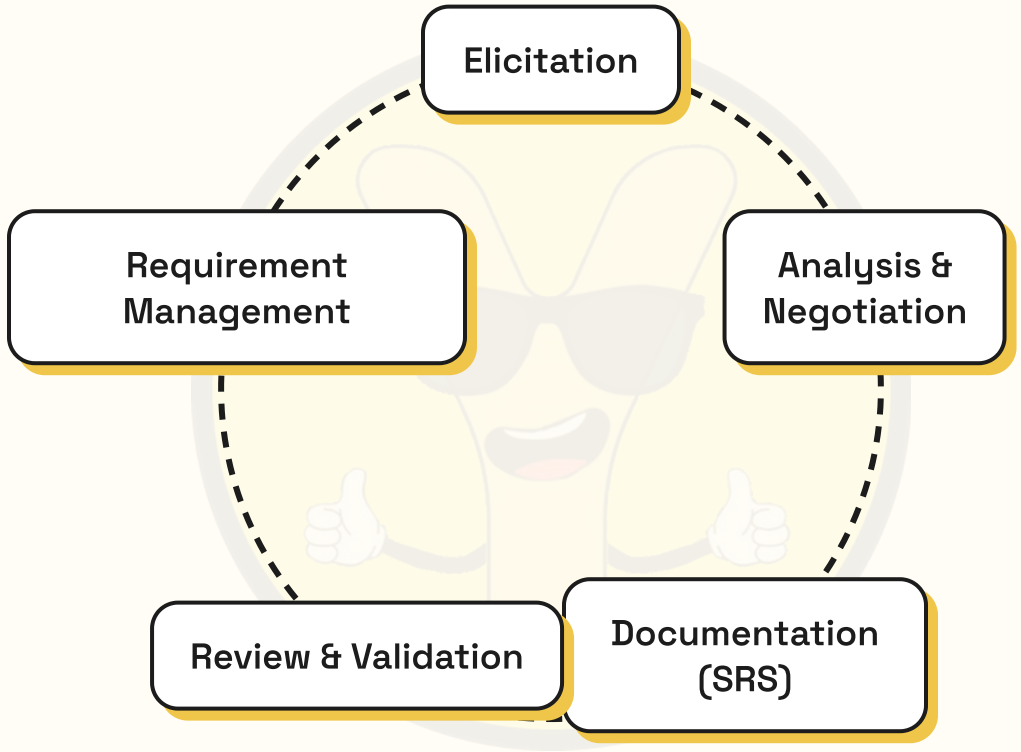
Software Requirement Specifications (SRS)

SOFTWARE REQUIREMENT SPECIFICATION (SRS)

An SRS is a comprehensive, formal document that describes the intended behavior, functional requirements, non-functional requirements, constraints, and external interfaces of a software system to be developed.

- Acts as a formal contract between system customers and software developers.
- Serves as the baseline for system design, implementation, testing, and project validation.
- Captures user needs (what system should do) and system constraints (how system must perform).

Requirement Engineering Process Flow



Iterative lifecycle of Requirement Engineering from initial discovery to ongoing management.

Requirement Elicitation and Analysis

Methods used to gather, analyze, and negotiate requirements from stakeholders.

- Elicitation Techniques
  - Interviews, surveys, domain analysis, and stakeholder brainstorming sessions.
  - Prototyping to clarify vague customer expectations and user interface needs.
  - Use Case modelling for capturing step-by-step user interaction scenarios.
- Requirement Analysis
  - Discovering conflicts, overlaps, and incomplete requirements across stakeholders.
  - Categorizing into functional features and non-functional quality attributes.
  - Negotiating trade-offs between project cost, timeline, and requested functionality.

Requirement Review and Management

Ensuring requirement correctness and controlling changes throughout development.

- Requirement Review
  - Formal technical reviews (FTR) to check completeness, consistency, and clarity.
  - Validation against stakeholder expectations before architectural design begins.
- Requirement Management
  - Requirements Traceability Matrix (RTM) maps requirements to design and test cases.
  - Change Control Board (CCB) evaluates, approves, or rejects requirement change requests.



Types of Feasibility Studies

Feasibility Type	Primary Focus	Key Evaluation Criteria
Technical	Technology & Tools	Does team possess required technical skills, hardware, and software tools?
Economic	Cost vs Benefits	Is project financially viable? Evaluates Return on Investment (ROI) and cost ratio.
Operational	User Acceptance	Will end-users accept and effectively operate the proposed system?
Schedule	Time Constraints	Can the software be delivered within strictly defined customer deadlines?
Legal & Ethical	Compliance & Law	Does software violate copyright laws, data privacy, or safety regulations?

Information Modelling Concepts

Abstract representation of data structures and relationships within a system domain.

- Focuses on data objects, attributes, and relationships rather than operational processing logic.
- Data Object: A representation of composite enterprise information (e.g., Student, Invoice).
- Attributes: Properties or characteristics defining a data object (e.g., Student_ID, Name, GPA).
- Relationships: Logical connections binding data objects together (e.g., Student Enrolls-In Course).
- Serves as the conceptual bridge between requirement analysis and relational database design.

Data Flow Diagram (DFD) Hierarchy



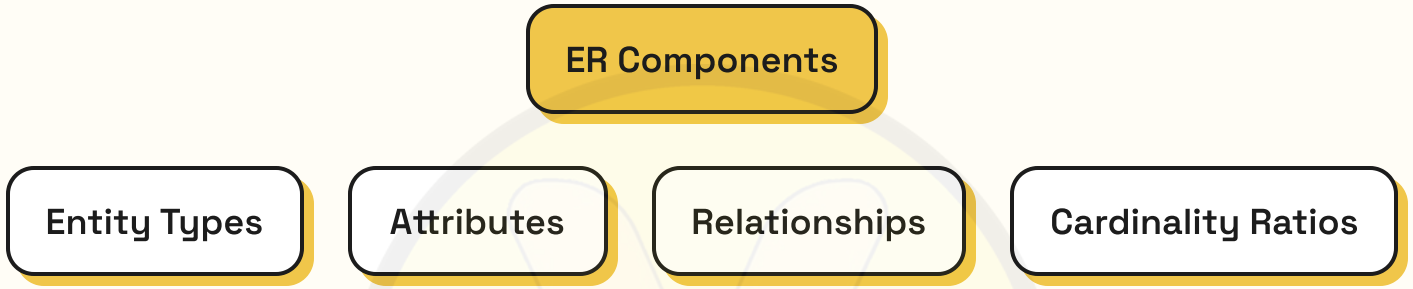
Hierarchical breakdown of DFDs showing increasing levels of system processing details.

DFD Notations and Design Rules

Key structural elements used to represent software data flows.

- DFD Notations (Gane-Sarson / Yourdon)
  - Process (Circle/Rounded Rect): Transforms incoming data into outgoing data.
  - Data Flow (Arrow): Represents movement of data packets across processes.
  - Data Store (Open Rect/Parallel Lines): Represents stored data at rest.
  - External Entity (Rectangle): Source or sink of external system data.
- Crucial DFD Rules
  - Data cannot flow directly between two external entities.
  - Data cannot flow directly between two data stores without a process.
  - Conservation of Data: All process outputs must be derived from process inputs.

Entity Relationship (ER) Diagram Components



Core structural components of Entity-Relationship conceptual data models.

Decision Table Structure and Worked Example

Rules & Conditions	Rule 1	Rule 2	Rule 3	Rule 4
Condition: Valid Student ID?	Y	Y	N	N
Condition: Fees Fully Paid?	Y	N	Y	N
Action: Issue Examination Admit Card	X	-	-	-
Action: Display Fee Dues Warning	-	X	-	-
Action: Reject Exam Registration	-	-	X	X

IEEE 830 Standard Structure for SRS

Standardized outline for software requirement specification documents.

- 1. Introduction: Purpose, Scope, Definitions, Acronyms, References, Overview.
- 2. Overall Description: Product perspective, Functions, User characteristics, Constraints.
- 3. Specific Requirements: Detailed functional, non-functional, and interface details.
- Functional Requirements: Exact input, processing, and output logic for features.
- Non-Functional Requirements: Reliability, performance, security, and maintainability metrics.
- External Interfaces: User, hardware, software, and communication interface specifications.

IEEE Standard Characteristics of a Good SRS

Qualities required for an SRS document to be effective and high quality.

- Correct: Every requirement stated represents an actual necessary system capability.
- Unambiguous: Every requirement has only one single, definitive interpretation.
- Complete: Includes all significant requirements, responses to inputs, and labels.
- Consistent: Requirements do not contradict other requirements or design constraints.
- Verifiable: Every requirement can be cost-effectively checked via test or audit.
- Traceable & Modifiable: Easily changeable with explicit backward and forward links.

Software Quality Assurance (SQA)

SOFTWARE QUALITY ASSURANCE (SQA)

SQA is a planned and systematic pattern of actions necessary to provide adequate confidence that a software product conforms to established technical requirements and quality standards.

- Focuses primarily on process quality to prevent defects during software development.
- Includes SQA planning, audits, technical reviews, testing strategies, and standards compliance.
- Defect Prevention Principle: Software Quality = Process Quality + Product Auditing.

Verification vs Validation (V&V)

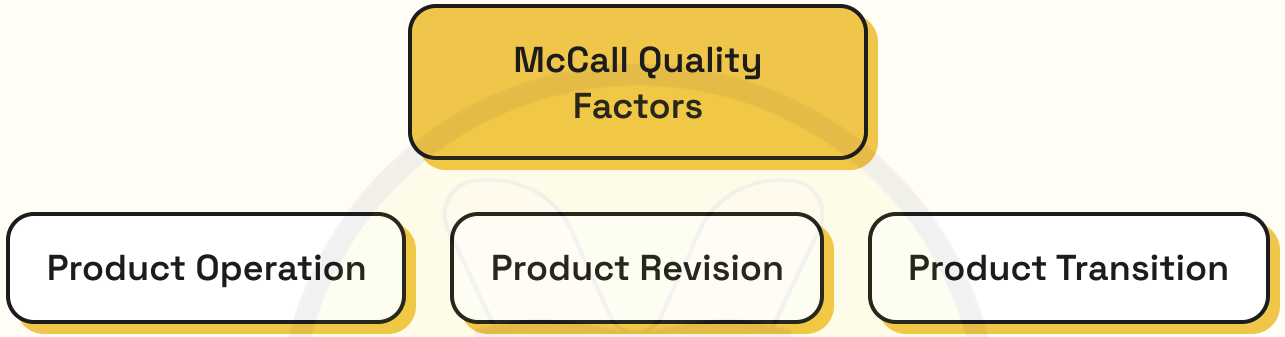
Parameter	Verification	Validation
Core Question	"Are we building the product right?"	"Are we building the right product?"
Focus	Process and intermediate artifacts (SRS, Code, Design)	Final executable software product
Techniques	Static testing: Inspections, Reviews, Walkthroughs	Dynamic testing: Unit, Integration, System Testing
Execution	Executed without running actual program code	Executed by running software test cases
Objective	Ensure compliance with input specifications	Ensure compliance with customer expectations

SQA Plan and Key Activities

Structured organization of quality activities throughout the development lifecycle.

- SQA Plan: Defines quality goals, roles, review schedules, and tools for a project.
- Formal Technical Reviews (FTR): Peer reviews of software artifacts to uncover defects early.
- Software Quality Metrics: Quantitative measures of defects, complexity, and test coverage.
- Quality Audits: Independent evaluations of project activities against organization standards.
- Software Configuration Management: Ensuring code and documentation versions remain stable.

McCall’s Software Quality Framework



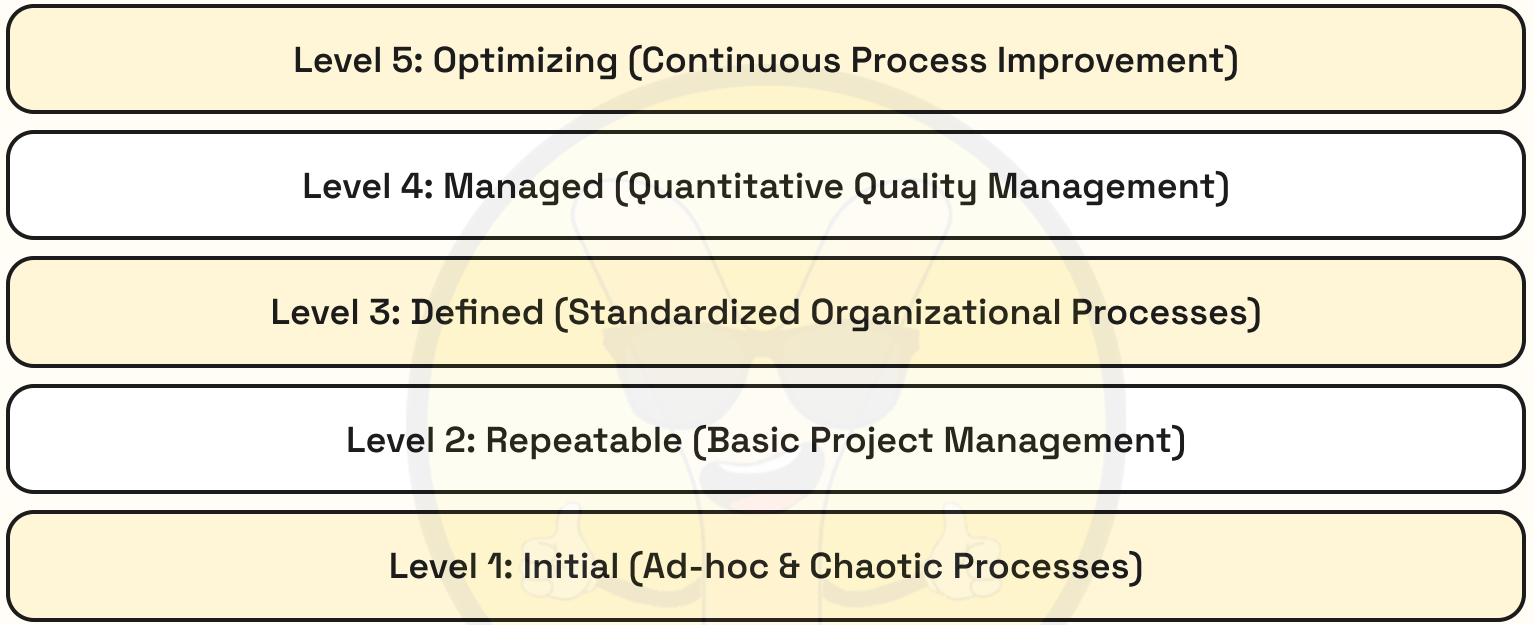
McCall's 3-pillar categorization of software quality characteristics.

ISO 9000 Models for Software Quality

International standard framework for organizational quality management systems.

- ISO 9001: General standard for quality management systems across design and development.
- ISO 9000-3: Specific guidelines for applying ISO 9001 to software development lifecycle.
- Process-Oriented Approach: Quality is achieved by defining and continuously improving workflows.
- Key Principles: Customer focus, leadership, process approach, continuous improvement.
- Certification: Requires third-party independent audits of organizational quality processes.

SEI Capability Maturity Model (CMM) Levels



The 5 maturity levels of SEI-CMM for evaluating software process maturity.

Exam Corner - University Questions

- **Q1** What is SRS? Explain the IEEE 830 standard format and characteristics of a good SRS document.
- **Q2** Differentiate between Functional and Non-Functional requirements with suitable examples.
- **Q3** Draw Level-0 (Context) and Level-1 DFD for an Online Library Management System.
- **Q4** Explain the Requirement Engineering process in detail with a neat block diagram.
- **Q5** Compare Verification and Validation. Explain Formal Technical Reviews (FTR) in SQA.
- **Q6** Explain the 5 levels of the SEI Capability Maturity Model (SEI-CMM) with Key Process Areas (KPAs).
- **Q7** What is a Decision Table? Construct a Decision Table for a University Student Admission system.

Unit 2 done. You are exam-ready.

Revise the Exam corner once more, then open Unit 3 on theycotes.com. Every AKTU subject, unit by unit, free. Notes by Chirag Jain, Ycotes.

