CS24353 • MODULE 2

Module 2: Software Requirements Engineering & SRS Standards

Topics 15 to 26 • Functional/Non-Functional Requirements, FURPS+, TELOS Feasibility, IEEE 830 SRS & RTM

Author: Shaswat Raj (BIT Mesra) **Academic Year:** 2024–25 (5th Sem CSE) **Standard:** IEEE & ISO/IEC Standards

Topic 7: The Requirements Engineering Lifecycle

Requirements Engineering (RE) is the systematic process of discovering, analyzing, documenting, and maintaining software requirements throughout the project lifecycle. The cost of fixing a requirements error escalates exponentially across subsequent phases:

$$\text{Relative Cost: Requirements } (1\text{x}) \rightarrow \text{Design } (5\text{x}) \rightarrow \text{Coding } (10\text{x}) \rightarrow \text{Testing } (20\text{x}) \rightarrow \text{Post-Release } (100\text{x–}200\text{x})!$$

The 5 Core Phases of Requirements Engineering

1. **Feasibility Study:** Assesses whether the proposed system is viable technically, economically, and operationally.
2. **Requirements Elicitation:** Gathers customer and user needs through interactive fact-finding techniques.
3. **Requirements Analysis & Modeling:** Resolves ambiguities, boundary conflicts, and overlaps using structured models.
4. **Requirements Specification:** Formally documents the functional and non-functional requirements in an **SRS**.
5. **Requirements Validation & Management:** Validates requirements with stakeholders and manages change requests via a **Requirements Traceability Matrix (RTM)**.

Topic 8: Functional vs. Non-Functional Requirements & The FURPS+ Framework

Category	Definition & Scope	Concrete Industry Examples
Functional Requirements (FR)	Specifies the exact behavioral services and computational functions the system must perform (*What the system does*).	• "The system shall calculate compound interest using formula $A = P(1 + r/n)^{nt}$." • "The system shall generate PDF invoices upon successful Stripe payment."
Non-Functional Requirements (NFR)	Specifies systemic quality attributes, performance bounds, security constraints, and operational standards (*How well it performs*).	• "The system shall maintain 99.99% monthly availability." • "API response latency shall not exceed 200ms at $10,000$ concurrent requests."

The FURPS+ Quality Attribute Taxonomy (Hewlett-Packard)

- **F — Functionality:** Feature set, capabilities, security encryption, mathematical algorithms.
- **U — Usability:** Human factors, aesthetics, user interface consistency, error recovery, accessibility (WCAG 2.1).
- **R — Reliability:** Failure frequency, MTTF, recoverability, predictability, fault tolerance.
- **P — Performance:** Response latency, throughput, memory consumption, resource limits.
- **S — Supportability:** Testability, extensibility, maintainability, internationalization (i18n), configurability.
- **+ — Constraints:** Design constraints, operating system dependencies, legal compliance (GDPR, HIPAA).

Topic 9: Feasibility Analysis — The TELOS Framework

Dimension	Evaluation Criteria	Quantitative Analytical Techniques
T — Technical	Can the system be constructed with available hardware, software stacks, and engineering skills?	Proof-of-concept prototyping, technology risk scoring.
E — Economic	Is the financial return on investment sufficient to justify capital expenditure?	Cost-Benefit Analysis (CBA), Net Present Value (NPV), ROI, Payback Period.
L — Legal	Does the project comply with intellectual property laws, data privacy, and industry regulations?	Compliance audits (GDPR, SOC2, HIPAA, PCI-DSS).
O — Operational	Will the end-users embrace the software within existing organizational workflows?	Change management assessments, user resistance surveys.
S — Schedule	Can the project meet strict market windows or hard statutory deadlines?	PERT schedule variance analysis, critical path evaluation.

🏛️ Solved Numerical 2: Economic Feasibility — Net Present Value (NPV) Calculation

A software project requires an initial development investment of $I_0 = \$100,000$. It generates annual net cash inflows of $C_1 = \$40,000, C_2 = \$50,000, C_3 = \$50,000$ over 3 years at a discount rate of $r = 10\%$ (0.10):

$$\text{NPV} = \sum_{t=1}^{n} \frac{\mathbf{C_t}}{(\mathbf{1}+\mathbf{r})^{\mathbf{t}}} - \mathbf{I_0}$$

- $$\text{PV}_1 = \frac{40,000}{(1.10)^1} = \frac{40,000}{1.10} = \mathbf{\$36,363.64}$$
- $$\text{PV}_2 = \frac{50,000}{(1.10)^2} = \frac{50,000}{1.21} = \mathbf{\$41,322.31}$$
- $$\text{PV}_3 = \frac{50,000}{(1.10)^3} = \frac{50,000}{1.331} = \mathbf{\$37,565.74}$$
- $$\text{Total Present Value} = 36,363.64 + 41,322.31 + 37,565.74 = \mathbf{\$115,251.69}$$
- $$\text{NPV} = \mathbf{\$115,251.69} - \mathbf{\$100,000} = +\mathbf{\$15,251.69}$$

$$\text{Conclusion: Since NPV} > \mathbf{0} \implies \text{Project is Economically Feasible \& Approved!}$$

Topic 10: Requirements Elicitation Techniques

- **Interviews:** Structured (formal question checklist) vs. Unstructured (open-ended discovery). Essential for executive stakeholder vision.
- **Questionnaires & Surveys:** Quantitatively samples large, geographically distributed user bases for statistical trend analysis.
- **FAST / JAD Workshops (Joint Application Development):** Intensive collaborative sessions bringing developers, users, and business analysts into a single room to construct agreed requirement baselines.
- **Ethnography & Observation:** Observing end-users in their natural work environment to discover tacit, unarticulated operational requirements.
- **Prototyping:** Building interactive UI mockups (Figma) to validate user mental models.

Topic 11: IEEE 830-1998 Standard for SRS Structure

📋 **Standard IEEE 830 SRS Document Hierarchy**

1. **1. Introduction:**
   - 1.1 Purpose (Audience and document objectives)
   - 1.2 Scope (Product name, benefits, and boundary limitations)
   - 1.3 Definitions, Acronyms, and Abbreviations
   - 1.4 References (Standards, project artifacts)
   - 1.5 Overview (Document structure)
2. **2. Overall Description:**
   - 2.1 Product Perspective (System context diagram, external interfaces)
   - 2.2 Product Functions (High-level functional summary)
   - 2.3 User Characteristics (Educational level, technical expertise)
   - 2.4 General Constraints (Regulatory, hardware limits)
   - 2.5 Assumptions and Dependencies
3. **3. Specific Requirements:** (Exhaustive detailed functional requirements, UI/hardware/software interfaces, database schema constraints, performance metrics, security encryption standards).

Topic 12: 9 Characteristics of a Good SRS (C-U-C-C-V-F-T-P-M)

Quality Attribute	Engineering Meaning & Verification Method
1. Correct	Every stated requirement accurately reflects true real-world user needs.
2. Unambiguous	Every requirement statement possesses exactly one possible interpretation (Zero vagueness: avoid words like "user-friendly", "fast").
3. Complete	Includes all essential capabilities, boundary conditions, error responses, and default states (No `TBD` sections).
4. Consistent	Contains zero internal contradictions between conflicting requirement specifications.
5. Verifiable (Testable)	There exists a finite, cost-effective automated or manual test case that can definitively prove requirement satisfaction.
6. Modifiable	Structured with unique numbering and cross-references so changes can be made without cascading errors.
7. Traceable	Each requirement has a unique identifier linked backward to business goals and forward to design modules and test cases.
8. Prioritized	Ranked by customer importance and technical stability (e.g., MoSCoW: Must-have, Should-have, Could-have, Won't-have).
9. Feasible	Achievable within system constraints, available budget, and technological capabilities.

Topic 13: University Exam Problem Bank & Model Answers

Q4. What is a Requirements Traceability Matrix (RTM)? Explain Forward and Backward Traceability with an Example Table. (8 Marks)

An **RTM** is a grid mapping requirements across all SDLC artifacts to guarantee full requirement fulfillment.
• **Forward Traceability:** Tracks each requirement forward into design components, source code classes, and test cases (ensures no requirement is forgotten).
• **Backward Traceability:** Tracks each test case and source code module backward to its originating requirement (prevents "gold plating" or building features nobody asked for!).

Req ID	Requirement Description	Architecture Component	Source Code File	Test Case ID
REQ-101	User password reset via OTP	AuthService.java	/auth/PasswordReset.java	TC-AUTH-01, TC-AUTH-02
REQ-102	PDF Invoice Generation	BillingModule.java	/billing/PdfGenerator.java	TC-BILL-05

Q5. Explain Why Ambiguous Terms in SRS Lead to Software Failure and How to Make Non-Verifiable Requirements Verifiable. (8 Marks)

Ambiguous terms lack quantitative boundaries, preventing quality assurance engineers from creating definitive pass/fail test cases.
• *Non-Verifiable Bad Requirement:* "The website shall load quickly and be user-friendly." (Subjective; impossible to test mathematically).
• *Verifiable Good Requirement:* "The web portal home page shall render the First Contentful Paint (FCP) within 1.5 seconds on a 4G mobile connection for 95% of global users, and achieve a System Usability Scale (SUS) score $\geq 80/100$ in usability testing."

Topic 24: Quality Function Deployment (QFD) & The House of Quality

QFD (Yoji Akao) is a structured technique that translates customer desires (*"Voice of the Customer"*) directly into actionable software engineering requirements:

QFD Requirement Category	Customer Expectation Profile	System Behavior & Satisfaction Impact
1. Normal Requirements	Explicitly requested in customer meetings.	Customer is satisfied if present, dissatisfied if absent (e.g., Search filter).
2. Expected Requirements	Implicit baseline capabilities assumed without asking.	Customer takes them for granted; absence causes catastrophic dissatisfaction (e.g., Data encryption, crash resilience).
3. Exciting Requirements	Unexpected, innovative delightful features.	Absence causes zero complaint; presence creates immense customer delight and market differentiation (e.g., AI auto-completion).

Topic 25: Class-Responsibility-Collaborator (CRC) Cards

CRC Modeling (Beck & Cunningham) provides an agile, physical index-card method for discovering object-oriented requirements:

📋 **Structure of a Standard 4×6 CRC Card**

- **Header:** Class Name (e.g., `OrderTransaction`).
- **Left Column (Responsibilities):** What the class knows (data) and what it does (computations: `CalculateTotal()`, `ValidateStock()`).
- **Right Column (Collaborators):** Other classes required to fulfill responsibilities (`Customer`, `InventoryDatabase`, `PaymentGateway`).

🏛️ Solved Numerical 8: Payback Period & Return on Investment (ROI)

An automated testing framework costs $\$60,000$ to engineer. It generates recurring annual QA labor savings of $\$25,000/\text{year}$ over a 4-year lifecycle:

- $$\text{Payback Period} = \frac{\text{Initial Investment}}{\text{Annual Cash Inflow}} = \frac{\mathbf{\$60,000}}{\mathbf{\$25,000}} = \mathbf{2.4}\ \text{Years}\ (28.8\ \text{Months})$$
- $$\text{Total 4-Year Inflow} = 4 \times \$25,000 = \$100,000 \implies \text{Net Profit} = \$100,000 - \$60,000 = \$40,000$$
- $$\text{ROI} = \frac{\text{Net Profit}}{\text{Total Cost}} \times \mathbf{100\%} = \frac{\mathbf{\$40,000}}{\mathbf{\$60,000}} \times \mathbf{100\%} = +\mathbf{66.67\%}$$

Topic 26: University Exam Problem Bank (Part II)

Q14. Explain the Delphi Method for Requirements Prioritization and Consensus Building. (8 Marks)

The **Delphi Method** achieves objective consensus among domain experts without peer pressure or dominance bias:
1. A facilitator circulates anonymous requirement ranking questionnaires to all distributed experts.
2. Experts independently assign priority scores and provide written justifications.
3. Facilitator compiles statistical summary (median and interquartile range) and redistributes it anonymously.
4. Experts review collective results and adjust their rankings over 2–3 iterative rounds until variance stabilizes into strong group consensus!

Q15. Explain Formal Methods in Requirements Engineering and the Role of Z-Notation. (8 Marks)

Formal Methods utilize mathematical discrete structures (first-order predicate logic, set theory) to specify software requirements with zero ambiguity.
• **Z-Notation:** Encapsulates system states and operations inside graphical formal boxes called **Schemas**. Each schema specifies *State Invariants*, *Pre-conditions* (what must hold before operation), and *Post-conditions* (what is guaranteed after operation: $x' = x + 1$). Essential for safety-critical systems (nuclear reactor controllers, pacemaker firmware).

Topic 27: Non-Functional Requirements (NFR) Framework & Softgoal Interdependency

The **Chung et al. NFR Framework** models non-functional requirements as **Softgoals** that are "satisficed" rather than strictly satisfied:

- **NFR Softgoal:** Abstract quality target (e.g., `Security`, `High Throughput`).
- **Operationalization:** Concrete design or architectural decision that contributes positively or negatively to the softgoal (e.g., `JWT Authentication`, `AES-256 Encryption`).
- **Interdependency Contribution Types:**
  - `MAKE (++)`: Strongly positive contribution.
  - `HELP (+)`: Moderately positive contribution.
  - `HURT (-)`: Moderately negative contribution (e.g., Heavy Encryption *HURTS* Latency!).
  - `BREAK (--)`: Catastrophic negative conflict.

🏛️ Solved Numerical 12: Requirements Prioritization via Analytical Hierarchy Process (AHP)

Compare 3 Requirements (R_1, R_2, R_3) in a Pairwise Comparison Matrix $\mathbf{A}$ where R_1 is $3\times$ more important than R_2, and R_1 is $6\times$ more important than R_3:

$$\mathbf{A} = \begin{bmatrix} 1 & 3 & 6 \\ 1/3 & 1 & 2 \\ 1/6 & 1/2 & 1 \end{bmatrix} \implies \text{Normalized Weight Vector } \mathbf{W} = \begin{bmatrix} \mathbf{0.667}\ (\mathbf{66.7\%}) \\ \mathbf{0.222}\ (\mathbf{22.2\%}) \\ \mathbf{0.111}\ (\mathbf{11.1\%}) \end{bmatrix}$$

$$\text{Priority Order: } \mathbf{R_1}\ (\text{Must-Have}) \gg \mathbf{R_2}\ (\text{Should-Have}) \gg \mathbf{R_3}\ (\text{Could-Have})$$

Topic 28: Formal Specifications in Safety-Critical Systems

Formal methods use discrete mathematics to prove program correctness before code generation:

📋 Model-Based vs. Property-Based Formal Specifications

- **Model-Based (Z-Notation, VDM, B-Method):** System state is modeled via sets, relations, functions, and sequences. Operations specify state transitions via mathematical preconditions and postconditions.
- **Property-Based (Algebraic Specifications, OBJ, Larch):** System behavior is defined implicitly through algebraic axioms relating operations (e.g., $\text{pop}(\text{push}(s, v)) = s$).

🏛️ Solved Problem 16: Z-Schema for Online Library Borrowing Transaction

```
┌─── BorrowBook ──────────────────────────┐
│ Δ LibraryState                          │
│ user? : UserID                          │
│ book? : BookID                          │
├─────────────────────────────────────────┤
│ user? ∈ RegisteredUsers                 │
│ book? ∈ AvailableBooks                  │
│ #(BorrowedBooks ▷ {user?}) < MaxLimit   │
│ AvailableBooks' = AvailableBooks \ {book?}│
│ BorrowedBooks' = BorrowedBooks ∪ {(book?, user?)}│
└─────────────────────────────────────────┘
```

Verification: Guaranteed to prevent exceeding checkout limits or borrowing unavailable books by mathematical proof!

🏛️ Solved Problem 21: Quality Function Deployment (QFD) Customer Importance Weighting

In a QFD House of Quality matrix, customer requirements (CR_1: Fast Search, CR_2: Data Security) have weights $W = [8, 10]$. Technical metrics (TC_1: Query Cache, TC_2: 256-Bit SSL) have relationship scores $R = [[9, 1], [3, 9]]$:

- $$\text{Technical Importance (Query Cache)} = 8(9) + 10(3) = 72 + 30 = \mathbf{102}$$
- $$\text{Technical Importance (256-Bit SSL)} = 8(1) + 10(9) = 8 + 90 = \mathbf{98}$$
- $$\mathbf{\text{Conclusion: Query Caching receives highest initial engineering allocation!}}$$

🏛️ Solved Problem 22: Requirements Change Impact Assessment Metric

For a proposed change impacting 4 classes in a 40-class system: $\text{Impact Ratio} = \frac{4}{40} = \mathbf{10.0\%}$. Change is classified as Moderate Impact and approved by CCB!

🏛️ Solved Problem 30: Use Case Modeling & Boundary Conditions in E-Commerce

Construct a complete Use Case Specification table for `Process Checkout`:

Field	Specification Description
Use Case ID	UC-CHK-01
Primary Actor	Authenticated Registered Customer
Pre-Conditions	Customer cart contains ≥ 1 available in-stock item; customer is logged in.
Main Success Scenario	1. Customer reviews cart. 2. Enters shipping address. 3. Selects payment method. 4. Authorizes charge. 5. System generates order ID and triggers PDF receipt email.
Extensions (Exception)	4a. Card authorization fails $\implies$ Prompt customer to retry with secondary payment method.
Post-Conditions	Inventory reserved, credit card charged, order record persisted in database.

🏛️ Solved Problem 31: Requirements Change Control Board (CCB) Formal Workflow

$$\text{Change Request (CR)} \xrightarrow{\text{Impact Analysis}} \text{CCB Review} \xrightarrow{\text{Approved}} \text{Baseline Updated} \xrightarrow{\text{RTM Modified}} \text{Engineering}$$

🏛️ Solved Problem 40: Software Requirements Inspection Checklist & Defect Classification

Classify requirements defects using IEEE standards:

- **Ambiguity:** "System shall load data quickly." $\implies$ Missing latency bound ($\leq 500\text{ms}$).
- **Incompleteness:** Missing behavior on network timeout $\implies$ Add automated retry with exponential backoff.
- **Inconsistency:** Section 2.1 states max users $= 1000$, while Section 3.4 specifies concurrent sessions $= 5000$ $\implies$ Reconcile system capacity baseline!

🏛️ Solved Problem 41: Object-Oriented Requirements Domain Modeling

In a University Course Registration domain: Extract entities (`Student`, `CourseOffering`, `Professor`, `Transcript`) and establish multiplicity constraints: A Student registers for $3..6$ CourseOfferings; a CourseOffering is taught by exactly 1 Professor!

🏛️ Solved Problem 50: Use Case Point (UCP) Estimation Method (Gustav Karner)

In Use Case Point analysis:

- $$\text{Unadjusted Actor Weight (UAW)} = \sum \text{Actors} \times \text{Weight} \quad (\text{Simple: 1, Average: 2, Complex: 3})$$
- $$\text{Unadjusted Use Case Weight (UUCW)} = \sum \text{UseCases} \times \text{Weight} \quad (\leq 3 \text{ steps: 5, 4–7 steps: 10, } \geq 8 \text{ steps: 15})$$
- $$\text{UUCP} = \text{UAW} + \text{UUCW} \qquad \text{UCP} = \text{UUCP} \times \text{TCF} \times \text{ECF}$$

🏛️ Solved Problem 51: Requirements Completeness Verification Matrix

In an ATM system specification: For every valid pin state, verify that all three failure conditions (Wrong PIN, Expired Card, Network Timeout) have explicit state transitions and localized error dialog prompts!

🏛️ Solved Problem 60: Requirements Volatility Index (RVI) Metric

If $R_{\text{initial}} = 80$, $R_{\text{added}} = 12$, $R_{\text{modified}} = 8$, and $R_{\text{deleted}} = 4$:

$$\text{RVI} = \frac{\mathbf{R}_{\text{added}} + \mathbf{R}_{\text{modified}} + \mathbf{R}_{\text{deleted}}}{\mathbf{R}_{\text{initial}}} \times \mathbf{100\%} = \frac{\mathbf{12 + 8 + 4}}{\mathbf{80}} \times \mathbf{100\%} = \frac{\mathbf{24}}{\mathbf{80}} \times \mathbf{100\%} = \mathbf{30.0\%}$$

An RVI $> 20\%$ signals high requirement volatility, requiring agile iterative planning rather than fixed-price waterfall contracts!

🏛️ Solved Problem 61: IEEE 830 Section 3 Software Interfaces Specification

Specify exact communication protocols, endpoints, payload formats (`application/json`), TLS 1.3 encryption, and error status codes (`400 Bad Request`, `401 Unauthorized`, `500 Server Error`) for zero interface mismatch!

🏛️ Solved Problem 69: Requirements Elicitation Cognitive Biases

Identify and mitigate cognitive biases in stakeholder interviews: (1) **Availability Heuristic** (users request features based on recent memorable outages), (2) **Anchoring Bias** (over-weighting the initial executive suggestion), (3) **Confirmation Bias** (analysts only hearing feedback supporting their pet architecture)!

🏛 Solved Problem 70: Use Case Generalization vs Extension

`Make Payment` is a generalized base use case specialized by `Make Credit Card Payment` and `Make Crypto Payment` via standard object-oriented polymorphism!

🏛 Solved Problem 79: Natural Language Requirements Ambiguity Patterns

Identify 4 common ambiguity pitfalls: (1) **Vague Adverbs** ("rapidly"), (2) **Loophole Conditionals** ("as far as possible"), (3) **Passive Voice** ("the password shall be hashed" $\implies$ by client or server?), (4) **Compound Conjunctions** ("A and B or C" $\implies$ $(A \land B) \lor C$ vs $A \land (B \lor C)$)!

🏛 Solved Problem 80: User Stories INVEST Criteria

$$\text{I} — \text{Independent} \bullet \text{N} — \text{Negotiable} \bullet \text{V} — \text{Valuable} \bullet \text{E} — \text{Estimable} \bullet \text{S} — \text{Small} \bullet \text{T} — \text{Testable}$$

🏛 Solved Problem 81: Requirements Traceability Forward and Backward Audit

In a Medical Device audit: Verify that 100% of safety requirements in SRS map forward to verified unit tests and hazard mitigation code modules with zero orphan code!

🏛 Solved Problem 93: Requirements Elicitation via User Personas

Construct realistic user personas (e.g., "Dr. Priya, Senior Radiologist, needs 1-click DICOM export within 2 seconds") to ground abstract requirements in real clinical workflows!

🏛 Solved Problem 94: Requirements Prioritization via Kano Model

- **Must-Be:** Core features users expect (login).
- **One-Dimensional:** Customer satisfaction scales linearly with performance (search speed).
- **Attractive / Delighters:** Innovative features creating high delight (AI instant summary)!

🏛 Solved Problem 95: Formal Verification of Requirements Consistency

Using SAT solvers (Z3) to mathematically prove that a set of 500 security rules contains zero mutually contradictory assertion deadlocks!

🏛 Solved Problem 107: Non-Functional Security Requirements Specification (OWASP ASVS)

Specify application security verification standards: Rate limiting ($100\ \text{req}/\text{min}/\text{IP}$), AES-GCM-256 at-rest database encryption, Argon2id password hashing, and zero raw PII in application debug logs!

🏛 Solved Problem 108: Requirements Traceability Forward/Backward Audit Checklist

Audit matrix verifying 100% of customer requirements have assigned test cases (≥ 1 pass/fail test per FR) and zero untested orphaned source code classes!

🏛 Solved Problem 109: System Architecture Boundary Definition in SRS

Explicitly delineate external third-party boundaries: Stripe Payment Gateway, Twilio SMS API, and AWS S3 Storage Interfaces with timeout fallback rules!

🏛️ Solved Problem 121: Requirements Elicitation via Ethnographic Work Observation

Shadowing air traffic controllers in live radar rooms to discover unstated cognitive overload bottlenecks during multi-aircraft conflict resolution!

🏛️ Solved Problem 122: Software Requirements Ambiguity Scoring & Defect Removal

Using natural language processing rule checkers to flag non-quantifiable adjectives ("efficient", "flexible", "intuitive") and auto-prompting requirement authors for quantitative bounding metrics!

🏛️ Solved Problem 129: Requirements Elicitation via Brainstorming vs Nominal Group Technique (NGT)

• **Brainstorming:** Unconstrained rapid generation of ideas without initial critique.
• **NGT:** Structured 4-phase technique (Silent idea generation $\rightarrow$ Round-robin recording $\rightarrow$ Group discussion $\rightarrow$ Anonymous mathematical voting), eliminating dominant stakeholder bias and yielding objective requirement priorities!

🏛️ Solved Problem 130: Functional Requirements CRUD Matrix Completeness Verification

In a Customer Management subsystem: Verify that for every data entity (`CustomerProfile`, `Invoice`, `CreditCard`), all 4 lifecycle operations (Create, Read, Update, Delete/Deactivate) have explicit business logic and authorization constraints!

🏛️ Solved Problem 139: Requirements Completeness Formal Proof

In a Medical Infusion Pump controller: Using model checking to verify that for every possible drug rate limit, the safety cutoff circuit is guaranteed to trigger within 50ms under all operational conditions!

🏛️ Solved Problem 143: Requirements Change Propagation Cost Model

Calculating the multiplier cost of requirements modification across the SDLC: $\text{Cost}(t) = C_0 e^{kt}$. A requirement defect caught in testing costs $20\times$ more to fix than during SRS inspection!

🏛️ Solved Problem 144: Formal Security Invariants Specification in Z-Notation

Mathematical proof that session tokens expire after $15\ \text{minutes}$ of user inactivity and cannot be forged without cryptographic secret key!

🏛️ Solved Problem 151: Requirements Engineering for Cloud-Native Distributed Systems

Specifying multi-region active-active database replication latency bounds ($< 50\text{ms}$) and automated partition tolerance guarantees adhering to the CAP Theorem!

🏛️ Solved Problem 152: User Story Acceptance Criteria using Gherkin BDD Syntax

```
Feature: User Password Authentication
  Scenario: Successful login with valid credentials
    Given the user is on the login portal
    When the user enters valid email and correct password
    Then the system should issue a signed JWT token and redirect to dashboard
```

🏛️ Solved Problem 159: Formal Verification of Requirements Completeness via Model Checking

Using Temporal Logic (LTL/CTL) to prove that the system guarantees safety invariant $\mathbf{G}(\text{Request} \implies \mathbf{F}\text{Acknowledge})$ across all 2^{30} reachable execution states with zero deadlock risk!

🏛️ Solved Problem 160: Non-Functional Scalability & Latency SLA Bounds

Contractual SLA engineering: Specifying 99.999% uptime (less than 5.26 minutes downtime per year) with p99 API latency bounded below 120ms at $50,000\ \text{QPS}$ load!