EVALUATING LLMS FOR AUTOMATED REQUIREMENT AND TEST CASE GENERATION IN RAILWAY SIGNALING SYSTEMS

Ionuț-Gabriel Oțelea
Răzvan Victor Rughiniș
Bogdan Pintea
Valentina Tîrșu

ROEDUNET CONFERENCE 2025

MOTIVATION

- **Requirements** lists for **complex technological products** quickly become **large** and **difficult to navigate**.
- Despite tools like **IBM DOORS** or **Polarion ALM**, specialists still spend significant **time refining** and **validating requirements**.
- This **manual effort** increases **development time** and **costs**, while reducing efficiency and resource availability for actual product development.

RELATED WORK

- Several studies propose **domain-specific languages** or other **model transformation** approaches to **formalize requirements** and automatically **derive test cases**.
- Systematic reviews show **400+ studies using NLP for requirements processing**, but only **~15 tools are publicly available**, with **limited industry adoption**.

RESEARCH GAP

- There is a **lack of systematic evaluations** of **LLM performance** in **safety-critical, regulated domains**.
- Current work focuses on **isolated tools** or **narrow case studies**, **without broad applicability**.
- **Without reliable benchmarks** across key dimensions (completeness, correctness, consistency, traceability), both **academic progress** and **industrial adoption** are **hindered**.

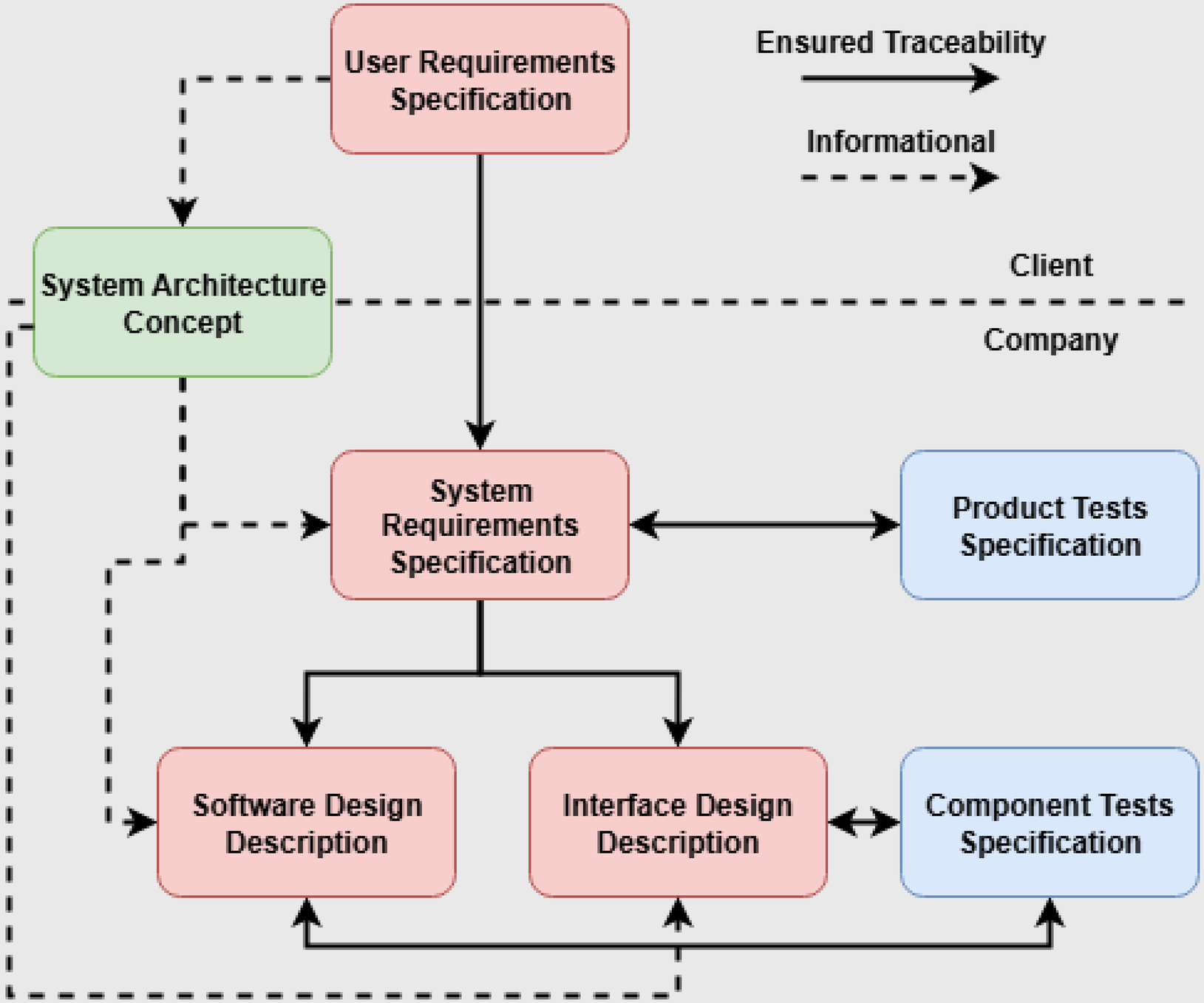
OBJECTIVES

- **First comparative evaluation** of multiple state-of-the-art **LLMs** (GPT-4, Claude, Gemini) in **railway signaling requirements engineering**.
- **Benchmark framework** grounded in **CENELEC standards** and **consistent metrics** (completeness, correctness, consistency, traceability).

RAILWAY SYSTEMS ARTIFACTS AND WORKFLOW

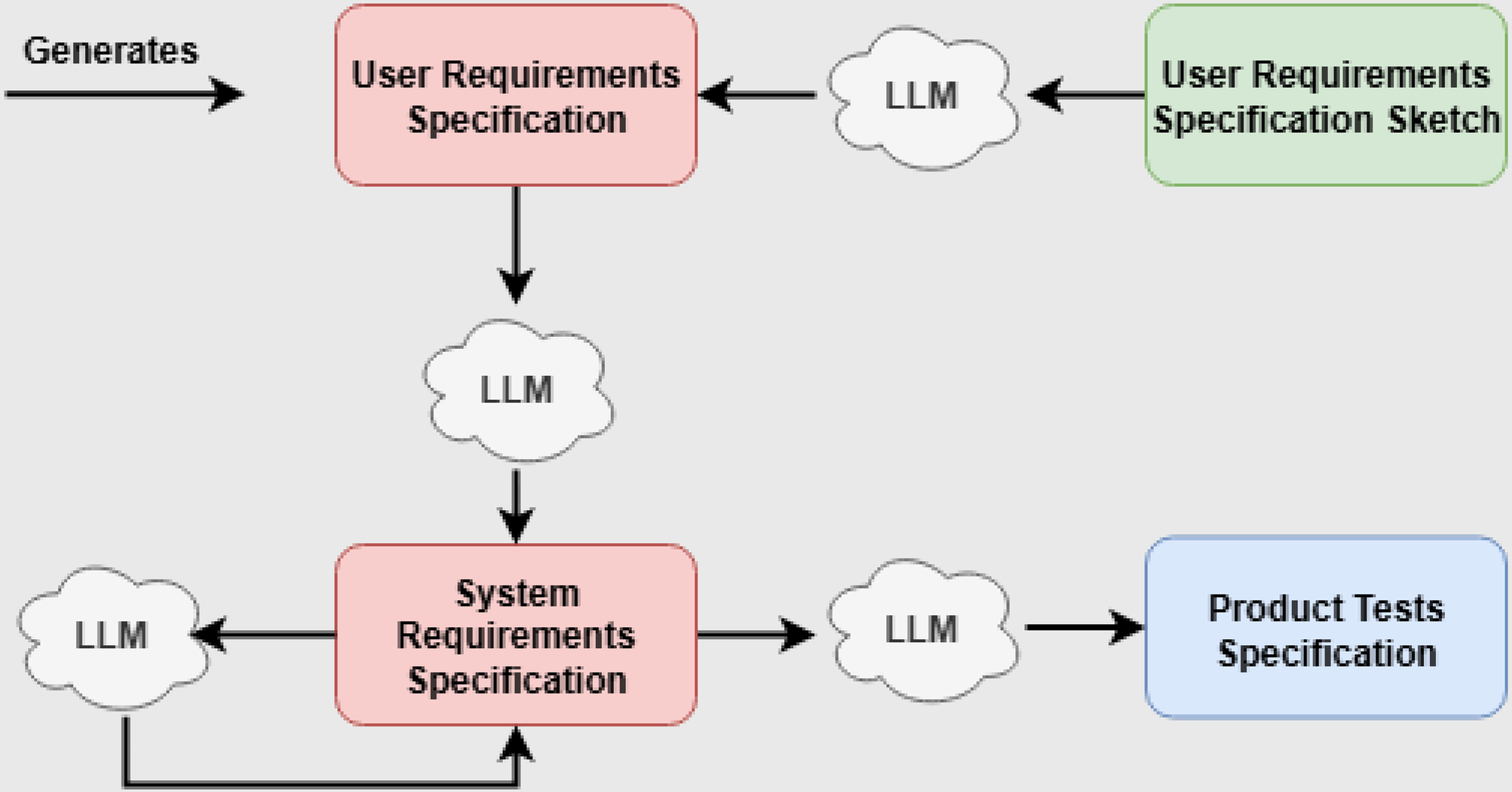
- **CENELEC EN 50126 / 50128 / 50129** define **strict rules** for railway software development to **ensure safety** and **quality**.
- **V-Model development cycle:** every **development stage** has a corresponding **verification & validation stage**.
- **Traceability chain:** user requirements → system requirements → design/interface requirements.

Example schematic of the relationship between a part of the railway system software products specific artifacts



TARGETED ARTIFACTS

- **Expand** generic customer **requirements**.
- **Evaluate** model **detection capability** when introducing **errors** & **inconsistencies.**
- **Generate** refined system **requirements** and **derive** corresponding product **test cases**.



The points of interaction between models and artifacts of interest

CHOSEN MODELS

- **GPT-4** (OpenAI), **Claude Sonnet 4** (Anthropic), **Gemini 2.5 Flash** (Google DeepMind/Microsoft).
- **Widely used**, strong **interaction quality**, **accessible APIs**, backed by **trusted companies**.
- **Open**/public **LLMs** exist, but often **lack APIs**, **require** significant local **infrastructure**, or deliver **weaker results**.

EVALUATION CRITERIA

- **Completeness, Correctness, Consistency, Traceability**.
- **Checklist** applied systematically to outputs from each model and **scoring** based on **fulfillment** of **sub-criteria**.

EVALUATION CHECKLIST FOR LLM OUTPUTS

Metric	Checklist Item
Completeness	Covers all aspects of the source requirement.
	Includes relevant conditions and constraints.
	Addresses boundary and failure cases.
	Provides sufficient detail for implementation or testing.
Correctness	Uses domain-specific terms accurately (e.g., railway signaling).
	Describes technically valid and logical behavior.
	Avoids internal contradictions or factual errors.
	Contains verifiable acceptance criteria or test steps.
Consistency	Uses consistent naming and terminology.
	Aligns with related requirements and artifacts.
	Maintains logical coherence across expansions or refinements.
	Follows defined templates or standards.
Traceability	System requirements reference their source user requirement.
	Each requirement or test case has a unique identifier.
	Test cases clearly link to the requirements they verify.
	Full trace chain is complete and can be verified and validated.

RESULTS

- Used model-specific **APIs** with **tailored prompts** and **input documents** for each artifact (requirements, refinements, tests).
- Each run was performed **without** prior conversation **history**, ensuring **predictable** and **unbiased outputs**.

USER REQUIREMENTS SPECIFICATION GENERATION

- Small set of **generic requirements** for a **track surveillance drone** used as **input prompt** under **CENELEC context**.
- All models produced structured documents with **headers** and **chapters** (Scope, Purpose, Approval, etc.).
  - **Gemini:** added subchapters but no unique IDs.
  - **Claude:** added IDs (e.g., UR-ENV-001), RAMS impacts, requirement priorities, SIL, and non-functional requirements.
  - **GPT-4:** concise, but generated additional unique requirements without being prompted.

SYSTEM REQUIREMENTS SPECIFICATION GENERATION

- **User Requirements Document** refined has been used as **input** for the models which were asked to derive technical **System Requirements** with **defined structure** and **obligation levels**.
  - **GPT-4:** Followed structure but mostly rephrased user requirements, with limited technical detail.
  - **Gemini:** created requirements with placeholders (TBD values), fragmented user requirements, little technical depth.
  - **Claude:** generated ~2.5× more requirements, introduced technical decisions (e.g., SIL2), imposed strict safety conditions, and referenced standards extensively.

SYSTEM REQUIREMENTS SPECIFICATION REFINEMENT

- Created a **modified System Requirements document** with **deliberate errors** (duplicate requirements/IDs, nonsensical values, contradictions, inconsistent units, editorial mistakes).
  - **GPT-4:** Missed duplicate requirements, flagged false conflicts (e.g., LiDAR + stereo cameras).
  - **Gemini:** Detected duplicates and corrected IDs, but missed editorial/physical impossibilities (e.g., 195% humidity).
  - **Claude:** Identified all introduced errors (even the ones introduced by itself before), flagged inconsistencies, and recommended additional safety requirements to align with CENELEC.

PRODUCT TESTS SPECIFICATION GENERATION

- **System Requirements** have been used as **input** and the models have been asked to **generate Product Test Specifications** (title, preconditions, steps, evaluation, manual/automated).
- All models **linked tests to requirements** and followed the requested structure, but **none achieved full 1:1 correlation** between actions and expectations.
  - **GPT-4:** Simplistic steps, limited detail.
  - **Gemini:** More detailed, but less technically rigorous.
  - **Claude:** Most advanced – inferred boundary values from other requirements, provided concrete limit testing, and added explicit pass/fail criteria.

EVALUATION #1

- **Scores:** GPT-4 - 9.5 / 16, **Claude - 13.5 / 16 (best overall)**, Gemini - 10.0 / 16
- **Key findings:**
  - **Completeness:** GPT-4 omitted edge cases; Gemini left placeholders; Claude inferred boundary values across requirements.
  - **Correctness:** GPT-4 valid but sometimes false positives; Gemini missed domain-specific errors (e.g., 195% humidity); Claude technically rigorous, though occasionally self-contradictory.
  - **Consistency:** GPT-4 missed duplicates; Gemini uniform but missed editorial issues; Claude best – detected contradictions, suggested restructuring.
  - **Traceability:** All models maintained it to some degree; Claude strongest (clear links + explicit pass/fail), GPT-4 sometimes incomplete, Gemini weakened by placeholders.

EVALUATION #2

- **Overall comparison:**
  - **GPT-4:** concise, fast, but weak in completeness & error detection.
  - **Gemini:** good structure, but struggled with technical correctness.
  - **Claude:** most rigorous – boundary cases + traceability, though not free from contradictions.
- **Failure modes:**
  - **GPT-4:** context length limits, overgeneralization.
  - **Gemini:** poor technical grounding, implausible outputs.
  - **Claude:** self-consistency issues across long outputs.

CONCLUSIONS

- LLMs can **support generating** and **maintaining requirements/test cases** in critical domains, but **human oversight is essential**.
- **Commercial API costs** become **prohibitive at industrial scale** and large documents and many users increase expenses.
- **Future directions:**
  - Explore open-source models with trainable weights on secure infrastructure.
  - Define robust validation pipelines & monitor model drift.
  - Develop best practices for prompt engineering and fine-tuning in safety-critical contexts.

THANK YOU

CONTACT: IONUT.OTELEA@UPB.RO

SPECIAL THANKS TO HITACHI RAIL GTS ROMANIA

HITACHI

Inspire the Next

QUESTIONS WELCOME