# AI4RE micro-credential

# Syllabus and study guide

Gunnar Harde, Matthias Herbert,
Simon Jimenez, Daniela Kreiner,
Michael Mey, Michael Tesar,
Franz Zehentner

0.99 beta 1 | January 8, 2026

## Terms of use

1. Individuals and training providers may use this syllabus and study guide as a basis for seminars, provided that the copyright is acknowledged and included in the seminar materials. Anyone using this syllabus and study guide in advertising needs the written consent of IREB for this purpose.
2. Any individual or group of individuals may use this syllabus and study guide as basis for articles, books or other derived publications provided the copyright of the authors and IREB e.V. as the source and owner of this document is acknowledged in such publications.

© IREB e.V.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without either the prior written permission of the authors or IREB e.V.

## Acknowledgements

This syllabus and study guide has been written by: Gunnar Harde, Matthias Herbert, Simon Jimenez, Daniela Kreiner, Michael Mey, Michael Tesar, Franz Zehentner.

Reviews were performed by: Rainer Grau and Ana Moreira.

Review comments were provided by: Gareth Rogers and Christian Ploninger. Copy editing by Stan Bühne and Stefan Sturm. English review by Gareth Rogers.

We thank everybody for their involvement.

Approved for release on October 23, 2025 by the IREB Council upon recommendation of Ana Moreira.

Copyright © 2016–2025 for this syllabus and study guide is with the authors listed above. The rights have been transferred to the IREB International Requirements Engineering Board e.V.

## Purpose of the document

This syllabus and study guide defines the content for the micro-credential **AI4RE (Artificial Intelligence for Requirements Engineering)** established by IREB. It serves two main purposes.

- Learners can use the syllabus and study guide to prepare for the self-assessment and to build a structured understanding of how Artificial Intelligence (AI) can be applied in Requirements Engineering (RE).
- IREB recognized training providers may use this syllabus and study guide as the basis for developing official training offerings that align with the learning objectives of the micro-credential.

The syllabus and study guide ensures consistency and transparency for both self-directed learners and training formats of IREB recognized training providers.

### Contents of this syllabus and study guide

This syllabus and study guide addresses the needs of all people involved in the topic of RE and seeking AI support for their work. It is aimed at anyone involved in Requirements Engineering activities, for example Requirements Engineers, Business Analysts, Product Owners, and Product Managers.

### Content scope

This syllabus and study guide was written to address the growing impact of artificial intelligence on RE. It reflects both IREB's perspective on responsible RE practices and an AI-oriented view of how RE can benefit from automation. Its content covers typical AI technologies and their application areas, the processing pipeline behind chatbots, and the specific strengths and limitations of approaches such as fine-tuning and retrieval-augmented generation (RAG). It highlights opportunities where AI tools can support eliciting, documenting, validating, and managing requirements, while stressing the importance of human oversight to ensure trustworthy outcomes and pointing out potential threats of AI usage.

### Level of detail

The level of detail of this syllabus and study guide allows internationally consistent learning. To achieve this, the syllabus and study guide includes:

- General educational objectives
- Contents with a description of the educational objectives
- References to further literature, where necessary

### Educational objectives / cognitive knowledge levels

All modules and educational objectives in this syllabus and study guide are assigned a cognitive level. The levels are classified as follows:

- **L1: Know** (describe, enumerate, characterize, recognize, name, remember, ...)—remember or retrieve previously learned material.
- **L2: Understand** (explain, interpret, complete, summarize, justify, classify, compare, ...)—grasp/construct meaning from given material or situations.

! The cognitive level L2 includes the cognitive level L1

**Example:**
A learning objective of the type "Understand the RE technique xyz" is at the cognitive knowledge level (L2). However, the ability to understand requires that learners know RE technique xyz (L1).

### Structure of the syllabus and study guide

The syllabus and study guide consists of 5 main chapters. Each chapter covers one educational unit (EU). The title of each main chapter reflects its cognitive level, defined by the highest level among its sub-chapters.

> **Example:** EO 3.1
> This example shows that educational objective EO 3.1 is described in subchapter 3.1.

In addition, there is a chapter on the core terminology which is essential for working with AI. This serves as a reference for the reader. In the examinations, these terms may be used in the context of RE-related questions.

### The self-assessment

This syllabus and study guide is the basis for the online multiple-choice self-assessment on AI4RE.

### Version history

| Version | Date | Comment |
|---|---|---|
| 0.99 beta | November 21, 2025 | Initial version |
| 0.99 beta 1 | January 8, 2026 | Formatting and numbering issues fixed |

Contents

1 AI Basics ..... 7

1.1 AI key technologies and their application areas ..... 7

1.2 Behind the scenes of chatbots: from prompt to response and RAG versus fine-tuning ..... 7

1.3 Key takeaways ..... 9

2 Large Language Models (LLMs) ..... 10

2.1 The fundamental concept of LLMs ..... 10

2.2 Key takeaways ..... 13

3 Prompt Engineering ..... 14

3.1 The importance of context ..... 14

3.2 Prompting patterns and prompting techniques ..... 15

3.3 Key takeaways ..... 17

4 Risks and Responsibilities in AI4RE ..... 18

4.1 Risks in AI4RE ..... 18

4.2 Responsibilities in AI4RE ..... 22

4.3 Key takeaways ..... 24

5 Use Cases for AI in RE ..... 25

5.1 Elicitation ..... 25

5.1.1 Exploring new domains ..... 26

5.1.2 Transcribing communication with stakeholders ..... 27

5.1.3 Extracting requirements across sources ..... 28

5.2 Documentation ..... 29

5.2.1 Formulating requirements ..... 30

5.2.2 Transforming requirements into other representations ..... 31

5.3 Validation ..... 33

5.4 Management ..... 34
5.4.1 Assignment of attributes to requirements ..... 34
5.4.2 Requirements prioritization ..... 34
5.5 Key takeaways ..... 35
6 AI Terminology ..... 36
7 References ..... 40

# 1 AI Basics

Level: L2

Goal: The candidate can differentiate among the various types of AI tools and models, understands how they are developed and deployed, and is familiar with key concepts such as information flow, RAG and fine-tuning.

### Educational objectives

EO 1.1 The candidate knows the variety of AI technologies and their application areas—not every AI is a chatbot. (L1)

EO 1.2 The candidate can explain what happens behind the scenes when interacting with a chatbot, can describe the concept of Retrieval-Augmented Generation (RAG) and compare this with fine-tuning. (L2)

## 1.1 AI key technologies and their application areas

Artificial Intelligence covers all methods that let computers perceive, reason or act in a way that we label intelligent. Depending on the author, you might find different categorizations of AI. Five appear in almost every taxonomy (for detailed definitions see chapter 6):

- **Machine learning (ML) / Deep learning**: Algorithms learn patterns directly from data.
- **Computer vision**: Computer vision transforms raw pixels into structured facts.
- **Natural language processing (NLP)**: NLP includes everything that turns speech or text into meaning and vice versa. This also includes Large Language Models (LLMs), discussed further in chapter 2.
- **Robotics**: Robotics combines sensor fusion, mapping, path planning and feedback control so machines can navigate and manipulate the physical world.
- **Expert systems**: Expert systems capture human expertise as explicit rules and ontologies, producing auditable, deterministic answers.

In summary, while these types of AI show that LLM-based chatbots are handy for requirements engineers to draft or re-phrase requirements, they represent only a part of the AI toolkit. Capabilities like meeting transcription (speech-to-text) and UI analysis (computer vision) are often integrated into a single interface. This can be achieved by combining specialized AI models or, increasingly, by using a single, multimodal model.

## 1.2 Behind the scenes of chatbots: from prompt to response and RAG versus fine-tuning

Chatbots became the best-known representatives of modern AI applications. The objective of this chapter is to understand the stages between typing a prompt and seeing a reply, as well as to distinguish fine-tuning from RAG.

A chatbot looks like a black box. Your request passes through several processing stages, each contributing to the answer you finally get.

Knowing this backstage flow helps you troubleshoot odd responses, craft sharper prompts and specify realistic system requirements.

The following simplified pipeline may vary slightly between different tool providers but captures the most important steps.

- **Input**: The user sends a prompt with optional context items like additional files, images or documents. If used in a chat, all the previous back-and-forth is added to the prompt as a pre-text. Imagine the chatbot not having a memory and having to read the whole conversation with each input again.
- **Pre-processing**: The system tokenizes [SeHB2016] text, removes unsafe content and normalizes language (in the case of a non-textual input type, for example speech, the system will also here first convert this input to text).
- **Context assembly**: The system context (see 3.1) and the session (or user) context are merged and form the final context applicable for the current conversation.
- **Inference:** The trained system, such as an LLM, uses its learned knowledge to generate an output or make a prediction from the new input data. It is the "run-time" phase where the model applies what it has learned during training. In RE, inference occurs when an LLM processes a prompt containing, for example, proposed requirements and then generates a reformulated requirement or identifies potential issues based on its understanding.
- **Post-processing**: The system converts raw tokens back to text or other result types (for example speech or images). Optional steps like fact-checkers, profanity filters or formatting rules may also be applied and vary strongly among different LLM model providers.
- **Output**: The system shows the result to the user in the chat window.

Providers may loop through context assembly and inference steps, call additional tools or follow up with chain-of-thought (see chapter 3.2) prompts until quality criteria are met.

In the step *context assembly*, two means of incorporating additional knowledge are:

1. **Fine-tuning** incorporates the additional knowledge directly in the model, thus making it available for all future interactions. A drawback of this method is that the model must be retrained every time you add new information, which can be quite resource-demanding. Also, the source of the knowledge is harder to pin-point, since after fine-tuning it becomes part of the model.
2. **RAG** utilizes a separate knowledge storage which can be accessed with a search engine triggered by the model. Newly added data/information is available instantly without re-training of the model and the source can be traced back to the knowledge storage.

Fine-tuning excels when you can afford recurring training effort and need low-latency answers without a separate knowledge store, but it risks becoming outdated and offers no built-in source citations. RAG keeps content fresh and traceable.

A drawback is that RAG introduces extra search latency and depends on the quality of your retrieval index, i.e. the retrieved information is only valuable if it is of good quality.

## 1.3 Key takeaways

AI has so much more to offer than chatbots. For some use cases, LLMs are not the right tool. Matching the task to the right AI technology is key to obtaining the best possible output; knowing the available possibilities is key to using the full potential of AI.

Adding knowledge to your current chatbot can be done by either fine-tuning and incorporating the knowledge into the model itself or by using RAG to be more flexible, adding knowledge from external sources on demand.

# 2 Large Language Models (LLMs)

Level: L1

Goal: The candidate can state that LLMs are probabilistic text prediction systems and list their common capabilities and limitations. The candidate knows what to look for when interacting with LLMs, recall criteria for reviewing their output, and list aspects to consider for their responsible integration into workflows.

#### Educational objectives

EO 2.1 The candidate knows the fundamental concept of how an LLM works at a high level. The candidate can describe how LLMs work based on probabilistic next-word prediction and not based on reasoning or understanding. (L1)

## 2.1 The fundamental concept of LLMs

#### The core mechanism: prediction, not understanding

In recent years, Large Language Models [ZhWX2023] have emerged as a class of AI systems capable of processing and generating human-like text. They are trained on massive amounts of textual data and use statistical patterns in language to perform tasks such as drafting documents, summarizing information, or generating test cases [VoFi2025]. For a Requirements Engineer, understanding the capabilities and limitations of LLMs is relevant because these systems can support a wide range of RE activities—from eliciting and documenting requirements to creating domain glossaries—while also introducing new risks and considerations [NoeA2024], [ZaDA2025].

When people try a Large Language Model for the first time, the answers can feel human. The text is fluent, the tone fits the question, and the result often seems to show understanding. The real story is simpler: LLMs are statistical text prediction systems that generate one word at a time based on patterns learned from training data. They do not understand or reason—they match patterns and continue writing. Given some words, they pick the next small chunk that is most likely. One can think of it as a very powerful autocomplete. That one idea explains most of its behavior.

#### Probabilistic generation and its consequences

The model is trained on a huge amount of text. During training it sees many examples of how words follow each other in different situations, from casual chat to technical writing. Its goal never changes: for any given context, guess the next token, which is a small piece of text that may be a word, part of a word, or punctuation. Later, when you ask a question, the model uses what it learned to continue your text one token at a time, each choice based on everything that came before.

This works well because language is full of patterns. Grammar creates structure, common phrases repeat across topics, and many documents follow a predictable structure. If a system is very good at continuing these patterns, the output feels coherent and relevant. Modern models use an architecture called a transformer [VaeA2017].

A key part of it, self-attention, helps the model decide which parts of your input are most important for the next step. That is why the model can tie a pronoun back to the right name or keep track of a topic that was introduced many sentences earlier.

### Implications and mental models for Requirements Engineering

The important point is what the system is not doing. It is not thinking like a person, and it does not have an inner picture of the world. There is no intent or understanding behind the sentences. The model recognizes and reproduces patterns it has seen across many texts. Because those patterns are rich and varied, the result can look like understanding, but the mechanism remains a statistical prediction.

At the moment of generation, the model produces a probability for each possible next token given the entire context so far. It then picks one and adds it to the output. Sometimes it selects the top choice, which tends to produce safe, steady text. Sometimes it samples from the distribution to add variety. A small amount of randomness can help avoid dull or repetitive answers. In every case the visible text is the result of many such small decisions, each one a choice of next token predicated on what is already there.

Since this is a probabilistic process, the model can make mistakes that still sound convincing. If you ask it to perform a long multiplication, for example, it may output a number that looks reasonable but is incorrect. It did not calculate it. It simply predicted a number that statistically follows the pattern of a multiplication problem, without performing the actual calculation. To correctly calculate deterministic tasks, i.e., tasks that follow a clear algorithm, certain models use a tool in the background

[SceA2023], such as a calculator or a code runner, while the language model processes the instructions and the explanation. The model continues to predict the text and uses the tool for the deterministic task.

People sometimes ask a model to show its steps [WeeA2022]. This can improve results on tasks that benefit from structure. The reason is simple: the model has seen many examples of step-by-step explanations in its training data (from textbooks, tutorials, Q&A data), so it can mimic that format. Reasoning systems even run private planning steps before giving an answer. They draft and revise internally, then present a result. This can raise quality, but it does not change the core. The system is still predicting text, just arranged into several rounds before you see it.

It helps to separate two layers (surface and factual) when you read model output. The *surface layer* covers fluency, tone, and structure, and it is usually very strong. These features are common in the data and easy to reproduce. The *factual layer* covers dates, quantities, names, and edge cases. This is where pure pattern matching can fail. If the training data is thin or inconsistent on any point, the most likely continuation may drift away from the truth. Connecting the model to search engines, databases, or function calls improves this layer because the next token prediction is grounded on fresh or computed information [LeeA2020]. The model still predicts, but it predicts with better inputs.

For a Requirements Engineer, the most effective mental model for an AI is a powerful autocomplete. This immediately clarifies its role: it is a drafting assistant, not a domain expert.

The model is highly sensitive to patterns, so provide a clear example of the output you want it to generate [BreA2020]. If you need a requirement in a specific style, include a short example and ask the model to match it for new content. This shifts the probabilities toward continuations that fit your template and your style. You may even copy several user stories you wrote in the past to set the tone and structure. If you care about quality criteria such as testability, unambiguous phrasing, or measurable criteria, state these directly. The model will often include those features because similar instructions co-occur in its training data.

In requirements engineering this matters for both drafting and reviewing. A vague instruction such as "write a requirement from this text" invites a vague continuation. A precise instruction that includes a template and a clear definition of the expected level of detail guides the output toward your standards. You can also ask the model to flag ambiguous terms and propose measurable alternatives. It will usually produce a useful first pass, since it has seen many examples of how people discuss clarity and measurement.

### A glimpse into the technical foundation: embeddings and vectors

These models are not deterministic by default. If you ask the same question twice, you might get different answers due to the probabilistic nature of how they generate text. Even a single word in your prompt can nudge the model onto a different path and produce a very different result. This sensitivity means the model can get sidetracked by one stray term or example and drift away from what you want. However, multiple different answers can all be correct and valid responses to your prompt.

A quick word on why this happens. When the model writes, it does not always pick the single most likely next token. It samples from a range of likely options. The setting that controls this randomness is called temperature: low temperature makes the model stick to safer, more obvious choices, which often leads to more consistent answers; higher temperature lets it explore and vary more, which can be useful for brainstorming but is less stable.

Regardless of this variability, you should always treat AI output as a draft that needs review. This review serves two important purposes: first, to identify and rule out hallucinations (factually incorrect information the model presents as true [YeeA2023]), and second, to optimize the result for your specific context and requirements. The model is not reliable in the way a calculator is—even when it produces consistent outputs, those outputs may still contain errors or may not perfectly match your needs.

This perspective also clarifies risk. If the model gives an incorrect number or cites the wrong standard, it is not failing at logic. It is succeeding at prediction in the wrong place. The remedy is verification and grounding. Provide the right context in your prompt, connect the model to retrieval or calculation tools when precision matters, and treat the output as a draft that needs review. None of this requires the model to understand. It only requires that you guide it toward the right pattern, and that you rely on tools and sources where patterns alone are not enough.

**If you are interested in why this may work:**
The model also represents tokens as vectors, which are long lists of numbers. Tokens that appear in similar contexts get similar vectors. This creates a kind of geometry that lines up with meaning in practice. A simple example from earlier methods makes the idea concrete. The difference between the vectors for king and queen tends to resemble the difference between man and woman [MiSC2013]. The system is not storing dictionary entries; it is learning numerical patterns that reflect how people use words.
This is why the model can grasp semantic meaning. Tokens that appear in similar contexts are given similar numerical representations, called embeddings. In practice, this means the vector for 'user requirement' is closer to 'stakeholder need' than it is to 'database schema'. This allows the LLM to identify similar requirements or find dependencies, even when different terminology is used.

## 2.2 Key takeaways

In summary, this chapter describes fundamental behaviors and terminology of LLMs that influence their application in RE. A Requirements Engineer who understands prompts, tokens, context windows, hallucinations, fine-tuning, and related concepts such as embeddings and RAG can better assess when and how to employ LLMs effectively, ensuring that generated outputs add value and do not compromise the quality of the requirements.

# 3 Prompt Engineering

Level: L2

Goal: The candidate understands prompting fundamentals including quality criteria, techniques and patterns, and can explain how these elements influence the quality of AI-generated outputs in RE.

### Educational objectives

EO 3.1 The candidate can explain the importance of context for producing accurate results when using AI-based tools. (L2)

EO 3.2 The candidate can name the general purpose of prompting patterns in RE and recognize how they differ from prompting techniques. (L1)

## 3.1 The importance of context

Large language models like GPT, Claude, Gemini or similar generate text by predicting the most probable next output token. This does not indicate an "understanding" of the input and the current situation in a conversation. A clearly defined context narrows the solution space and guides the model towards domain expertise, resulting in more precise and actionable output.

There are two layers of context that shape the quality of results:

- **System context:** The system context is fixed for every session. It originates from the system or initial prompt provided by the model's creator, the model's training data, and its built-in settings like context window length. These factors determine aspects such as the default language, the tone of responses, and the knowledge cut-off date. For example, a model may always answer in English, use a neutral style, and only include knowledge up to a certain date. The system context is usually not changeable but can be complemented by user input.
- **User context:** The user context is dynamic and changes with each conversation or even with each message you exchange with an LLM. This may include details about the project or product, such as a medical device compliant with ISO 13485, a PSD2-compliant banking system, or an e-learning platform. It also covers the perspective of stakeholders like nursing staff, financial analysts, or students, as well as applicable standards and regulations such as IEC 62304 [IEC62304], GDPR [GDPR2016], or FDA 21 CFR Part 11 [FDA1997]. In addition, the user context can define the role the AI should take (for example, "You are an expert Requirements Engineer"), specify the purpose of the query (such as generating test cases or validating requirements), and provide supporting material like templates or examples.

When context is missing, the model relies solely on its training data and produces answers based on statistical probability. Such output may sound plausible but often lacks domain relevance and practical value. In RE, this usually means the result is neither actionable nor of sufficient quality.

## 3.2 Prompting patterns and prompting techniques

When interacting with large language models, output quality depends not only on context (what the model should know) but also on how instructions are phrased.

### Prompting patterns

Prompting patterns are reusable structures that help formulate prompts consistently and effectively.

One common pattern is Role-Task-Format (RTF):

- **Role**: A description of the role the AI should assume.
- **Task**: The specific job to perform.
- **Format**: The expected structure of the output.

> **Example**:
> You are a professional business analyst who creates clear summaries of workshop results.
> Read the following workshop notes and create a structured summary that highlights goals, insights, decisions, and next steps.
> Provide the output as short bullet points under clear section headings.

By combining these three elements, users can guide the model away from generic answers and towards more precise and actionable results.

Many such patterns exist (see [WheA2023]). Some important examples are:

- **Chain-of-Thought**: Ask the AI to explain its reasoning step by step before giving the final result.
- **CRISPE**: Detailed pattern with Context, Role, Instructions, Steps, Parameters, Examples.

### Benefits of prompting patterns

Prompting patterns offer several advantages that improve both the quality of AI outputs and the efficiency of the RE process.

- **Practical benefits:** Prompting patterns increase the usability of answers by providing clear instructions upfront. Defining the desired format reduces the need for post-processing and saves time. Well-structured prompts also shorten the number of iterations required to reach useful results.
- **Cognitive benefits:** Using prompting patterns forces users to think carefully about their needs before asking the AI. By making role, task, and format explicit, ambiguity is minimized and the reasoning behind a prompt becomes transparent and reproducible.

- **Methodological benefits:** Prompting patterns can be reused across different tasks, ensuring consistency and lowering the learning curve for new users. Within a team, they create a shared standard for working with AI. When combined with prompting techniques (see below), they provide both a stable structure and situational flexibility, leading to more reliable and adaptable results.

### Prompting techniques

Prompting techniques are situational ways of phrasing prompts. Unlike patterns, they don't prescribe structure but influence how the model interprets requests:

- **Zero-Shot:** a request without providing examples

> **Example**:
> Please read the following interview transcript and generate a concise summary that highlights the main topics, key arguments, and important statements. Avoid personal interpretations.
> Transcript: [add interview transcript here]*
> * Text in brackets [] is a placeholder for input provided by the user

- **Few-Shot:** provide a few examples, then make your request

> **Example**:
> Here are two examples of how to convert requirements into user stories
> Requirement: "Search tasks" → User Story: As a user, I want to search tasks so that I can find items quickly.
> Requirement: "Reset password" → User Story: As a user, I want to reset my password so that I can regain access.
> Now your task:
> Convert the following requirements into User Stories:
> – Assign priorities to task
> – Add comments to a task
> – Filter tasks by status

- **Template-Based:** provide a structure the AI shall use for its results

> **Example**:
> Read the following workshop notes and summarize the results. Use exactly this structure:
> 1. Workshop Title:
> 2. Date & Participants:
> 3. Main Goals: 3–5 bullet points
> 4. Key Insights: bullet list
> 5. Decisions Taken: bullet list
> 6. Open Questions / Next Steps: bullet list
> Workshop Notes: [add workshop notes here]*
> * Text in brackets [] is a placeholder for input provided by the user

## 3.3 Key takeaways

Large language models predict, they do not understand. In RE, reliable AI results depend on precise and complete context. Without context, outputs fall back on clichés and assumptions; with rich context, they reflect domain expertise. Yet good context alone is not sufficient: the way prompts are designed determines how effectively AI can use that context. Prompting patterns provide reusable structures, while prompting techniques refine interpretation and style. Together, they support consistent, context-aware results tailored to stakeholder needs.

# 4 Risks and Responsibilities in AI4RE

Level: L2

Goal: The candidate can explain the risks, limitations, and data protection requirements when using AI in RE and can describe key aspects of responsible and compliant use.

### Educational objectives

EO 4.1 The candidate can name typical risks associated with the use of AI in RE. (L1)
EO 4.2 The candidate can explain the evolving role of the Requirements Engineer in AI-supported processes. (L2)

The effective application of AI in RE demands comprehensive human proofing, validation and curation across all supported activities. AI-generated questions, glossary entries, and extracted content require careful evaluation to ensure alignment with the project context and stakeholder needs. While AI can identify terminology patterns and process interview transcripts, automated outputs may misinterpret domain-specific concepts or introduce ambiguities that compromise shared understanding (CPRE Principle 3, [IREB2024]). The validation process becomes particularly critical when AI processes written communication. Casual observations and speculative discussions require human interpretation to distinguish genuine requirements from preliminary ideas that may not warrant formal documentation through established elicitation techniques (see chapter 4.2 in [GleA2024]).

Template-based AI outputs, including formatted requirements and user stories, demand rigorous verification to ensure semantic accuracy and stakeholder alignment. AI-generated requirements must maintain their original meaning while adhering to quality criteria (see chapter 3.8 in [GleA2024]), supporting value-orientation (CPRE Principle 1, [IREB2024]) rather than achieving mere formal compliance. Similarly, AI-created visual representations, such as prototypes and UML diagrams, require validation to confirm that they are at an appropriate abstraction level (see chapter 3.1.2 in [GleA2024]) and aligned with the intended purpose of model-based work products (see chapter 3.4 in [GleA2024]), and do not, for example, introduce any premature design assumptions. If requirements attributes and metadata are assigned with the help of AI, human judgment is required to confirm that critical factors such as business value and stakeholder approval status have been correctly evaluated. The proposed assignments must accurately reflect project status and support effective requirements management (see chapter 6.5 in [GleA2024]) and should not result in misleading information.

## 4.1 Risks in AI4RE

The use of AI in RE offers significant benefits but also introduces specific risks that can impact the accuracy, relevance, and trustworthiness of outputs. These risks arise from both the technical limitations of AI models and the way they are applied in RE contexts. Understanding the risks discussed below is essential for Requirements Engineers, as it enables them to anticipate potential issues, implement mitigation strategies, and maintain the integrity of the requirements process.

### Hallucinations

LLMs can produce plausible but factually incorrect outputs, inventing details or references that simply do not exist. In RE, this can result in fabricated standards, stakeholder statements, or system behaviors being treated as valid requirements. The model has no internal concept of "knowing", so it cannot indicate when it is guessing or lacking sufficient data.

> **Example**:
> The AI cites "ISO 20258p" as a security guideline, even though no such standard exists.

### False precision

Some AI outputs appear formally correct but are semantically meaningless or unverifiable. Requirements phrased in technically precise language may still be untestable or ambiguous in practice.

> **Example**:
> "The system shall perform optimally at all times" may sound plausible, but "optimally" is undefined and "at all times" is unrealistic and cannot be tested.

### Bias amplification

AI models may reinforce or amplify biases present in their training data, leading to systematically skewed requirements or personas. This can introduce unfairness or discrimination into the system design if undetected.

> **Example**:
> In an HR system requirements project, the AI is tasked with generating personas for job applicants. Based on biased historical recruitment data, it consistently describes leadership candidates as male, aged 35–50, and with a background in finance. This bias, if not corrected, risks embedding discriminatory assumptions into automated hiring workflows and even violating equal opportunity laws.

### Data leakage

Sensitive or confidential information can be unintentionally exposed when used with external AI services. Even anonymized data may be identifiable if combined with other sources.

> **Example**:
> In a healthcare project, anonymized transcripts of patient interviews are sent to a cloud-based AI summarization service outside the European Union[1]. The responsible data protection authority later determines that unique combinations of symptoms and treatment dates in the transcripts are sufficient to identify individual patients. This constitutes a GDPR [GDPR2016] violation and results in a very high fine.

### Intellectual property violations

AI outputs may reproduce protected or confidential material from training data or earlier prompts, raising legal and contractual risks.

> **Example**:
> The AI produces a requirements specification that contains verbatim text from a competitor's product manual.

### Outdated knowledge

Models trained on historical data may enter operation already outdated, for example lacking recent standards, technologies, or regulations. This is a static limitation present from deployment and is particularly critical in regulated domains, where outdated requirements can lead to non-compliance.

> **Example**:
> The AI recommends a constraint involving a network protocol that was deprecated two years ago.

[1] While examples in this chapter may reference EU regulations, they are intended as illustrative only and should be adapted to the legal framework and cultural context of each project.

### Model drift

Over time, the accuracy of an AI system's output can degrade due to evolving domain knowledge, data patterns, or operational conditions. This is a dynamic risk that appears during use. Without monitoring, this drift can result in subtle but significant errors.

> **Example**:
> An AI fine-tuned for banking compliance in 2023 starts producing outdated recommendations after regulatory changes in 2024.

### Context window overflow

When the amount of input exceeds a model's context capacity, earlier information may be dropped, leading to incomplete or inconsistent outputs.

> **Example**:
> During a large-scale requirements elicitation workshop, the AI model (with a 16,000-token context limit) processes meeting notes in real time. Early stakeholder statements about security requirements are silently dropped from the output because they exceed the model's context capacity, resulting in incomplete specifications and the omission of critical security features.

### Scope misalignment

AI may generate requirements that are out of scope for the project or that conflict with agreed objectives, particularly if prompts are vague or incomplete.

> **Example**:
> While drafting requirements for a mobile app, the AI starts proposing features for a desktop version that was never planned. The development team spends several days analyzing these irrelevant features before realizing they are out of scope, delaying the agreed design phase.

### Over-reliance on AI

Relying too heavily on AI outputs without adequate human validation can lead to the acceptance of flawed or biased requirements, reducing overall quality and increasing project risk.

> **Example**:
> A project team in the automotive sector adopts AI-generated safety requirements for an autonomous driving feature without proper human review. Months later, during regulatory inspection, auditors discover that several requirements contradict mandatory safety standards, forcing a costly redesign and delaying product launch.

### Opacity

Many AI models function as black boxes, making it difficult to trace how specific outputs were derived. In RE, this lack of transparency can undermine stakeholder trust and complicate the validation of requirements.

> **Example**:
> The AI proposes prioritizing certain requirements but cannot explain the ranking logic, making review difficult.

### Transparency obligations

Regulations such as the EU AI Act introduce a legal requirement to disclose when content is generated by an AI, though these rules are quite specific and do not apply to all outputs. This creates a compliance risk if teams publish certain types of AI output without the necessary labels or control.

For text output, which is common in RE, the obligation to label content as AI-generated applies specifically when the text is intended to inform the public on matters of public interest. Most typical RE artifacts, such as internal specifications, user stories, or backlog items, would not fall into this category.

Disclosure is also not required if the AI-generated output has been subject to a process of human review and a person or organization takes editorial responsibility for its publication. In an RE context, this aligns with the principle that a Requirements Engineer, Product Owner, or another stakeholder must always validate and take ownership of requirements before they are accepted.

> **Example**:
> A team uses an AI to generate a published report from a set of approved requirements, detailing how their new financial software complies with government transparency regulations. This report is a matter of public interest. If the report were published automatically without a compliance officer or project lead reviewing it and taking official responsibility for its content, it would likely breach AI transparency laws. The organization could face legal penalties for presenting AI-generated analysis on a public matter without the required disclosure or human oversight.

## 4.2 Responsibilities in AI4RE

The introduction of AI into RE processes does not replace the Requirements Engineer; instead, it *broadens and deepens* the role.

The Requirements Engineer now acts not only as elicitor, documenter, and validator of requirements, but also as *governor of AI-supported processes*, ensuring that outputs are relevant, trustworthy, and compliant with applicable rules and ethical standards.

Key aspects of the expanded role include:

### AI process designer, integrator & technical collaborator

The Requirements Engineer contributes to the selection of AI tools, defines their integration into the RE process, and collaborates closely with technical teams (e.g., AI developers, data scientists, IT) to ensure tools are properly configured, trained, and maintained. This includes assessing their suitability for specific RE tasks (e.g., classification, summarization, quality checks) and aligning input/output formats with modelling and documentation standards.

> **Example**:
> Selecting an LLM for removing requirement duplicates, integrating it with the team's repository, and defining rules for how duplicates are flagged and reviewed.

### Prompt designer and quality reviewer

The Requirements Engineer crafts and refines prompts that guide AI towards relevant, precise, and context-aware outputs, then critically evaluates these outputs against domain knowledge, stakeholder needs, and quality criteria. This ensures that AI suggestions are never adopted without human verification.

> **Example**:
> Designing a prompt template for generating acceptance criteria that explicitly includes performance thresholds, and rejecting any AI-generated requirement that is not testable.

### Risk and compliance manager

The Requirements Engineer identifies and mitigates legal, ethical, and operational risks of AI use—from protecting personal data and avoiding bias to ensuring that outputs meet regulatory and contractual requirements. This includes choosing processing options (e.g., local vs. cloud), ensuring data minimization, and detecting model drift.

> **Example**:
> Pseudonymizing stakeholder feedback before uploading to an AI service hosted outside the jurisdiction and routinely checking whether the AI is still producing regulation-compliant suggestions.

### Stakeholder communicator and trust builder

The Requirements Engineer ensures that stakeholders are transparently informed about the use of AI, its role in the RE process, its benefits, and its limitations. This includes facilitating learning, negotiation and consensus building activities among stakeholders: that is, the human activities involved in any RE process but requiring particular attention when aspects of that process have been automated or accelerated with AI.

It also includes explaining AI-supported outputs and providing opportunities for discussion when there are ethical concerns or reservations about the use of AI itself. Building trust means actively addressing these points rather than avoiding them.

**Example**:
In a requirements elicitation workshop, the Requirements Engineer explains which steps are AI-assisted, invites stakeholders to express concerns, and facilitates a discussion on how to handle sensitive data and potential AI limitations in the project.

## 4.3 Key takeaways

AI intensifies, rather than replaces, critical thinking in RE. The Requirements Engineer must recognize and mitigate risks, critically evaluate AI outputs, and ensure that only validated results enter the requirements baseline. Beyond monitoring, the Requirements Engineer takes on a strategic role: designing AI-supported processes, safeguarding compliance, and documenting decisions for auditability. Technological competence, domain expertise, and human judgment remain decisive to ensure that AI enhances value and fosters stakeholder trust.

# 5 Use Cases for AI in RE

Level: L2

Goal: The candidate can explain how AI tools can support RE activities, describe their limitations, and clarify the need to validate AI-generated outputs.

Terms: Elicitation, documentation, validation, management, exploration, transcription, extraction, system context, formulation, transformation, semantic accuracy, quality criteria

### Educational objectives

EO 5.1 The candidate can name typical use cases for AI in eliciting, documenting, validating, and managing requirements. (L1)

EO 5.2 The candidate can explain the importance of human curation of AI outputs in RE and describe the Requirements Engineer's role in ensuring valid results. (L2)

The advent of AI technologies presents new opportunities to support Requirements Engineers in their core activities. However, it is crucial to understand that AI serves as a supportive tool rather than as a replacement for human expertise and judgment in RE work.

AI technologies can assist Requirements Engineers across the four core RE activities: elicitation, documentation, validation, and requirements management [IREB2024]. They offer the potential to accelerate tasks, broaden perspectives, and enhance consistency checks. This chapter explores illustrative use cases for each activity, while also underscoring that AI-generated outputs must undergo careful human review. Ultimately, the Requirements Engineer remains accountable for ensuring that such outputs are purposeful, methodologically sound, and aligned with project goals and quality standards.

The proper application of AI in RE demands a thorough understanding of both the capabilities and limitations of these technologies. Requirements Engineers must develop the competence to critically assess AI outputs, recognizing where they add value but also where they may introduce risks or inaccuracies. When working practically with AI tools, all principles, practices, and quality criteria established in the preceding chapters of this document remain fully applicable and must be carefully considered. For comprehensive information on fundamental RE concepts and practices, consult the CPRE Foundation Level syllabus [IREB2024] and the accompanying handbook [GleA2024].

Note that you may find additional practical examples on implementing AI support in RE practices in the #AIREB Special Interest Group booklet [IREB2025].

## 5.1 Elicitation

Elicitation is at the heart of RE. It involves gathering, interpreting, and structuring information from sources like stakeholders, documents, and systems to understand what a system must achieve. This process is not limited to asking questions: it requires active listening, contextual understanding, and the ability to translate informal input into formal, actionable requirements.

There are three areas within elicitation where AI can provide effective support:

- **Exploration:** Before requirements can be elicited, Requirements Engineers must first understand the domain, goals, stakeholders, and constraints. AI tools can support this by summarizing domain knowledge, identifying stakeholder groups, and helping prepare structured interview guides and workshops. This initial exploration ensures that elicitation efforts are well-informed and focused.
- **Transcription:** Stakeholder interviews and workshops generate rich, qualitative data that must be captured accurately. AI-assisted transcription converts spoken dialogue and visual artifacts into structured, searchable formats while reducing manual effort and improving traceability. This helps to ensure that valuable insights are recorded and accessible for further analysis.
- **Extraction:** Once stakeholder input is documented, the next step is to extract clear and actionable requirements. AI can assist by identifying and categorizing functional requirements, quality attributes, and constraints from raw input such as meeting notes, emails, or transcripts. Grouping these requirements by topic or feature area further helps in clarifying priorities and dependencies.

Together, these activities constitute a comprehensive approach to elicitation, combining human expertise with AI support to enhance efficiency, coverage, and quality.

### 5.1.1 Exploring new domains

The exploratory phase of RE establishes the foundation for all subsequent RE activities by developing a comprehensive understanding of the domain, stakeholders, and operational context before formal requirements elicitation begins. AI can accelerate this critical phase by supporting research activities, broadening analytical perspectives, and structuring stakeholder engagement approaches, though their outputs serve only as starting points and require validation through human expertise and direct stakeholder interaction.

AI can summarize internal documentation and publicly available sources to provide domain overviews encompassing goals, terminology, challenges, and regulatory contexts, while also analyzing organizational structures for potential stakeholder groups that might otherwise be overlooked. It can generate structured interview guides with open-ended questions tailored for exploratory conversations and assist in organizing requirements workshops by identifying key topics, stakeholder roles, and potential areas of conflict or alignment. These capabilities demonstrate how AI can enhance Requirements Engineers' exploratory work by improving research efficiency and stakeholder coverage. Yet the true value depends on using these tools as catalysts for deeper inquiry, rather than as substitutes for the critical thinking and stakeholder engagement that are essential to establishing a shared understanding (CPRE Principle 3, [IREB2024]).

#### Identification of system context

AI can significantly improve the identification of the system context elements described in the CPRE Foundation Level Syllabus (CPRE Principle 4, [IREB2024] and [GleA2024]). Where artifacts like stakeholder lists, system landscape diagrams, process descriptions, or org charts are available, AI can extract the relevant information and classify this according to the five core aspects (stakeholders, documents, systems, processes, and events), as well as revealing relationships and dependencies across sources. This supports Requirements Engineers in forming a comprehensive view of system and context boundaries and reduces the risk of missing critical elements.

Alternatively, AI can generate structured question lists and checklists specifically designed to elicit information from stakeholders about each of the five system context aspects. These AI-generated instruments can serve as comprehensive elicitation tools during stakeholder interviews or workshops, ensuring systematic coverage of all context dimensions. The AI can tailor questions for specific domains or system types, incorporating best practices from requirements elicitation techniques (see chapter 4.2 in [GleA2024]) while maintaining consistency with established quality criteria for work products (see chapter 3.8 in [GleA2024]).

#### Establishing a shared understanding

The establishment of a shared understanding (CPRE Principle 3, [IREB2024]) fundamentally depends on the consistent use of terminology among all project participants. Glossaries (see chapter 3.5 in [GleA2024]) serve as central collections of definitions that mitigate the risk of misinterpretation and ensure that stakeholders, Requirements Engineers, and developers operate with the same conceptual foundation. AI can significantly accelerate the creation and maintenance of comprehensive glossaries by analyzing existing project documentation, stakeholder communications, and domain-specific materials to identify terms that require definition. AI can detect inconsistent usage of terminology across different documents, flag potential homonyms (same terms used for different concepts), and suggest synonyms that should be standardized. Additionally, AI can propose initial definitions based on contextual analysis of how terms are used within the project environment, providing Requirements Engineers with a solid starting point for glossary development.

### 5.1.2 Transcribing communication with stakeholders

Effective stakeholder communication forms the foundation of successful RE. Yet capturing and preserving insights from interviews and workshops can be a challenge if valuable information emerges in informal, unstructured formats. The conversion of spoken dialogue and visual artifacts into structured, editable documentation helps ensure that critical insights are retained and available for analysis, validation, and integration into formal RE processes. This supports traceability and team alignment, which are essential for a project's success.

AI-based transcription tools can significantly enhance this conversion process by generating transcripts from stakeholder interview recordings, cleaning raw text, and summarizing key discussion points.

This can accelerate the transition from conversation to formal requirements documentation. For collaborative workshops, AI can transcribe whiteboard content, sticky notes, and sketches into structured formats that facilitate sharing and collaborative refinement, particularly valuable in agile or distributed development environments. These capabilities support the creation of transparent and traceable requirements and reduce manual documentation effort. At the same time, AI-generated outputs must be carefully reviewed and refined by Requirements Engineers to ensure accuracy, completeness, and alignment with stakeholder intentions and project objectives.

#### Processing stakeholder interviews and audio recordings

Generative AI [FHJZ2024] can substantially enhance the efficiency of requirements elicitation through automated transcription of stakeholder interviews, transforming audio recordings into structured text that facilitates systematic analysis and documentation. This capability enables Requirements Engineers to focus entirely on the interview dynamics and stakeholder engagement during elicitation sessions (see chapter 4.2 in [GleA2024]), rather than dividing attention between active listening and note-taking. AI transcription services can identify different speakers, timestamp conversations, and even highlight potential requirements statements or key terminology that may warrant inclusion in project glossaries (see chapter 3.5 in [GleA2024]).

#### Analyzing transcribed interviews

Furthermore, AI can analyze transcribed interviews to extract preliminary requirements, identify stakeholder concerns, and suggest follow-up questions that align with systematic elicitation techniques, such as the questioning methods or collaboration techniques outlined in [GleA2024], 4.2.

### 5.1.3 Extracting requirements across sources

#### Analyzing emails, chats, meeting minutes, and collaboration threads

AI can systematically analyze emails, chats, meeting minutes, and collaboration threads to surface requirements-relevant content often missed in such informal channels. This automated extraction strengthens requirement source identification (see chapter 4.1 in [GleA2024]) by capturing implicit requirements, stakeholder concerns, and contextual cues from ongoing communications. AI can classify items by type (functional requirements, quality requirements, and constraints), detect conflicts between formal and informal statements, and flag inconsistencies with official documents. This is especially valuable in agile settings, where requirements evolve through continuous stakeholder interaction and informal feedback.

#### Analyzing existing documents and cross-referencing requirements

Furthermore, generative AI [FHJZ2024] can systematically analyze existing documents from various sources (see chapter 4.1 in [GleA2024]), including business process documentation, regulatory documents, legacy system specifications, and market analysis reports.

It can then identify both explicit and implicit requirements that may be embedded within this material. AI can process multiple document formats simultaneously and extract potential functional requirements, quality requirements, and constraints, categorizing them according to the requirements classification framework outlined in [GleA2024], chapter 1.1. AI can also cross-check extracted requirements against project scope and system boundaries to ensure their relevance. Furthermore, it can identify relationships among requirements found in different source documents. This automated extraction supports a comprehensive coverage of requirements sources beyond direct stakeholder input, helping to ensure that documented organizational knowledge and regulatory obligations are properly captured in the requirements specification.

#### Analyzing patterns and providing suggestions for further investigation

AI can analyze patterns across extracted stakeholder input to identify potential requirements gaps, implicit assumptions, or overlooked system aspects that stakeholders may have considered self-evident but failed to articulate explicitly. By comparing stakeholder statements against comprehensive requirements frameworks and domain-specific checklists, AI can suggest areas for further investigation and propose questions that help elicit subconscious requirements as described in the Kano model (see chapter 4.2 in [GleA2024]). AI can also identify inconsistencies among different stakeholder perspectives: these may indicate missing requirements or unresolved conflicts requiring attention through conflict resolution techniques (see chapter 4.3, in [GleA2024]). However, the Requirements Engineer must validate all AI-generated insights through systematic stakeholder engagement and verification processes. They must ensure that identified gaps are genuine requirements rather than AI assumptions; that these requirements do indeed represent actual stakeholder desires and needs (CPRE Principle 2, [IREB2024]) and not just a theoretical completeness.

## 5.2 Documentation

After domain exploration, transcription and extraction of requirements, the next challenge is to express them precisely and clearly. This involves two closely connected activities:

- **Formulation:** Formulation focuses on structuring requirements using standardized formats and quality criteria. Whether expressed as IEEE-style statements, use cases, or user stories, well-formulated requirements improve clarity, comparability, and traceability. AI tools can assist in selecting an appropriate format, refining the language, and in decomposing complex requirements into actionable sub-elements. Additionally, synonym tables generated by AI can help to maintain consistency across documentation, especially in domains with specialized terminology.
- **Transformation:** Transformation, on the other hand, ensures that requirements are understandable and usable across diverse stakeholder groups and technical contexts. This includes rewriting requirements in plain language, tailoring summaries for specific audiences, and converting between formats—such as generating models from text or vice versa.

AI also supports the creation of personas from stakeholder input, the generation of test cases from requirements, and even the comparison of specifications with source code to validate the implementation. Conversely, AI can help extract requirements from existing codebases, supporting documentation and modernization efforts.

Together, formulation and transformation bridge the gap between technical precision and stakeholder comprehension. They ensure that requirements are not only correct and complete, but also usable, testable, and aligned with business goals.

### 5.2.1 Formulating requirements

The formulation phase transforms raw stakeholder input into clear, structured requirement statements suitable for implementation and validation. AI tools can significantly enhance this process by supporting standardization, maintaining terminological consistency, and decomposing complex requirements into manageable components that align with established RE practices.

#### Applying standardized IEEE templates and phrase templates

Generative AI [FHJZ2024] can support Requirements Engineers using standardized IEEE templates and phrase templates (see chapter 3.3 in [GleA2024]) by automatically structuring requirement statements according to established syntactic patterns and formatting guidelines. AI can analyze informal stakeholder input or preliminary requirement descriptions and transform them into properly formatted individual requirements that follow IEEE standards such as ISO/IEC/IEEE 29148, ensuring consistent structure and improving the overall quality of the specification.

#### Eliminating common pitfalls

AI can suggest appropriate conditional clauses, specify measurable criteria, and, in technical writing, (see chapter 3.2 in [GleA2024]) can help eliminate common pitfalls such as passive voice, universal quantifiers, or incomplete descriptions. This automated formatting support enables Requirements Engineers to focus on content validation and stakeholder alignment rather than spending time on style corrections. It ensures that all requirements adhere to established template guidelines that promote clarity and consistency across the specification.

#### Formulating stakeholder input as user stories

AI can assist in formulating stakeholder input as well-structured user stories, following established phrase templates such as the widely used format "As a [user type], I want [goal] so that [benefit]" ([GleA2024], chapter 3.3). AI can analyze stakeholder descriptions, interview transcripts, or informal requirement statements to identify user roles, desired functionality, and business value, then structure this information according to user story conventions.

#### Suggesting acceptance criteria

AI can also suggest appropriate acceptance criteria, identify missing story elements, and ensure that user stories maintain focus on user value rather than implementation details. This capability proves particularly valuable in agile development contexts where user stories serve as the primary requirements artifacts in product backlogs (see chapter 3.6 in [GleA2024]). That way AI helps to maintain consistency in story format and completeness across large story collections.

#### Generating domain glossaries and synonym tables

By supporting the generation of domain glossaries and synonym tables, AI helps clarify domain-specific terminology and promote consistent language usage across teams, addressing ambiguity concerns found in natural-language-based work products (see chapter 3.2 in [GleA2024]).

#### Decomposing requirements

Additionally, AI can identify logical subdivisions within high-level requirements, helping decompose them into actionable sub-requirements, system-level specifications, or acceptance criteria that facilitate implementation alignment with business objectives. This formulation work builds upon domain understanding established during exploration activities, documented insights from transcription processes, and structured content from extraction phases. It ensures that requirements are expressed in formats that support effective collaboration, development, and validation throughout the RE process.

### 5.2.2 Transforming requirements into other representations

In modern RE, transformation is more than just translating text from one language to another: it is about transforming requirements across formats and stakeholder perspectives. As systems grow more complex and teams more interdisciplinary, the ability to transform requirements effectively becomes essential for clarity, alignment, and traceability.

#### Converting natural language requirements into visual models

AI tools support Requirements Engineers by enabling fluid movement between representations and stakeholder needs. They can convert natural language requirements into visual models such as UML or BPMN diagrams (see chapter 3.4 in [GleA2024]), making system structure, workflows, and relationships easier to understand than via text alone. Conversely, AI can describe visual models in structured text to improve accessibility for non-technical stakeholders and support documentation. Beyond general diagrams, AI can also generate specific UML artifacts such as class, activity, or use case diagrams, as well as BPMN workflows that visualize decisions and processes, always focusing on requirements specification rather than detailed design.

### Creating prototypes

Generative AI [FHJZ2024] can support Requirements Engineers in creating exploratory prototypes (see chapter 3.7 in [GleA2024]) by automatically generating wireframes, mock-ups, or even functional prototypes based on requirements specifications or user stories. AI can analyze functional requirements and user interface descriptions to produce visual representations that help stakeholders both to better understand proposed system behavior and to validate requirements through concrete examples. AI can create prototypes at different fidelity levels, from simple wireframes for early concept validation to more sophisticated mock-ups that demonstrate user interaction flows and interface concepts. This automated prototype generation enables rapid iteration and collection of stakeholder feedback, supporting the validation process (see chapter 4.4 in [GleA2024]) by providing tangible representations of abstract requirements that stakeholders can evaluate and refine.

### Creating dynamic visual content

AI offers diverse further applications for enhancing communication and validation in RE beyond core RE activities. These capabilities can improve stakeholder engagement and requirements quality when properly employed. From transforming requirements documentation into dynamic visual content such as explainer videos, to translating technical requirements into simplified language for non-technical stakeholders and generating targeted summaries for specific audiences (e.g., executives, end users, developers), AI can help ensure that information delivery is relevant for its target audience, while maintaining specification consistency.

### Creating user personas

AI technology can synthesize stakeholder input to create user personas that complement traditional stakeholder identification methods (see chapter 4.1 in [GleA2024]).

### Deriving test cases

AI can derive test cases from requirements to support validation activities (see chapter 4.4 in [GleA2024]).

### Analyzing legacy code

AI can analyze legacy code to extract implicit requirements for documentation efforts.

### Comparing specifications

AI can also compare specifications with implemented code to ensure compliance or identify discrepancies, particularly valuable in regulated industries requiring strict traceability of requirements.

## 5.3 Validation

Generative AI [FHJZ2024] faces inherent limitations when creating requirements that Requirements Engineers must recognize and address. AI may generate requirements that appear to be syntactically correct and follow established templates (see chapter 3.3 in [GleA2024]) but that do not in fact address genuine stakeholder desires and needs (CPRE Principle 2, [IREB2024], chapter 2) or that are not grounded in the actual system context (CPRE Principle 4, [IREB2024]). Furthermore, AI-generated requirements often lack the necessary precision and verifiability that characterize high-quality requirements, potentially introducing ambiguities or incomplete specifications that violate fundamental quality criteria. Requirements Engineers must remain vigilant for AI outputs that may seem comprehensive but actually miss critical functional requirements, quality requirements, or constraints, or that propose solutions rather than capturing actual requirements according to the Problem-Requirement-Solution principle (CPRE Principle 5, [IREB2024]).

### Maintaining consistency

AI can support Requirements Engineers in maintaining consistency across requirements specifications and product backlogs by automatically detecting contradictions, overlapping functionality, and terminology inconsistencies that may emerge as work products evolve over time. AI can cross-reference requirements against established glossaries (see chapter 3.5 in [GleA2024]) and identify instances where the same concepts are described using different terms. This supports the consistent use of terminology that is essential for shared understanding (CPRE Principle 3, [IREB2024]). AI tools can also analyze dependencies among requirements and flag potential conflicts that might compromise the overall coherence of the specification. However, ensuring true consistency requires more than automated pattern matching; it demands an understanding of the underlying business logic, stakeholder priorities, and system architecture. The Requirements Engineer must validate that AI-detected inconsistencies represent genuine problems rather than acceptable variations and must ensure that AI-suggested consistency improvements align with project goals and do not inadvertently introduce new conflicts or compromise the adequacy of individual requirements.

### Value-orientation, adequacy and necessity

The validation of AI-generated requirements demands rigorous assessment according to the value-orientation principle (CPRE Principle 1, [IREB2024]) to ensure that each proposed requirement contributes meaningfully to satisfying stakeholder desires and needs, rather than merely appearing reasonable in isolation. Requirements Engineers must evaluate whether AI-generated requirements are truly necessary by tracing them back to their stakeholder source (see chapter 4.1 in [GleA2024]) and verifying their alignment with identified business objectives and user goals. As for human-generated requirements, the focus should be on adequacy and necessity (see chapter 3.8 in [GleA2024]): checking that AI has not introduced requirements that exceed the actual scope of stakeholder needs or project constraints.

#### Abstraction level

The Requirements Engineer must also assess whether AI-generated requirements fit within the chosen abstraction levels (see chapter 3.1.2 in [GleA2024]) and are at an appropriate level of detail (see chapter 3.1.3 in [GleA2024]) for the current project phase. It should be ensured that the requirements serve the intended purpose rather than creating unnecessary complexity or scope creep that could compromise project success.

## 5.4 Management

AI can support two important tasks of requirements management: *assignment of attributes to requirements and requirements prioritization*.

### 5.4.1 Assignment of attributes to requirements

AI can streamline the assignment of attributes to requirements by analyzing their content and proposing values based on established patterns and project-specific criteria. As outlined in [GleA2024] chapter 6.5, attributes document important metadata that enables stakeholders to access relevant information throughout the project lifecycle. Through textual analysis and comparison with similar requirements from current or previous projects, AI can also suggest values for common attributes such as priority, complexity, source, or responsible person. Missing attribute assignments can also be identified, and completeness can be ensured across large requirement sets, thereby supporting the systematic documentation practices essential for effective requirements management (see chapter 6.1 in [GleA2024]).

### 5.4.2 Requirements prioritization

Generative AI [FHJZ2024] can assist Requirements Engineers in prioritizing requirements (see chapter 6.8 in [GleA2024]), analyzing multiple criteria simultaneously and proposing rankings based on an evaluation of business value, urgency, effort, dependencies, and other relevant factors. AI can process a large quantity of requirements and stakeholder input to identify patterns and trade-offs that might be overlooked during manual prioritization, while ensuring systematic consideration of all prioritization steps outlined in [GleA2024] chapter 6.8. AI can also generate comparative analyses among requirements, highlighting potential conflicts or dependencies that might influence prioritization decisions. Furthermore, it can adapt prioritization suggestions based on different stakeholder perspectives or changing project constraints.

Effective prioritization extends far beyond algorithmic analysis and requires a deep understanding of stakeholder needs, business strategy, and project context—elements that AI cannot fully comprehend., Involving appropriate stakeholders in priority decisions remains essential (see chapter 6.8 in [GleA2024]), and AI suggestions must be validated against these judgments and strategic objectives. To maintain alignment with the value-orientation principle (CPRE Principle 1, [IREB2024]), prioritization must reflect genuine stakeholder needs rather than abstract metrics.

In iterative development, the dynamic nature of prioritization further demands ongoing human oversight to ensure AI recommendations remain relevant as project conditions evolve.

## 5.5 Key takeaways

AI can enhance all core RE activities—elicitation, documentation, validation, and requirements management—by improving efficiency, structure, and consistency. It should be seen as a supportive tool that accelerates tasks but does not replace human expertise. Requirements Engineers must curate AI outputs to ensure they are correct, relevant, and aligned with project goals. Ultimately, AI serves as an aid to human judgment, while responsibility for requirements quality remains with the Requirements Engineer.

# 6 AI Terminology

Level: L1

Goal: The candidate knows terminology which is essential for working with AI.

### Educational objectives

EO 6.1 The candidate can remember and identify core terminology essential for working with AI. (L1)

This chapter explains key terminology that is essential for understanding and working with AI, especially large language models.

### Chatbot

A chatbot is an AI system designed to simulate human conversation through text or speech [ZhWX2023]. Modern chatbots based on LLMs can understand context, maintain dialogue history, and generate coherent, context-aware responses. In RE, chatbots can assist Requirements Engineers by transcribing stakeholder interviews, retrieving information from documents, or supporting documentation tasks through interactive dialogue.

### Chain-of-thought reasoning

Chain-of-thought reasoning is an approach in which the LLM is prompted to explain its reasoning step-by-step before providing the final answer [WeeA2022]. In RE, this can support transparency and traceability, for example, when analyzing conflicting requirements or evaluating design alternatives.

### Computer vision

Computer vision transforms raw pixels into structured facts [Szel2022]. Modern vision networks detect objects, read text from an image and estimate spatial depth. The best-known example is probably the face-unlocking feature of modern phones.

### Context window

The context window defines the maximum number of tokens an LLM can consider at one time (including both the prompt and the generated output) [BreA2020]. This limit constrains how much of a requirements specification or dialogue history can be taken into account when generating responses. For RE tasks, such as validating requirements or analyzing stakeholder feedback, awareness of the context window size helps in structuring inputs efficiently. However, this is not always a "bigger is better" thing: the more input you give an LLM, the better it is at the start, but too much input can also cause the model to get distracted.

### Deep learning

Deep learning is a subset of Machine learning that uses neural networks (such as transformers for language and convolutional networks for images) to handle complex data types and tasks, including reinforcement learning [GoBC2016].

### Embeddings

Embeddings are numerical vector representations of text that capture semantic meaning [SiKe2024]. In RE, embeddings enable semantic search, duplicate detection, clustering, and traceability between requirements and related artefacts—even when the wording differs. Embeddings form the basis for many retrieval and analysis functions but do not generate text themselves.

### Expert systems

Expert systems capture human expertise as explicit rules and ontologies, producing auditable, deterministic answers [Bari2025]. They are still favored for domains like tax law or medical triage, where transparency trumps probability and absolutely no room for errors is allowed. Modern tax software can, for example, utilize AI to analyze your bills, income statements and other data to calculate the amount of tax that you owe.

### Fine-tuning

Fine-tuning is the process of adapting a pre-trained LLM to a specific domain or task by training it on additional, relevant data [HoRu2018]. For RE tasks, fine-tuning can align the model with domain terminology, regulatory language, or organizational standards, thereby improving the precision of generated requirements or compliance checks. Fine-tuning requires effort, domain expertise, and quality-controlled data to be effective.

### Few-shot / zero-shot learning

Few-shot and zero-shot learning are techniques in which the model performs a task with few or no examples in the prompt [BreA2020], [DAIR2024]. In RE, these can be used to classify requirements, generate acceptance criteria, or rephrase specifications without extensive task-specific training data.

### Hallucination

Hallucination is a direct consequence of the probabilistic mechanism described in chapter 1.1 [FDA1997] FDA 21 CFR Part 11: U.S. Food and Drug Administration. Title 21 Code of Federal Regulations Part 11: Electronic Records; Electronic Signatures. U.S. Government Publishing Office, 1997. https://www.ecfr.gov/current/title-21/chapter-I/subchapter-A/part-11 Last visited November 2025.

[FHJZ2024]. A hallucination occurs when an LLM produces statistically likely text that is factually incorrect or unsupported by source data. In RE, hallucinations can be particularly problematic if fabricated details are mistaken for stakeholder statements or domain facts. Requirements Engineers using LLMs must therefore validate AI-generated content against authoritative sources, in line with the CPRE principle 6 that non-validated requirements are useless [IREB2024]. All AI-generated output must be treated as an unverified draft—something also required by the EU AI Act.

### Machine learning (ML)

In Machine learning algorithms learn patterns directly from data [Bish2006]. Classic ML predicts numbers or categories; deep-learning networks such as transformers and

convolutional neural networks handle language, images and reinforcement-learning policies. The output quality rises with training-set size and relevance. For example, a model could take past house sales data and predict prices for properties to be sold.

### Natural language processing (NLP)

Natural language processing includes everything that turns speech or text into meaning and back again [QieA2020]. It ranges from rule-based parsers to today's LLM chatbots plus speech-to-text and text-to-speech modules.

### Prompt

A prompt is the input provided to an LLM to guide its output [Kuka2024]. It may be a question, instruction, or partial text. In RE practice, prompts determine the relevance and accuracy of generated outputs, for example, when asking an LLM to propose alternative formulations of a requirement or to extract functional requirements from stakeholder interviews. The quality of the prompt has a direct impact on the usefulness of the response.

### Prompt engineering

Prompt engineering is the practice of designing and refining prompts to reliably achieve desired outputs from an LLM [CZLZ2025]. This is akin to crafting precise interview questions or template structures in RE: clarity and structure improve the quality of the information obtained.

### Retrieval-augmented generation (RAG)

Retrieval-augmented generation (RAG) is a process that uses embeddings to find relevant documents and then passes them into an LLM to generate an answer [LeeA2020]. In RE, RAG ensures AI outputs are based on actual specifications, stakeholder notes, or regulatory documents, reducing the risk of hallucinations and keeping results project-specific rather than based purely on the model's own internal training data.

### Robotics

Robotics combines sensor fusion, mapping, path-planning and feedback control so machines can navigate and manipulate the physical world [Szel2022]. Applications range from warehouse pickers and autonomous drones to vacuum cleaners that map out your living room and can avoid obstacles.

### System prompt / instruction

A system prompt is a special type of prompt that sets the overarching behavior or role of the LLM (e.g., "You are an assistant Requirements Engineer…") [CZLZ2025]. This ensures that generated outputs follow organizational style guides or RE best practices.

### Temperature

Temperature is a parameter that controls the randomness of an LLM's output [BreA2020]. Low values make the model more deterministic, producing consistent phrasing, while higher values create more varied and creative results. In RE, lower temperatures are often preferable for consistency in requirements wording, whereas higher values can be useful for

generating diverse solution ideas. However, in many LLMs temperature cannot be set by the users.

### Token

A token is a unit of text—often a word, part of a word, or punctuation—that the model processes [SeHB2016]. LLMs operate on tokens rather than entire sentences, and their performance and cost are partly determined by the number of tokens processed. In RE contexts, token limits can affect how much specification text, stakeholder input, or glossary content can be processed in a single interaction.

### Transformer

Transformers are the neural network architecture behind most modern LLMs [VaeA2017]. It uses an attention mechanism to assess the importance of words in relation to each other, regardless of position, enabling efficient handling of long sequences and complex language patterns. In RE, transformers allow AI tools to interpret and generate requirements text with high contextual awareness, for example identifying dependencies or maintaining consistent terminology across documents.

### Vector

A mathematical representation of data as an ordered list of numbers [MCCD2013]. In the context of embeddings, vectors capture the semantic meaning of text by placing similar words, sentences, or requirements close to each other in a multi-dimensional space. In RE, vectors enable operations such as similarity search, clustering of related requirements, or linking stakeholder needs with existing documentation, even when different wording is used.

# 7 References


[Bari2025] Barisic, Davor: Exploring the Landscape of Expert Systems: A Review. International Journal of Management Trends: Key Concepts and Research, 4(1), 58–68, 2025.

[Bish2006] Bishop, Christopher M.: Pattern Recognition and Machine Learning. Springer, 2006.

[BreA2020] Brown, Tom et al.: "Language models are few-shot learners." Advances in Neural Information Processing Systems 33 (2020): 1877–1901.

[CZLZ2025] Chen, Banghao, Zhaofeng Zhang, Nicolas Langrené, and Shengxin Zhu: Unleashing the Potential of Prompt Engineering for Large Language Models. Patterns 6, no. 6 (2025).

[DAIR2024] DAIR.AI. Zero-Shot Prompting. Prompting Guide, 2024. https://www.promptingguide.ai/de/techniques/zeroshot. Last visited October 2025.

[FDA1997] FDA 21 CFR Part 11: U.S. Food and Drug Administration. Title 21 Code of Federal Regulations Part 11: Electronic Records; Electronic Signatures. U.S. Government Publishing Office, 1997. https://www.ecfr.gov/current/title-21/chapter-I/subchapter-A/part-11 Last visited November 2025.

[FHJZ2024] Feuerriegel, Stefan, Jonas Hartmann, Christian Janiesch, and Patrick Zschech: Generative AI. Business & Information Systems Engineering vol. 66(1), pages 111–126, February 2024.

[GDPR2016] European Union. General Data Protection Regulation (EU) 2016/679. Official Journal of the European Union, 2016. https://eurlex.europa.eu/eli/reg/2016/679/oj Last visited November 2025.

[GleA2024] M. Glinz et al.: Handbook for the CPRE Foundation Level according to the IREB Standard (Version 1.2.0). IREB e.V., Karlsruhe, 2024. https://cpre.ireb.org/en/downloads-and-resources/downloads#cpre-foundation-level-handbook. Last visited September 2025.

[Glin2024] M. Glinz: A Glossary of Requirements Engineering Terminology (Version 2.1.0). IREB e.V., Karlsruhe, 2024. https://cpre.ireb.org/en/downloads-and-resources/downloads#cpre-glossary . Last visited September 2025.

[GoBC2016] Goodfellow, Ian, Yoshua Bengio, and Aaron Courville: Deep Learning. MIT Press, 2016.

[HoRu2018] Howard, Jeremy, and Sebastian Ruder. Universal Language Model Fine-tuning for Text Classification. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL), 2018.

[IEC62304] International Electrotechnical Commission. IEC 62304: Medical Device Software—Software Life Cycle Processes. IEC, 2006.

https://webstore.iec.ch/en/publication/6792 (Note: Confirmed as current in 2021) . Last visited October 2025.

[IREB2025] #AIREB Booklet on RE and AI. Retrieved from https://cpre.ireb.org/en/downloads-and-resources/downloads#ai-snippets Last visited October 2025.

[IREB2024] IREB Certified Professional for Requirements Engineering Foundation Level Syllabus (version 3.2.0). IREB e.V., Karlsruhe, 2024. https://cpre.ireb.org/en/downloads-and-resources/downloads#cpre-foundation-level-syllabus. Last visited September 2025.

[Kuka2024] Kuka, V. (2024, October 22). Prompt Engineering, https://learnprompting.org/docs/basics/prompt_engineering. Last visited October 2025.

[LeeA2020] Lewis, Patrick et al.: "Retrieval-augmented generation for knowledge-intensive NLP tasks." Advances in Neural Information Processing Systems 33 (2020): 9459-9474.

[MiSC2013] Mikolov, T., Sutskever, I., Chen, K., et al. (2013). "Distributed representations of words and phrases and their compositionality." Advances in Neural Information Processing. https://proceedings.neurips.cc/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html. Last visited October 2025.

[MCCD2013] T. Mikolov, K. Chen, G. Corrado, J. Dean: Efficient Estimation of Word Representations in Vector Space. 2013. arXiv:1301.3781 (2013).

[NoeA2024] Norheim, J.J., Rebentisch, E., Xiao, D., Draeger, L., et al. (2024). "Challenges in applying large language models to requirements engineering tasks." Design Science https://www.cambridge.org/core/journals/design-science/article/challenges-in-applying-large-language-models-to-requirements-engineering-tasks/1FC7666F0A0B4E7091D2D4B2D46321B5. Last visited October 2025.

[QieA2020] Qiu, Xipeng et al.: Pre-trained Models for Natural Language Processing: A Survey. Science China Technological Sciences vol. 63, no. 10 (2020): 1872-1897.

[RaeA2019] Radford, Alec e. al: "Language models are unsupervised multitask learners." OpenAI blog 1, no. 8 (2019): 9. https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf. Last visited October 2025.

[SceA2023] Schick, T., Dwivedi-Yu, J., Dessì, R., et al. (2023). "Toolformer: Language models can teach themselves to use tools." Advances in Neural Information Processing Systems. https://proceedings.neurips.cc/paper_files/paper/2023/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html. Last visited October 2025.

[SeHB2016] Sennrich, Rico, Barry Haddow, and Alexandra Birch: Neural Machine Translation of Rare Words with Subword Units. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 1715–1725. Association for Computational Linguistics, Berlin, Germany, 2016.

[SiKe2024] Siebert, Jan, and Patrick Kelbert. How do LLMs work? A look inside large language models. Fraunhofer IESE Blog, 2024 https://www.iese.fraunhofer.de/blog/wie-funktionieren-llms/. Last visited October 2025.

[Szel2022] Szeliski, Richard: Computer Vision: Algorithms and Applications (2nd ed.). Springer Nature, 2022.

[VaeA2017] Vaswani, Ashish et al.: "Attention is All you Need." In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (eds.), Advances in Neural Information Processing Systems 30 (NeurIPS 2017). Curran Associates, 2017.

[VoFi2025] A. Vogelsang, J. Fischbach (2025). Using Large Language Models for Natural Language Processing Tasks in Requirements Engineering: A Systematic Guideline. In: A. Ferrari, G. Ginde (eds.) Handbook on Natural Language Processing for Requirements Engineering. Cham: Springer Nature. Preprint available for free at https://arxiv.org/pdf/2402.13823

[VoFi2025] A. Vogelsang, J. Fischbach: Using Large Language Models for Natural Language Processing Tasks in Requirements Engineering: A Systematic Guideline, 2025. In: A. Ferrari, G. Ginde (eds.) Handbook on Natural Language Processing for Requirements Engineering. Cham: Springer Nature. Preprint available at https://arxiv.org/pdf/2402.13823.

[WeeA2022] Wei, Jason et al.: "Chain-of-thought prompting elicits reasoning in large language models." Advances in Neural Information Processing Systems 35 (2022): 24824–24837.

[WheA2023] White, Jules et al.: "A prompt pattern catalog to enhance prompt engineering with chatgpt." arXiv preprint arXiv:2302.11382 (2023).

[YeeA2023] Ye, H., Liu, T., Zhang, A., Hua, W., Jia, W. (2023). "Cognitive mirage: A review of hallucinations in large language models." arXiv preprint arXiv:2309.06794

[ZaDA2025] Zadenoori, M.A., Dąbrowski, J., Alhoshan, W., et al. (2025). "Large Language Models (LLMs) for Requirements Engineering (RE): A Systematic Literature Review." arXiv preprint arXiv:2509.11446

[ZhWX2023] Zhao, Wayne, Yuan Wang, Xing Xie, et al.: A Survey of Large Language Models. 2023. arXiv:2303.18223 (2023).