GRADION
scaling business

From Vibe coding to AI assisted spec driven development

Jan Moser
Principal Consultant at GRADION

January 2026

Goals

GRADION
scaling business

Takeaways

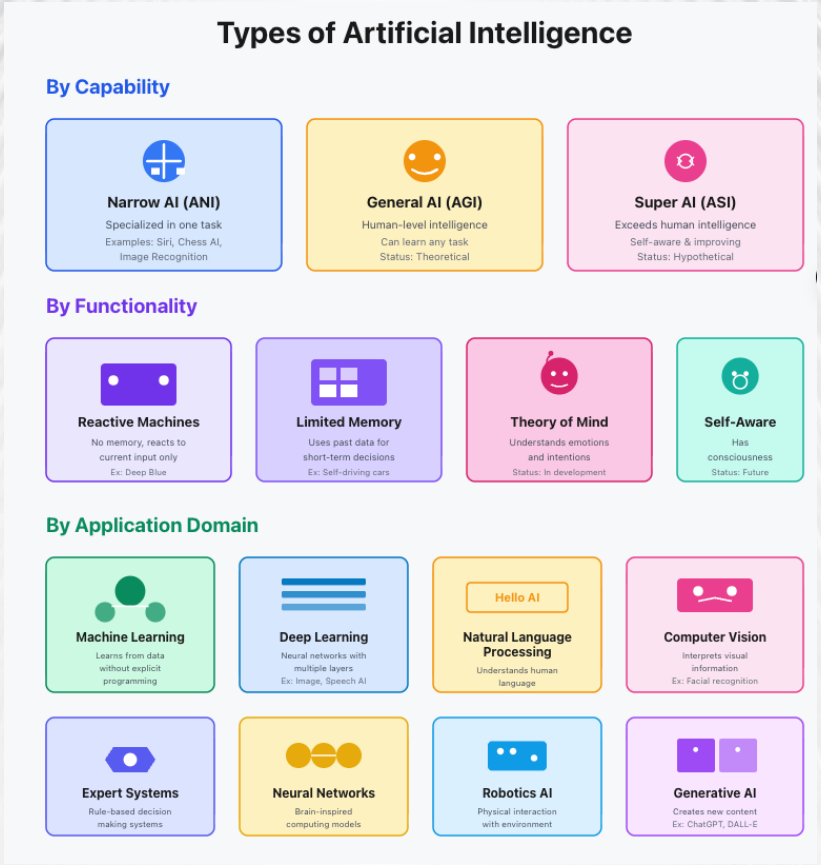
- Understanding what Vibe Coding means
- Understanding the basic elements of LLMs and their attributes
- Understanding the importance of context
- Understand what Spec Driven Development is
- Get an impression about how coding this way looks
- Get an overview over the tooling landscape (January 2026)

GRADION
scaling business

AI 101

The different types of AI

GRADION
scaling business



A typical LLM Agent

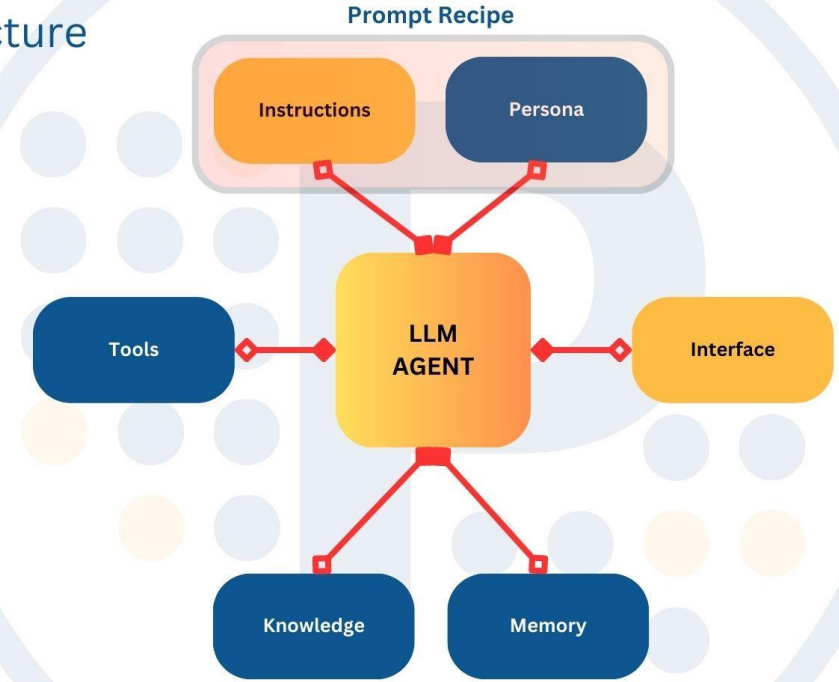
GRADION
scaling business

Typical **LLM Agent** Structure

- Mandatory Component
- Optional Component

- Prompt Recipe guides how the agent will proceed with the task and how to process the output
- Agent must generally interface with a Human, another agent or an API
- Agent can generate "memories" as well has access to specific domain knowledge and tools

PromptEngineering.org



The currency of AI Models

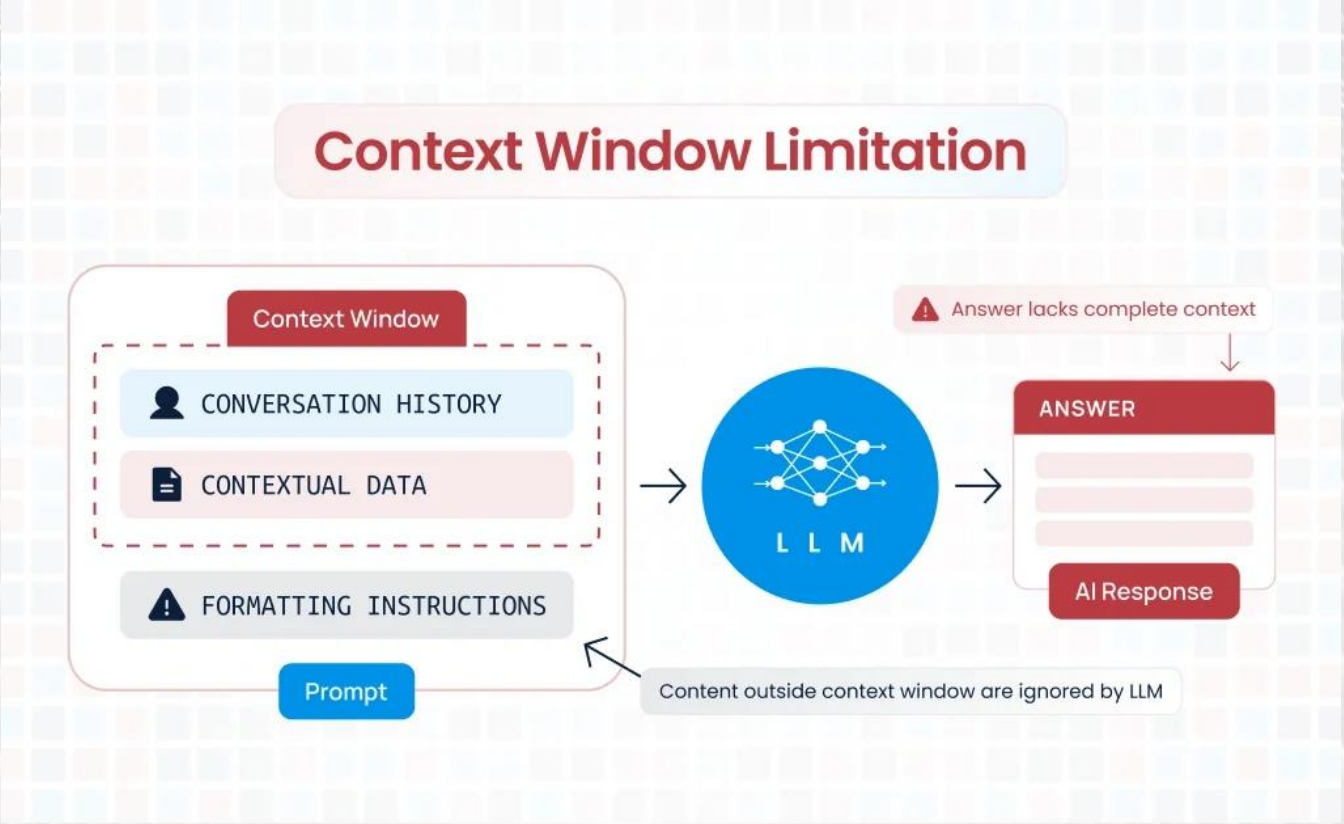
GRADION
scaling business

Tokens	Characters
94	382

We're no strangers to love
You know the rules and so do I (do I)
A full commitment's what I'm thinking of
You wouldn't get this from any other guy
I just wanna tell you how I'm feeling
Gotta make you understand
Never gonna give you up
Never gonna let you down
Never gonna run around and desert you
Never gonna make you cry
Never gonna say goodbye
Never gonna tell a lie and hurt you

Text Token IDs

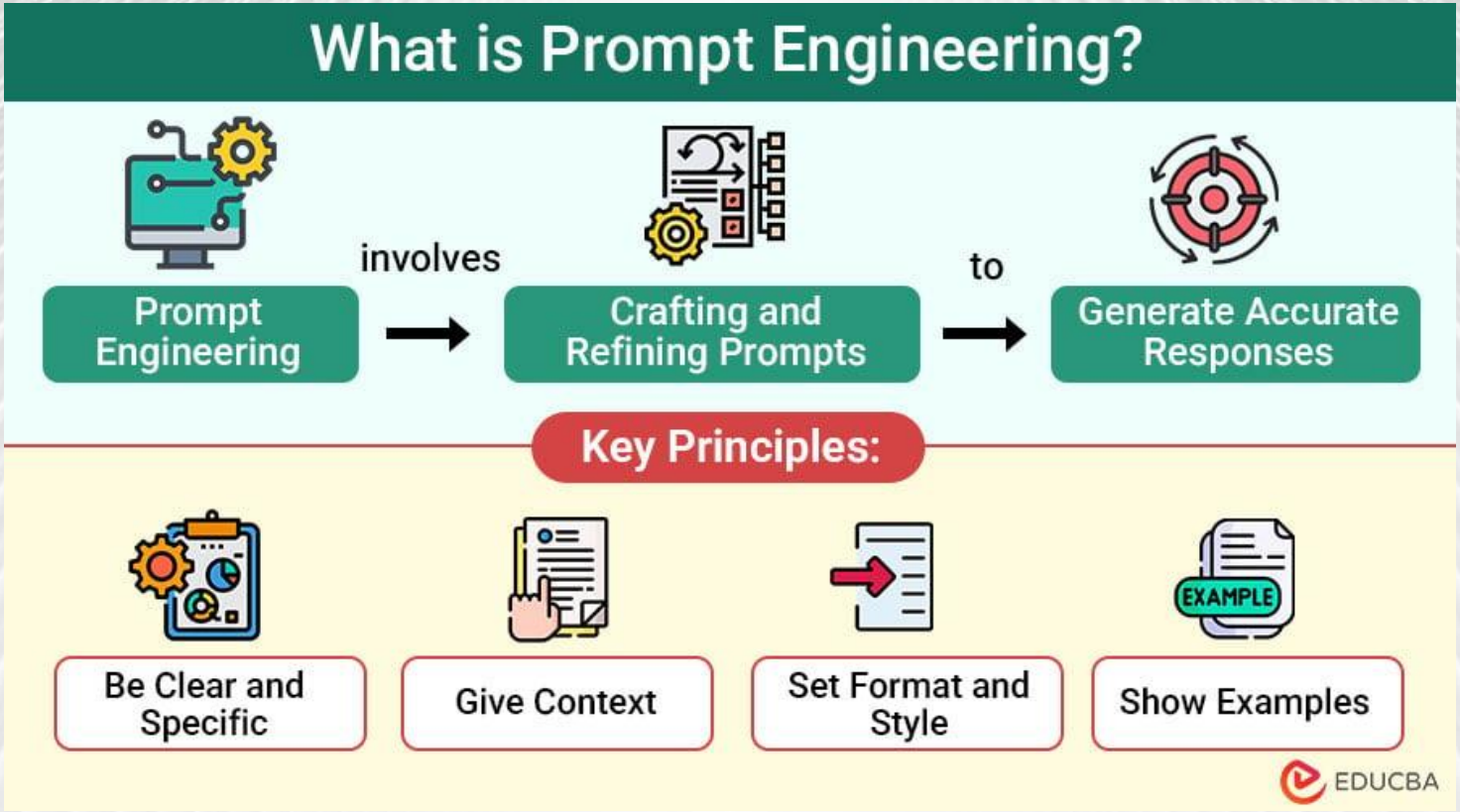
The context window



GRADION
scaling business

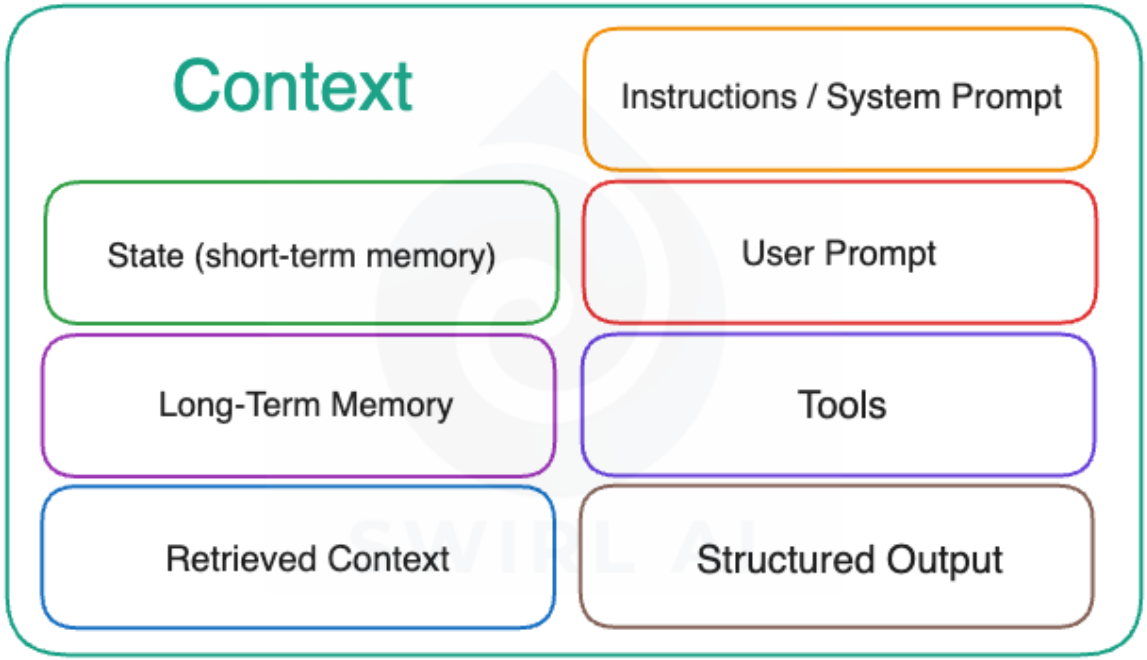
Prompting 101

Steps of prompting



Elements of LLM Context

GRADION
scaling business



Summary

GRADION
scaling business

1.System Prompt (Instructional Context)

1. Defines the *AI's role, behavior, tone, and constraints*.
2. Often invisible to the user but sets foundational rules (e.g., "You are an expert technical assistant").
3. Guides reasoning style, priority, and how user input is interpreted.

2.User Prompt (Input Context)

1. The *active request or query* from the user.
2. Includes explicit instructions ("Explain…", "Write…") and implicit signals (style, tone, domain).
3. Primary driver of output — quality depends on clarity and structure.

1.Knowledge Context (Static or Retrieved Information)

1. Background information available to the model (training data, embeddings, external retrieval sources).
2. Can include references, documents, or structured data fetched through RAG or tools.

2.Memory Context (Dynamic/Session State)

1. Represents *conversation history* and *saved facts about the user or session*.
2. Enables continuity across turns (e.g., referring back to previous answers or user preferences).
3. Can be short-term (within one chat) or long-term (persistent identity, goals, data).

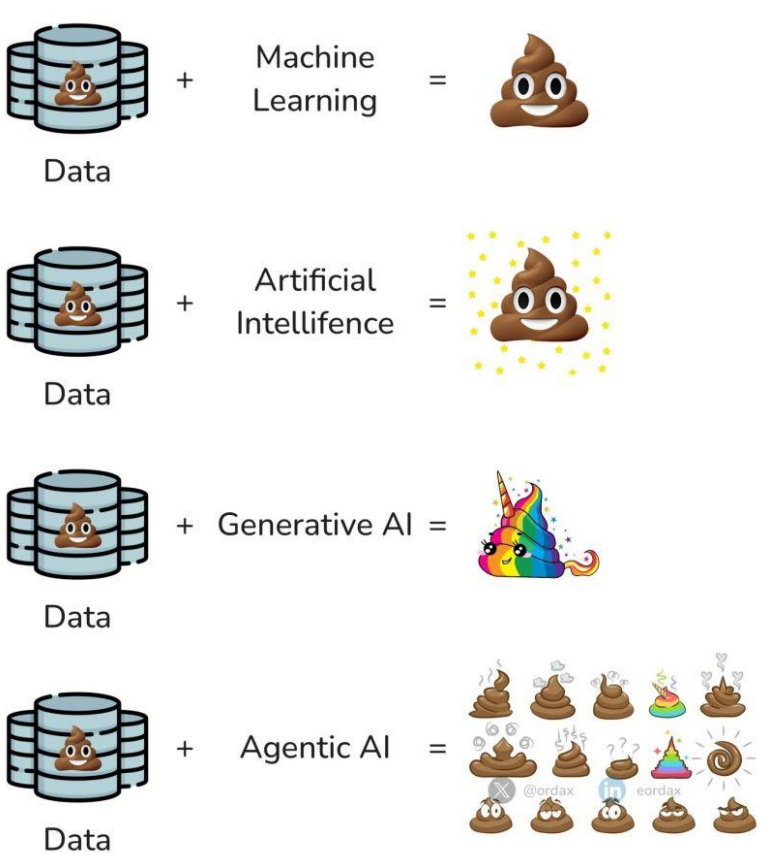
Summary

GRADION
soaling business

Elements of a Good Prompt

- **Clear Intent**
  Specify exactly what you want the model to do (e.g., summarize, analyze, compare).
- **Defined Role/Voice**
  Assign a perspective or persona (“Act as a security architect…”).
- **Contextual Background**
  Provide relevant details or examples to narrow scope.
- **Output Format**
  Define structure or constraints (e.g., bullet points, table, JSON).
- **Tone and Style Guidance**
  Indicate desired tone (technical, conversational, formal, etc.).
- **Evaluation or Goal Criteria**
  Mention what success looks like (“focus on trade-offs between…”).
- **Progressive Refinement**
  Encourage iterative clarification (“you can ask follow-up questions if unclear”).

Summary


Data
+
Machine Learning
=
Data
+
Artificial Intellifence
=
Data
+
Generative AI
=
Data
+
Agentic AI
=
@ordax
eordax

GRADION
soaling business

Vibe coding

What is Vibe Coding

GRADION
scaling business

- Vibe coding is an AI-assisted software development approach where developers describe functionality in plain language and a large language model generates the code.
- Rather than carefully crafting and reviewing each line, the human primarily steers the system through prompts and feedback loops.
- This shifts the focus from low-level implementation details to the overall behavior and feel of the application.
- The term was popularized in 2025 to describe this conversational, AI-centric development style.

What is Vibe Coding

GRADION
soaling business

- Natural-language prompts are the primary “interface,” replacing most direct editing of source code.
- A large language model generates and iteratively refines the code based on conversational feedback.
- The human focuses on goals, UX and outcomes, often treating the AI like a coding agent or teammate.
- Code is often accepted with minimal manual reading, relying instead on tests, execution, and further prompting to correct issues.
- It lowers the barrier to entry for non-experts by abstracting away syntax and many traditional programming details.

Vibe coding issues

GRADION
scaling business

Account Verification

We have just sent the code **435841** to your phone number: xxx-xxx-8247

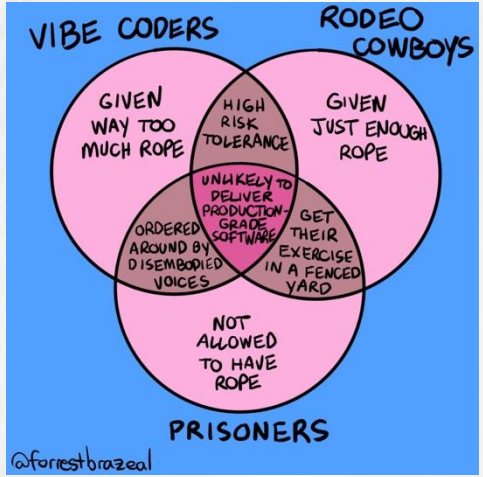
Please enter the code below to access your account:

Developers in 2020:

```
function isOdd(num) {
  if (num === 0) return false;
  if (num === 1) return true;
  if (num === 2) return false;
  if (num === 3) return true;
  if (num === 4) return false;
  if (num === 5) return true;
  if (num === 6) return false;
```

Developers in 2025:

```
function isOdd(num) {
  const response = OpenAI.prompt(`Is ${num} odd?`);
  return response.content;
}
```



GRADION
soaling business

Spec driven development

Spec driven development

GRADION
soaling business

- Spec driven development is a software methodology where detailed, formal specifications are created first and then used as the single source of truth for generating code, tests, and documentation.

- In modern AI workflows, these specs are written for both humans and coding agents, which consume them to produce and evolve the implementation.

- This is nothing new… Spec driven development is around since the early 2000s

Structured input

GRADION
scaling business

- Structured input is key for enterprise ready AI solutions
- By creating specifications in markdown-format you can ensure a sustainable, and reproduceable quality and output

- The DSPI workflow provides a structured, spec driven approach to AI assisted coding.
- It splits the development lifecycle in 4 phases.
- Those are then worked off in a sequential order with human-in-the loop reviews.
- The phases are **D**iscovery → **S**pecification → **P**lanning → **I**mplementation

DSPI Flow

GRADION
scaling business

Discovery: Understand the system and define the feature. Extract global specs and write a clear story set describing intent, value, and boundaries.

Specification: Convert the story into precise, technology-agnostic specifications split by type: business logic, data model, API contract, and UI design. Resolve open questions early and link decisions back to the story.

Planning : Translate specifications into an implementation plan: phases, tasks, testing strategy, risks, and integration notes. Every task maps to a spec, making scope and success measurable.

Implementation: Execute the plan iteratively. Keep documentation and code synchronized; validate with automated checks and human review. When tests pass and specs are satisfied, the feature is complete.

Caveats

GRADION
soaling business

- The creation of meaningful spec files is a challenge, time consuming and needs to be practised.
- Creation of the artefacts is not a 3min task usually…you won’t get magically 10x faster or better.
- Context is your enemy. The bigger your spec files grow, the more likely you run into issues (most models start to significantly decrease in accuracy and performance after only 10% usage of their context window)..use helpers such as Claude skills, or subagents or use small fragmented context files, that you manually then add to the context.
- Spec driven development is not token-friendly at all and can become quite expensive quite fast unless you use local models

Summary

GRADION
scaling business

- Spec driven development (SDD) provides in my opinion a future-proof way of AI assisted coding
- Unless the Vibe coding mentality it requires people to have in-depth understanding of domain, specifications and code
- In the optimal scenario your spec is the only source of truth. To change code, you update the appropriate spec file
- A meaningful context handling is essential for getting good results
- The process will not make you a 10x employee…if you are really good, maybe 2-3x is possible
- Depending on your spec files and llm provider this is not a cheap thing to do unless you use self-hosted models
- There are several tools out there to support you such as Specs CLI, GitHub SpecKit or Kiro from AWS (I used Speckit from Github for the demo)

List of useful links

GRADION
scaling business

- https://promptengineering.org/ *(Good guide to get tips and tricks for prompt and context engineering)*
- https://github.com/github/spec-kit *(Github SpecKit, a framework for Spec Driven AI dev)*
- https://github.com/anthropics/claude-code/tree/main/plugins/ralph-wiggum (*Ralph Wiggum is a plugin by Anthropic for Claude Code to improve and assess generated code)*
- https://platform.claude.com/docs/en/agents-and-tools/agent-skills/overview *(Introduction of Claude Agent Skills)*
- https://code.claude.com/docs/en/sub-agents *(Introduction to Claude subagents)*
- https://github.com/specs-cli/specs-cli *(Another Spec Kit….the readme has a nice explanation of the DSPI workflow)*
- https://roocode.com/ *(Persona-based Agentic AI coding assistant, can also connect via Ollama to local models)*
- https://www.vischer.com/en/artificial-intelligence/ *(a very good blog series about legal aspects of AI)*

Thanks for your interest! And in case you want to know more, feel free to contact me!

Jan Moser
jan.moser@gradion.com