# Effectieve Prompt Engineering voor Programmeurs

## Wat Maakt een Prompt Effectief?

**Duidelijkheid en Specificiteit:** Een effectieve prompt laat geen ruimte voor misinterpretatie. Dit betekent dat je de AI **expliciet context, instructies en beperkingen meegeeft**. Vermijd vage of ambiguë verzoeken – beschrijf in plaats daarvan duidelijk het probleem en wat je van de AI verwacht[[1]](https://documentation.suse.com/suse-ai/1.0/html/AI-preventing-hallucinations/index.html#:~:text=The%20most%20common%20causes%20of,hallucinations%20are)[[2]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=,different%20from%20what%20you%20desire). Uit best practices blijkt dat goede prompts context bieden over het systeem of de code, het gewenste outputformaat specificeren en relevante vereisten of constraints benoemen[[3]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=Before%20diving%20into%20specific%20prompts%2C,effective%20prompts%20share%20certain%20characteristics). Bijvoorbeeld, in plaats van “Geef wat goede testcases”, kun je beter schrijven: *“Genereer 5 testcases voor een REST API die gebruiker-authenticatie afhandelt, inclusief randgevallen voor ongeldige credentials, token-verloop en rate limiting. Formatteer elke testcase met precondities, stappen, verwachte response-codes en validatiechecks.”* Zo’n specifieke prompt levert veel gedetailleerdere en bruikbare resultaten[[4]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=When%20a%20prompt%20tester%20experiments,useful%20and%20detailed%20test%20cases)[[5]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=massively,useful%20and%20detailed%20test%20cases). Het kost wellicht wat meer moeite om een prompt op te stellen, maar die investering betaalt zich terug in nauwkeurigheid van de output – *“2 minuten extra besteden aan promptformulering kan je een uur aan heen-en-weer gespar met de AI schelen,”* aldus een praktijkgids voor test-automatiseerders[[6]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=Be%20Specific%20and%20Detailed)[[7]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=,to%20you%3F%20Don%E2%80%99t%20be%20lazy).

**Structuur en Opmaak:** Structuur aanbrengen in je prompt helpt de AI om de onderdelen beter te begrijpen. Ervaren prompt-engineers adviseren om secties of onderdelen duidelijk te scheiden, bijvoorbeeld door opsommingstekens, kopjes of zelfs opmaak zoals Markdown of XML-tags te gebruiken[[8]](https://www.ssw.com.au/rules/ai-prompt-xml/#:~:text=understand). Door delen van de prompt logisch te groeperen (bv. *Context*, *Data*, *Instructies*, *Uitvoer-formaat*), verminder je de kans dat belangrijke details verloren gaan in de massa tekst[[9]](https://www.ssw.com.au/rules/ai-prompt-xml/#:~:text=Why%20use%20Markdown%20or%20XML,tags). Een officiële richtlijn stelt: *“Als het in één oogopslag moeilijk leesbaar is voor mensen, zal de AI er ook moeite mee hebben”*[[10]](https://www.ssw.com.au/rules/ai-prompt-xml/#:~:text=struggle%20too). Markdown-opmaak met kopjes of lijsten is een mensvriendelijke manier om een prompt te structureren, en helpt het model om jouw bedoelingen segment per segment te parsen[[8]](https://www.ssw.com.au/rules/ai-prompt-xml/#:~:text=understand)[[11]](https://www.ssw.com.au/rules/ai-prompt-xml/#:~:text=%2A%20Clarity%20,prompt%20by%20having%20everything%20atomic). Alternatief kun je XML-achtige tags gebruiken om onderdelen af te bakenen; dit biedt de AI expliciete grenzen tussen secties. Inderdaad, **gestructureerde prompts leveren aantoonbaar betere resultaten** – in experimenten bleken zowel Markdown- als XML-geformatteerde prompts tot hogere success rates te leiden dan ongestructureerde tekst, terwijl bijvoorbeeld JSON-format minder effectief was[[12][13]](https://medium.com/@isaiahdupree33/optimal-prompt-formats-for-llms-xml-vs-markdown-performance-insights-cef650b856db#:~:text=,negative%20score%20across%20the%20board). Het mooie is dat je hiervoor geen formele schema’s hoeft te volgen: *“Het helpt zowel mens als AI om een duidelijke structuur te hebben,”* zelfs als de tags puur illustratief zijn[[14]](https://www.ssw.com.au/rules/ai-prompt-xml/#:~:text=Using%20Markdown%20or%20XML%20tags,quality%20outputs). Recent is gebleken dat met name Anthropic’s Claude-model deze aanpak waardeert. Claude’s interne systeemprompt voor versie 4.5 maakt voor het eerst op betekenisvolle wijze gebruik van XML-tags om instructies te structureren[[15]](https://www.prompthub.us/blog/everything-you-need-to-know-about-claude-4-5#:~:text=For%20all%20the%20noise%20Anthropic,them%20in%20a%20meaningful%20way) – een teken dat dergelijke opmaak als best practice wordt gezien bij geavanceerde AI’s. OpenAI’s eigen GPT-5 promptgids toont een vergelijkbare benadering: zij segmenteren instructies met aangepaste tags (bijv. <context\_gathering> en <persistence>) om de modelgedragingen fijn te regelen[[16]](https://cookbook.openai.com/examples/gpt-5/gpt-5_prompting_guide#:~:text=many%20ideas%3A)[[17]](https://cookbook.openai.com/examples/gpt-5/gpt-5_prompting_guide#:~:text=%3Cpersistence%3E%20,assumptions%2C%20as%20you%20can%20always). Dit onderstreept hoe belangrijk een duidelijke scheiding van context, doelen en criteria is voor effectieve prompt-engineering.

**Roltoewijzing en Tone-of-Voice:** Een krachtige techniek is om de AI een *rol* of *perspectief* te geven in je prompt. Door te beginnen met iets als *“Jij bent een senior software engineer gespecialiseerd in security testing…”* breng je het model in de juiste mindset om gericht advies of output te geven. Zo’n **rolprompting** kan de antwoorden gedetailleerder en deskundiger maken, omdat het model put uit kennis die past bij die rol[[18]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=%2A%20Chain,to%20not%20mix%20everything%20together). Bijvoorbeeld, testers lieten een model *“optreden als een ervaren security-tester”* om gerichte penetratietest-cases te krijgen[[18]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=%2A%20Chain,to%20not%20mix%20everything%20together). In Agile-teamcontext liet men een AI *“optreden als een Agile coach gespecialiseerd in waarde-prioritering”* om backlog items te ranken naar business value en risico’s[[19]](https://dzone.com/articles/ai-prompts-for-agile-scrum-developers#:~:text=%3E%20Prompt%3A%20,impact%2C%20risk%2C%20and%20delivery%20speed). Dit leverde gestructureerde aanbevelingen op die ontwikkelaars helpen om onderbouwde beslissingen te nemen. Roltoewijzing werkt goed in combinatie met een gewenste schrijfstijl: voor documentatie kun je bijvoorbeeld vragen *“Leg uit in eenvoudige termen, alsof je een collega onboardt”*. Markdown kan hier weer van pas komen door bijvoorbeeld een prompt te beginnen met een titel of rolindicatie als ## Rol: Senior DevOps Engineer gevolgd door ## Taak: ... enzovoort, of door codeblokken en citaatblokken te gebruiken om voorbeeldinput te scheiden.

**Voorbeelden en Few-shot Prompting:** Wanneer mogelijk, geef **voorbeeldinput en -output**. Het tonen van 1-2 voorbeelden (few-shot prompting) “traint” het model impliciet op het gewenste formaat en niveau van detail[[20]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=%2A%20Few,your%20field%20for%20your%20technical). In de softwarewereld kan dit betekenen dat je bijvoorbeeld eerst een correcte functie met unit test laat zien, en vervolgens vraagt om iets soortgelijks voor een nieuwe functie. Het aanhalen van voorbeelden blijkt zeer effectief om consistentie in de resultaten te krijgen[[20]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=%2A%20Few,your%20field%20for%20your%20technical). Zorg wel dat de voorbeelden relevant zijn voor de taak en het model niet afleiden met irrelevante details. Een variant hierop is **template prompting** – het gebruik van een vast sjabloon waarin je de variable delen invult. Bijvoorbeeld: *“Hieronder staat een goed bug-report voorbeeld. Volg exact dit format voor de nieuwe bug: [Beschrijving]…”* De AI zal geneigd zijn het format over te nemen. Let er bij voorbeelden op dat je duidelijke **scheidingstekens of delimiters** gebruikt (zoals ``` voor code of quotes voor teksten) zodat het model begrijpt wat voorbeeld is en wat opdracht[[21]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=a%2035%20years%20old%20security,to%20not%20mix%20everything%20together).

## Prompt Lengte: Kort vs. Lang?

Een prangende vraag in prompt engineering is of een **korte prompt** of een **uitgebreide prompt** betere resultaten geeft. Het antwoord blijkt genuanceerd en taakafhankelijk. Onderzoek naar large language models in specifieke domeinen laat zien dat rijkere context meestal de prestaties verbetert: *“Langere prompts verbeteren over het algemeen de modelprestatie, terwijl kortere prompts soms schadelijk kunnen zijn”*[[22]](https://arxiv.org/html/2502.14255v1#:~:text=findings%20indicate%20that%20longer%20prompts,for%20a%20deeper%20understanding%20of). In complexe taken – bijvoorbeeld code-analyse of financieel tekstbegrip – kan een langere prompt met meer achtergrondinformatie het model helpen om nauwkeuriger en relevanter te antwoorden[[22]](https://arxiv.org/html/2502.14255v1#:~:text=findings%20indicate%20that%20longer%20prompts,for%20a%20deeper%20understanding%20of). Denk aan het meegeven van relevante API-documentatie of functionele specificaties in je prompt; dit vergroot de kans dat de AI factueel correcte en bruikbare output genereert in lijn met jouw context.

Er is echter een keerzijde: **te lange of rommelige prompts kunnen de aandacht van het model verspreiden en de output juist verslechteren**. Praktijkervaringen tonen dat “meer context” na een bepaald punt omslaat in *ruis*. Een industry whitepaper waarschuwt dat extreem uitgebreide prompts het model kunnen afleiden: *“Uitgebreide prompts verdunnen de focus van het model, waardoor antwoorden vager en generieker worden dan bij bondige prompts”*[[23]](https://blog.promptlayer.com/disadvantage-of-long-prompt-for-llm/#:~:text=Reliability%20and%20Output%20Quality%20Degradation). Bovendien hebben LLMs een ingebouwd recency-bias: ze wegen recentere tokens zwaarder dan vroegere. Bij een zeer lange prompt bestaat het risico dat cruciale instructies aan het begin ondergesneeuwd raken of zelfs uit de context window vallen[[24]](https://blog.promptlayer.com/disadvantage-of-long-prompt-for-llm/#:~:text=Recency%20bias%20emerges%20as%20a,2%2C000%20tokens%2C%20wasting%20resources%20while). Zo kan een prompt van 10.000 tokens in de praktijk functioneren alsof alleen de laatste ~2.000 tokens volledig meegenomen worden[[25]](https://blog.promptlayer.com/disadvantage-of-long-prompt-for-llm/#:~:text=Recency%20bias%20emerges%20as%20a,resources%20while%20missing%20key%20context). Er is ook bewijs dat hallucinatierisico’s stijgen met promptlengte: in één experiment versloeg een goed gestructureerde prompt van 16k tokens (met gerichte retrieval van feiten) een monolithische prompt van 128k tokens qua accuraatheid en relevantie[[26]](https://blog.promptlayer.com/disadvantage-of-long-prompt-for-llm/#:~:text=missing%20key%20context). Met andere woorden, **kwaliteit van context weegt zwaarder dan kwantiteit** – gerichte, relevante details zijn beter dan een dump van alle denkbare informatie.

**Balans is dus essentieel.** Streef naar volledigheid, maar voorkom overbodige uitweidingen. Een praktische vuistregel is: *zo lang als nodig, maar zo beknopt als mogelijk*. Begin eventueel beknopt en vraag de AI om aan te geven of er extra informatie nodig is; je kunt dan iteratief extra context toevoegen. Moderne modellen als GPT-5 en Claude 4.5 hebben enorme contextwindows (honderdduizenden tokens), maar dat betekent niet dat elke prompt die grootte moet benaderen. Gebruik die ruimte verstandig voor complexe taken (bijv. meegeven van meerdere codebestanden voor een uitgebreide refactoring), maar houd de structuur en relevantie scherp. Onthoud ook dat langere prompts kosten- en tijdimpact hebben: meer tokens betekent hogere API-kosten en latency[[27]](https://blog.promptlayer.com/disadvantage-of-long-prompt-for-llm/#:~:text=The%20most%20immediate%20impact%20of,sensitive%20applications)[[28]](https://blog.promptlayer.com/disadvantage-of-long-prompt-for-llm/#:~:text=,04%20one). Soms is het efficiënter om een probleem op te delen in kleinere prompts/stappen dan één reuzenprompt te bouwen.

**Conclusie rond promptlengte:** verzorg een *rijke, maar gerichte* prompt. Geef voldoende informatie en context voor het probleem, maar respecteer de cognitieve beperkingen van het model. Waar mogelijk kun je ook **retrieval-augmented generation (RAG)** inzetten in plaats van een enorme statische prompt: laat de AI gericht bronnen ophalen op basis van een zoekvraag. Dit geeft je het beste van twee werelden – relevante content zonder dat je handmatig alles in de prompt hoeft te stoppen, wat bovendien hallucinaties reduceert (hierover meer in een volgende sectie).

## Geavanceerde Technieken en Moderne Modellen

**Chain-of-Thought Prompting:** Voor complexe problemen waarbij redenering in stappen nodig is (zoals debugging, algoritmische vraagstukken), is het effectief de AI te vragen haar denkproces stap-voor-stap uit te werken. Deze techniek, bekend als *Chain-of-Thought (CoT) prompting*, stuurt het model om eerst te *redeneren* en dan pas het eindantwoord te geven[[20]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=%2A%20Few,your%20field%20for%20your%20technical). Bijvoorbeeld: *“Denk stap voor stap na over dit probleem en geef dan je conclusie.”* Hiermee dwing je het model als het ware om logische tussenstappen te verwoorden, wat vaak tot correctere en beter beargumenteerde antwoorden leidt. Uit onderzoek is CoT een pionierstechniek gebleken om redeneerfouten te verminderen[[29]](https://arxiv.org/html/2502.14255v1#:~:text=Most%20existing%20work%20on%20prompt,Thoughts%20%28GoT%29%20prompting%C2%A0%5B43%5D%20leverages). Vele varianten zijn inmiddels ontwikkeld – van *self-consistency*, tot *tree-of-thoughts* en *graph-of-thoughts* – maar de kern blijft: je prompt ontlokt tussenstappen in plaats van direct een eindantwoord[[29]](https://arxiv.org/html/2502.14255v1#:~:text=Most%20existing%20work%20on%20prompt,Thoughts%20%28GoT%29%20prompting%C2%A0%5B43%5D%20leverages). In de praktijk van softwareontwikkeling kan CoT betekenen dat je de AI eerst vraagt om een plan of pseudo-code op te stellen, voordat deze de daadwerkelijke code schrijft. Dit maakt het makkelijker fouten of misinterpretaties te spotten en bevordert transparantie in de oplossing.

**Chain-of-Verification en “Step-Back”:** Nieuwere promptmethoden bouwen voort op CoT om hallucinaties verder tegen te gaan en nauwkeurigheid te vergroten. Een voorbeeld is *Chain-of-Verification (CoVe)*, waarbij het model na een eerste antwoord zelf controle-vragen formuleert en beantwoordt om het eigen antwoord te checken[[30]](https://www.prompthub.us/blog/three-prompt-engineering-methods-to-reduce-hallucinations#:~:text=Chain)[[31]](https://www.prompthub.us/blog/three-prompt-engineering-methods-to-reduce-hallucinations#:~:text=The%20researchers%20tested%20CoVe%20across,method%20and%20the%20experiments%20here). Pas daarna geeft het een definitief antwoord. Dit creëert als het ware een interne verificatielus. Onderzoekers rapporteerden dat CoVe in sommige gevallen prestaties met **tot 23%** verbeterde t.o.v. standaard prompting[[32]](https://www.prompthub.us/blog/three-prompt-engineering-methods-to-reduce-hallucinations#:~:text=%E2%80%8D)[[31]](https://www.prompthub.us/blog/three-prompt-engineering-methods-to-reduce-hallucinations#:~:text=The%20researchers%20tested%20CoVe%20across,method%20and%20the%20experiments%20here). Evenzo is *“Step-Back prompting”* een techniek waarbij de AI eerst op een hoger abstractieniveau over het probleem praat voordat het de specifieke vraag oplost[[33]](https://www.prompthub.us/blog/three-prompt-engineering-methods-to-reduce-hallucinations#:~:text=Chain%20of%20thought%20reasoning%20is,the%20end%20of%20your%20prompt)[[34]](https://www.prompthub.us/blog/three-prompt-engineering-methods-to-reduce-hallucinations#:~:text=%E2%80%8D). Dit zou je kunnen doen door eerst te vragen: *“Wat zijn de belangrijkste factoren die spelen in dit probleem?”* en daarna: *“Gebruik dat inzicht om de specifieke vraag op te lossen.”* Deze stap-vooraf aanpak geeft het model een moment om de context te herstructureren. In experimenten versloeg Step-Back prompting traditionele Chain-of-Thought met **tot 36%** betere nauwkeurigheid in sommige datasets[[34]](https://www.prompthub.us/blog/three-prompt-engineering-methods-to-reduce-hallucinations#:~:text=%E2%80%8D). Zulke cijfers suggereren dat een *reflectieve* promptaanpak de AI helpt om vollediger en correcter te antwoorden, vooral bij complexe vraagstukken.

**GPT-5 en Claude 4.5 – Nieuwe Generatie Modellen:** De nieuwste LLMs van 2025, zoals OpenAI’s GPT-5 en Anthropic’s Claude 4.5 (bijgenaamd “Sonnet”), brengen verbeteringen in contextbegrip, coderingsvaardigheden en alignement. Hoewel de precieze parameters bedrijfsgeheim zijn, weten we dat GPT-5 drastisch verbeterde *instruction-following* en *steerability* heeft[[35]](https://cookbook.openai.com/examples/gpt-5/gpt-5_prompting_guide#:~:text=GPT,coding%2C%20raw%20intelligence%2C%20and%20steerability). Dit betekent dat het model instructies doorgaans **nog betrouwbaarder volgt**, mits je de prompt correct opstelt. Veel van de **universele principes blijven echter geldig**: ook GPT-5 presteert het best met duidelijke, gestructureerde prompts en voldoende context. OpenAI’s eigen prompt-engineering gids voor GPT-5 benadrukt het belang van best practices en noemt dat het toepassen van deze technieken significante kwaliteitswinst oplevert[[36]](https://cookbook.openai.com/examples/gpt-5/gpt-5_prompting_guide#:~:text=We%E2%80%99ve%20seen%20significant%20gains%20from,remember%20that%20prompting%20is%20not). Men raadt zelfs aan een *prompt optimizer tool* in te zetten om de prompt iteratief te verbeteren[[36]](https://cookbook.openai.com/examples/gpt-5/gpt-5_prompting_guide#:~:text=We%E2%80%99ve%20seen%20significant%20gains%20from,remember%20that%20prompting%20is%20not). GPT-5 is verder ontworpen met **developers in mind**, inclusief betere ondersteuning voor *tool use* en langere gesprekken, wat betekent dat je in prompts kunt vertrouwen op nieuwere API-features als function calling en gereedschappen om taken op te lossen[[37]](https://cookbook.openai.com/examples/gpt-5/gpt-5_prompting_guide#:~:text=Agentic%20workflow%20predictability). Maar hoe krachtig het model ook is, een slecht geformuleerde prompt kan nog steeds tot ongewenste output leiden – garbage in, garbage out blijft van toepassing.

Claude 4.5 van Anthropic richt zich specifiek op coderen en langdurige *agentic* taken. Het wordt door Anthropic gepositioneerd als “de beste coding model beschikbaar” in 2025[[38]](https://www.prompthub.us/blog/everything-you-need-to-know-about-claude-4-5#:~:text=Coding). Voor prompt-engineering met Claude is één opvallende tip het gebruik van **XML-achtige structuur**. Anthropic heeft al langer gehint dat hun modellen goed overweg kunnen met gestructureerde prompts, en in Claude 4.5 is dat expliciet te zien: de system prompt bevat XML-tags om de instructies te organiseren[[15]](https://www.prompthub.us/blog/everything-you-need-to-know-about-claude-4-5#:~:text=For%20all%20the%20noise%20Anthropic,them%20in%20a%20meaningful%20way). Gebruikers merken dat Claude “dol is op XML” – bijvoorbeeld om input, constraints en gewenste outputdelen te labelen – waarschijnlijk omdat dit de AI helpt de onderdelen van de taak niet te verwarren[[39]](https://www.reddit.com/r/PromptEngineering/comments/1j4ia54/2_prompt_engineering_techniques_that_actually/#:~:text=2%20Prompt%20Engineering%20Techniques%20That,prefers%20XML%20formatting%20for%20input)[[15]](https://www.prompthub.us/blog/everything-you-need-to-know-about-claude-4-5#:~:text=For%20all%20the%20noise%20Anthropic,them%20in%20a%20meaningful%20way). Waar GPT-modellen vaak Markdown in antwoorden gebruiken, lijkt Claude een voorkeur te hebben om input zelf in een gelabelde vorm te krijgen[[40]](https://www.reddit.com/r/PromptEngineering/comments/1j4ia54/2_prompt_engineering_techniques_that_actually/#:~:text=1%20is%20called%20markdown%20formatting%2C,prefers%20XML%20formatting%20for%20input). Desalniettemin blijven de algemene principes hetzelfde: ook voor Claude 4.5 geldt dat context, duidelijkheid en stapsgewijze begeleiding cruciaal zijn voor een goed resultaat. Het model is minder vatbaar voor bepaalde valkuilen (zoals het makkelijker afwenden van prompt-injecties en sycophantic gedrag dan zijn voorgangers)[[41]](https://www.prompthub.us/blog/everything-you-need-to-know-about-claude-4-5#:~:text=Prompt%20injections), maar bij onduidelijke opdrachten kan ook Claude hallucineren of de plank misslaan. Daarom loont het om juist bij deze geavanceerde modellen *best practices* consequent toe te passen – zij zullen de nuance in goede prompts benutten om nog betere resultaten te leveren.

## Hallucinaties Voorkomen

**Hallucinaties** – het fenomeen dat de AI overtuigend klinkende onzin of onjuiste feiten verzint – zijn een bekend probleem. Prompt engineering speelt een sleutelrol in het minimaliseren van deze ongewenste verzinsels. Enkele strategieën die tot op heden (21 oktober 2025) als het meest effectief zijn gebleken, ondersteund door **feitelijk bewijs**, zijn:

* **Glasheldere Instructies en Context:** Veel hallucinaties ontstaan uit *ambiguïteit* of te weinig informatie[[42]](https://documentation.suse.com/suse-ai/1.0/html/AI-preventing-hallucinations/index.html#:~:text=The%20most%20common%20causes%20of,hallucinations%20are). Als de vraag vaag is, gaat het model gokken. Door specifieke, eenduidige prompts te schrijven verminder je de ruimte voor verzinsels[[43]](https://documentation.suse.com/suse-ai/1.0/html/AI-preventing-hallucinations/index.html#:~:text=2.1%20Set%20clear%20expectations%20). Bijvoorbeeld, liever *“Geef een samenvatting van NASA’s recente Mars-missies met feitelijke details uit officiële rapporten”* dan *“Vertel iets over de ruimte”*[[44]](https://documentation.suse.com/suse-ai/1.0/html/AI-preventing-hallucinations/index.html#:~:text=,%E2%80%9D). De SUSE-documentatie (2025) benadrukt dat een duidelijke prompt het model richting bekende feiten stuurt en zo hallucinaties tegengaat[[43]](https://documentation.suse.com/suse-ai/1.0/html/AI-preventing-hallucinations/index.html#:~:text=2.1%20Set%20clear%20expectations%20). Ook is het raadzaam om **bekende bronnen expliciet te noemen** als context. Een techniek genaamd *“According to… prompting”* vraagt het model te antwoorden gebaseerd op een bepaalde bron (“Volgens Wikipedia…, volgens een bepaald rapport…”). Dit werkt verrassend goed om antwoorden te grounded in feiten te houden – onderzoekers zagen hiermee tot **20% hogere accuraatheid** in antwoorden[[45]](https://www.prompthub.us/blog/three-prompt-engineering-methods-to-reduce-hallucinations#:~:text=Researchers%20found%20this%20method%20was,method%20and%20the%20experiments%20here).
* **Opsplitsen van Taken:** Breek complexe of brede vragen op in deelvragen of stapsgewijze prompts. Hiermee beperk je de scope waarover het model redeneert, wat het risico op hallucineren verlaagt[[46]](https://documentation.suse.com/suse-ai/1.0/html/AI-preventing-hallucinations/index.html#:~:text=2,). In plaats van één groot prompt voor een ingewikkeld probleem, kun je eerst een subvraag stellen, daarna doorvragen op details, etc. Deze *divide-and-conquer* aanpak dwingt de AI gefocust te blijven. Bijvoorbeeld, vraag eerst *“Wat zijn de laatste technologische ontwikkelingen in AI?”* en vervolgens *“Hoe worden die toegepast in de zorgsector?”* in plaats van direct *“Leg AI uit en hoe het de wereld verandert.”*[[46]](https://documentation.suse.com/suse-ai/1.0/html/AI-preventing-hallucinations/index.html#:~:text=2,). Het SUSE-gids benadrukt dat zo’n gefaseerde aanpak de kans verkleint dat het model buiten de lijntjes gaat kleuren[[46]](https://documentation.suse.com/suse-ai/1.0/html/AI-preventing-hallucinations/index.html#:~:text=2,).
* **Retrieval-Augmented Generation (RAG):** Voor feitelijke vragen is het ideaal als de AI uit een externe bron kan putten in plaats van uit z’n geheugen. Als je een kennisbank of documentatie tot je beschikking hebt, **verwerk dat in de prompt of via tools**. Bijvoorbeeld: *“Op basis van het volgende document [inhoud]… beantwoord X”*[[47]](https://documentation.suse.com/suse-ai/1.0/html/AI-preventing-hallucinations/index.html#:~:text=Techniques%20). Als je een API hebt die documenten kan ophalen (zoals een vector database), stuur dan eerst relevante context mee. Modellen die via RAG werken, hallucineren veel minder omdat ze concrete teksten hebben om op te leunen[[48]](https://documentation.suse.com/suse-ai/1.0/html/AI-preventing-hallucinations/index.html#:~:text=%2A%20No%20retrieval,need%20to%20generate%20specific%20information)[[49]](https://documentation.suse.com/suse-ai/1.0/html/AI-preventing-hallucinations/index.html#:~:text=2.3%20Use%20retrieval,). Een simpele prompt-hack is zinnetjes als *“Volgens de volgende bron…”* of *“Uit de officiële documentatie blijkt…”* te gebruiken, waardoor de AI geneigd is zijn antwoord feitelijk te onderbouwen[[50]](https://www.prompthub.us/blog/three-prompt-engineering-methods-to-reduce-hallucinations#:~:text=%E2%80%9CAccording%20to%E2%80%A6%E2%80%9D%20prompting). Tools of plugins kunnen dit automatiseren, maar ook in een statische prompt kun je stukken documentatie meesturen, mits binnen de contextlengte.
* **Beperk de Output en Vraag om Bronnen:** Hoe langer een AI’s antwoord door ratelt, hoe groter de kans dat er ergens een hallucinatie insluipt[[51]](https://documentation.suse.com/suse-ai/1.0/html/AI-preventing-hallucinations/index.html#:~:text=,chance%20that%20hallucinations%20can%20happen). Je kunt dit beperken door de **gevraagde output in omvang te begrenzen** of heel gericht te maken. Bijvoorbeeld: *“Geef mij in maximaal 3 zinnen de kern van…”* of *“Noem 5 feiten over… en niet meer dan dat.”*[[52]](https://documentation.suse.com/suse-ai/1.0/html/AI-preventing-hallucinations/index.html#:~:text=Limit%20the%20length%20or%20scope,topic%20or%20hallucinating%20extra%20details)[[53]](https://documentation.suse.com/suse-ai/1.0/html/AI-preventing-hallucinations/index.html#:~:text=,%E2%80%9D). Dit dwingt de AI om zich te concentreren op de belangrijkste punten. Daarnaast kun je de AI vragen haar bron of redenering te **verifiëren**: bijvoorbeeld *“Geef aan welke bron je voor elk feit gebruikt hebt”* of *“Controleer of elk bewering klopt met bekende feiten en meld als iets onzeker is.”*[[54]](https://documentation.suse.com/suse-ai/1.0/html/AI-preventing-hallucinations/index.html#:~:text=2.5%20Prompt%20for%20verification%20)[[55]](https://documentation.suse.com/suse-ai/1.0/html/AI-preventing-hallucinations/index.html#:~:text=Examples%20). Door deze extra instructie zal het model voorzichtiger zijn met uit de lucht gegrepen uitspraken en eerder geneigd zijn te zeggen “volgens [bron] is X” of zelfs “ik ben niet zeker”. Dit is in lijn met technieken als de eerder genoemde Chain-of-Verification, waarbij het model zijn eigen antwoorden kritisch naloopt. Hallucinaties worden zo gereduceerd doordat de AI actief bezig is met *fact-checken* binnen de prompt.
* **Gebruik van Redeneerketens (CoT):** Zoals genoemd, dwingt *Chain-of-Thought* een AI om intern logische stappen te doorlopen[[20]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=%2A%20Few,your%20field%20for%20your%20technical). Dit helpt niet alleen bij juiste oplossingen, maar ook om misverstanden te voorkomen. Een AI die stap voor stap redeneert (“Feit A, feit B, dus conclusie C”) zal minder gauw ineens feit D uit z’n duim zuigen omdat dat niet in de redeneerlijn past. Bovendien kun je de AI vragen om aan het eind te controleren of haar conclusie consistent is met de stappen. Hiermee leg je een vorm van zelfcontrole in de prompt in.
* **Geavanceerde Hallucinatie-reductie:** Naast bovenstaande, zijn er geavanceerde methodes in opkomst. We noemden *CoVe* en *Step-Back*, die expliciet bedoeld zijn om accuratesse te verhogen. Zulke methodes integreren eigenlijk meerdere prompts of stappen in één. In de praktijk zou je dit als eindgebruiker kunnen nadoen door multi-turn interacties: laat de AI eerst antwoorden, dan in een vervolgprompt dat antwoord analyseren of aanvullen, en dan samenvatten. Hoewel dit niet *één prompt* is, is het wel *prompt engineering* van de gehele sessie om hallucinaties uit te filteren. Ten slotte, als je programma’s bouwt rond LLMs, kun je zogenaamde **“kritische prompt loops”** implementeren: laat de AI na een antwoord expliciet reflecteren met een prompt als *“Klopt dit wel? Zijn er aannames die ik moet checken?”* en dan het verbeterde antwoord geven. Dit concept wordt ook onderzocht in academische kringen als onderdeel van *Reflexive Prompting* (2025) – waarbij het promptproces zelf terugkoppeling en correctie inbouwt.

## Prompt Engineering in de Software Development Cyclus

AI-prompting is niet alleen nuttig voor programmeurs die code genereren. Het strekt zich uit over de **hele software development cyclus** – van planning tot testen en onderhoud – en elke rol kan er voordeel uit halen met de juiste technieken:

### Voor Ontwikkelaars (Coding en Debugging)

Software engineers kunnen LLM’s inzetten als co-piloot bij het schrijven van code, refactoren en probleemoplossen. Effectieve prompts voor codetaken bevatten doorgaans:

* **Relevante Code Context:** Bijvoorbeeld, geef de functiedeclaratie of klasse waarop je wil voortbouwen, of beschrijf de module-architectuur in het prompt. *“Hier is de interface van klasse X... Voeg nu een methode toe die Y doet.”* Hoe meer de model weet van je codebase (hetzij via prompt of retrieval), hoe gerichter de output[[56]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=ChatGPT%20lacks%20direct%20access%20to,application%2C%20which%20creates%20several%20challenges)[[57]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=Solution%3A%20Provide%20relevant%20code%20snippets%2C,behaviour%20when%20crafting%20your%20prompts). Vergeet niet dat de AI je projectcontext niet *kent* tenzij je het vertelt – expliciet vermelden van taal, framework, en versie voorkomt misinterpretaties[[58]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=,to%20feed%20them%20into%20AI).
* **Duidelijke Taakomschrijving:** Zeg precies wat je wilt dat de code doet of wat voor oplossing je zoekt. Bijvoorbeeld: *“Schrijf een functie die een array sorteert met de QuickSort-methode in Python. Zorg voor docstrings en voer best-case en worst-case complexiteit aan in comments.”* Hier zie je dat we functionele eisen combineren met format-eisen (docstrings, commentaar). Dat leidt tot output die direct bruikbaar en informatief is.
* **Output-verwachting:** Als je wilt dat de AI code produceert, kun je dat expliciet sturen door bijvoorbeeld af te sluiten met *“Geef alleen de volledige functiedefinitie in Python, tussen triple backticks.”* Hierdoor zal de AI minder snel omliggende tekst of uitleg geven, maar direct de code uitspuwen in het juiste formaat. Markdown-codeblokken (```) zijn hierbij handig zodat de code netjes geformatteerd terugkomt.
* **Iteratief benaderen:** Voor complexe coding tasks kan het handig zijn eerst een plan of pseudocode te vragen (Chain-of-Thought toepassen). Prompt bijvoorbeeld: *“Beschrijf stap-voor-stap hoe je functionaliteit X zou implementeren.”* Na het plan kun je een nieuwe prompt geven: *“Schrijf nu de code uit voor stap 1 t/m 3.”* Dit bootst een beetje *rubber duck debugging* na, waar het model eerst de oplossing beredeneert en daarna uitvoert. Zo vang je fouten of verkeerde aannames vroeg.
* **Validatie in prompt:** Je kunt de AI ook inzetten om zijn eigen code te controleren. Vraag na het genereren: *“Kun je unit tests voor deze functie bedenken?”* of zelfs *“Controleer of de bovenstaande code edge cases zoals null-handling aankan en pas aan indien niet.”* Dit moedigt het model aan om verder te denken dan de eerste oplossing en verhoogt de kwaliteit.

GPT-5 en vergelijkbare modellen zijn zeer sterk in code; ze kunnen hele modules schrijven, maar de beste resultaten komen als je ze **geleid met goede prompts**. Interessant is dat GPT-5’s API de mogelijkheid biedt om *tools* te gebruiken – in een coding context zou dat kunnen betekenen dat de AI direct compileerfouten krijgt of documentatie ophaalt. In een prompt kun je daar gebruik van maken door het model te laten weten dat het een *REPL* of *linter* ter beschikking heeft, als dat in jouw toepassing zo is. Kortom: zie een LLM als een junior developer die je aanstuurt – hoe beter je de opdracht en context formuleert, hoe meer het resultaat zal lijken op het werk van een senior.

### Voor Testers en QA

Quality Assurance en test-ingenieurs ontdekken dat **prompt engineering een versneller kan zijn** in hun werk[[59]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=Prompt%20engineering%20is%20already%20an,bugs%20that%20actually%20impact%20users). Enkele toepassingen:

* **Testcase Generatie:** Zoals te zien in de *aqua cloud* voorbeelden, kun je prompts schrijven die op basis van een feature-beschrijving tientallen testgevallen voorstellen[[60]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=Let%E2%80%99s%20start%20with%20the%20core,efforts%2C%20test%20case%20generation%20prompts)[[61]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=1.%20Requirement,%E2%80%9D). Een goed prompt voor testcases geeft de functionaliteit en specificeert dat zowel *happy flow* als *edge cases* gewenst zijn, inclusief velden als precondities, stappen en verwachte resultaten. Bijvoorbeeld: *“Genereer testcases voor de inlogfunctionaliteit. Neem zowel positieve scenario’s (correct wachtwoord) als negatieve (verkeerd wachtwoord, leeg veld, brute force) mee. Formatteer ieder testgeval met: ID, titel, preconditie, stappen, verwacht resultaat.”* Zo’n prompt levert een hele suite van testgevallen in een paar seconden, waar een mens uren over zou doen – en uit anekdotisch bewijs blijkt dat de dekking hierdoor aanzienlijk verhoogd kan worden[[62]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=,include%20relevant%20constraints%20or%20requirements)[[61]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=1.%20Requirement,%E2%80%9D).
* **Bug Reports & Analyses:** Testers kunnen AI gebruiken om een beknopte maar volledige bugmelding op te stellen. Een prompt kan bijvoorbeeld een foutbeschrijving en stack trace bevatten en vragen: *“Genereer een bugrapport inclusief samenvatting, stappen om te reproduceren, verwacht vs. actueel gedrag, omgeving, en mogelijke oorzaak.”* Het model zal gestructureerd zo’n rapport uitschrijven[[63]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=Bug%20Reporting%20Prompts). Evenzo kun je een gegenereerde foutmelding laten analyseren: *“Analyseer deze error en suggesteer mogelijke oorzaken en oplossingen.”* De AI fungeert dan als een second-opinion of *rubber duck*, wat junior testers enorm kan helpen om sneller tot de kern van een probleem te komen[[64]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=severity%20assessment%2C%20and%20potential%20impact).
* **Test Data en Edge Cases:** Het verzinnen van testdata (bijv. JSON records, namen, nummers) is typisch saai werk. Met prompt engineering kun je dit automatiseren: *“Genereer 10 voorbeeldrecords in JSON voor gebruikers, met velden X, Y, Z (verschillende leeftijden, datums in de laatste 3 jaar, etc.).”*[[65]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=Test%20Data%20Generation) De AI produceert direct een divers datasetje. Je kunt zelfs vragen om extreme gevallen: *“Maak testtransacties op de grens van het maximumlimiet, met vreemde valuta en ongewone karakters in de beschrijving.”*[[66]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=%28free%2Fpremium%2Fenterprise%29%2C%20and%20usage%20statistics). Dit helpt om je systeem te testen op robuustheid.
* **Risicoanalyse en Prioritering:** QA’s in een agile team kunnen prompts gebruiken om snel risico’s in kaart te brengen. Bijvoorbeeld: *“Analyseer deze nieuwe featurebeschrijving op mogelijke technische en UX-risico’s. Geef een lijst van risico’s met impact en kans, en stel mitigaties voor.”*[[67]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=Risk%20Assessment%20Prompts). De AI somt dan potentiele pijnpunten op (misschien dingen waar je zelf nog niet aan dacht). Ook kan het helpen bij testplanning: *“Je hebt 3 dagen testtijd en volgende user stories: [X]. Welke tests zou je prioriteit geven en waarom?”*[[68]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=impact%20and%20probability%2C%20and%20suggest,%E2%80%9D). Dit dwingt de AI een prioritering te maken – handig als sparring om je eigen planning aan te scherpen.

**Prompt Tips specifiek voor QA:** Wees hier extra zorgvuldig met details. Test-prompts werken het best als ze super specifiek zijn over format (bijv. “geef antwoord in tabelvorm” voor testcases, of “gebruik koppeltekens voor elke stap” in een bugrepro). De QA-ervaring leert dat *“plain text resultaten lage kwaliteit hebben; vraag om tabellen, genummerde lijsten of JSON waar passend”*[[69]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=and,If%20you%20don%E2%80%99t%20have%20the)[[70]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=,to%20you%3F%20Don%E2%80%99t%20be%20lazy). Daarnaast kun je rolprompting inzetten (“Handel als een QA-manager van 10 jaar ervaring…”) om het model een kritischer, kwaliteitsgericht perspectief te geven[[18]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=%2A%20Chain,to%20not%20mix%20everything%20together). Combineer dat met chain-of-thought: *“Denk stap voor stap na of elk scenario volledig is.”* – zodat je geen gaten mist. En, net als bij developers, geldt: **verifieer de output zelf**. AI kan verouderde terminologie of syntaxis gebruiken (bv. een oudere Selenium API noemen)[[71]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=Technical%20Accuracy%20Concerns). Daarom moet een mens altijd reviewen en finetunen. Zie de AI als assistent die het zware tilwerk doet, maar jij houdt de controle over de uiteindelijke kwaliteit[[72]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=Solution%3A%20Always%20review%20and%20verify,refine%20rather%20than%20final%20products)[[73]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=Solution%3A%20Use%20AI%20as%20a,assistance%20and%20manual%20testing%20efforts).

### Voor Scrum Masters en Agile Teams

Zelfs in het domein van planning, scrum en samenwerking kunnen LLM’s goed van pas komen:

* **User Stories & Backlog Grooming:** Een Scrum Master of Product Owner kan AI vragen helpen bij het schrijven of verbeteren van user stories. Prompt bijvoorbeeld: *“Verfijn de volgende user story volgens INVEST-criteria: 'Als gebruiker wil ik X zodat Y'. Voeg acceptatiecriteria toe.”* De AI zal een nettere user story formuleren met duidelijke criteria. Ook prioritering kan, zoals eerder genoemd, met prompts gebeuren: een AI in de rol van *Agile coach* kan backlog items ranken op waarde en risico[[19]](https://dzone.com/articles/ai-prompts-for-agile-scrum-developers#:~:text=%3E%20Prompt%3A%20,impact%2C%20risk%2C%20and%20delivery%20speed), wat als input kan dienen voor discussie in planning.
* **Sprint Planning Assistance:** Ontwikkelaars hebben bijvoorbeeld prompts gemaakt om vage stories in subtaken te breken. Een voorbeeld uit de praktijk: *“Als een expert in Agile backlog refinement, help me om deze story op te breken: '[story beschrijving]'. Geef een lijst van subtaken met realistische inschattingen in uren. Wijs op ontbrekende requirements.”*[[74]](https://dzone.com/articles/ai-prompts-for-agile-scrum-developers#:~:text=%3E%20Prompt%3A%20,Flag%20any%20missing%20requirements). Dit leverde een concreet subtaken-overzicht op, inclusief eventuele onduidelijkheden die nog opgehelderd moeten worden – iets dat direct bruikbaar is in een planningsmeeting. Dergelijke prompts, ondersteund door onderzoek naar AI in agile (Alamu et al., 2024; Verma et al., 2025), laten zien dat *gestructureerde prompt templates agile rituelen efficiënter kunnen maken*[[75]](https://dzone.com/articles/ai-prompts-for-agile-scrum-developers#:~:text=In%20this%20guide%2C%20we%20share,just%20for%20your%20Scrum%20Master).
* **Stand-ups en Status Updates:** Voor remote teams of gewoon om iedereen scherp te houden kan een LLM helpen om stand-up updates samen te vatten of te verbeteren. Een mogelijke toepassing is dat elke developer hun ruwe notities invoert, en de AI vraagt: *“Combineer deze stand-up updates tot een bondig verslag, highlight eventuele blockers en voorgestelde oplossingen.”* Zo krijg je binnen seconden een net geformuleerd verslag van wat er in het team gaande is, wat je weer kunt delen.
* **Retrospectives en Reviews:** Je zou zelfs een AI kunnen vragen om sentiment uit retro-feedback te distilleren: *“Gegeven de volgende retrospectieve opmerkingen van het team, som de top 3 verbeterpunten en top 3 dingen die goed gingen op.”* Dit versnelt de analyse van wat anders een tijdrovende klussen kan zijn, en het model kan ook suggesties doen voor actiepunten (“Misschien kunnen we volgende sprint…”) die de SM kan overwegen.

Bij al deze Agile-toepassingen geldt: de AI **vervangt de mens niet**, maar ondersteunt. De scrum master behoudt leiding, maar met prompt engineering kun je routinewerk (zoals notulen maken, eerste drafts van documenten schrijven, info ordenen) offloaden naar de AI. Dit laat meer tijd over voor het menselijke werk: beslissingen nemen, teamdynamiek managen, enz.

**Let op:** communicatie en nuance zijn cruciaal in deze context. Als je AI inzet om teksten te genereren die voor een team bedoeld zijn, zorg dat je de output naloopt op tone-of-voice en juistheid. Prompt eventueel in de stijl van je teamcultuur (formeel vs informeel). En vermijd dat vertrouwelijke of gevoelige teaminformatie ongefilterd in prompts belandt – *sanitize* die indien nodig (denk aan geen persoongsgegevens of bedrijfsgeheimen direct voeren, of gebruik een on-premise LLM oplossing als dat een zorg is)[[76]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=Privacy%20and%20Security%20Considerations)[[77]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=Solution%3A%20Sanitise%20sensitive%20information%20before,descriptions%20when%20discussing%20proprietary%20systems).

## Conclusie

Prompt engineering is anno oktober 2025 uitgegroeid tot een onmisbare vaardigheid voor iedereen in softwareontwikkeling – van de senior developer tot de tester en zelfs de scrum master[[59]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=Prompt%20engineering%20is%20already%20an,bugs%20that%20actually%20impact%20users). We hebben inmiddels een goed beeld van wat een prompt effectief maakt: **wees specifiek, geef context, structureer je vraag, en leid de AI door complexe taken met voorbeelden of tussenstappen**. Gebruikmakend van deze principes, en ondersteund door feitelijk bewijs uit onderzoek en praktijk, kun je AI-hulpmiddelen zoals GPT-5 en Claude 4.5 maximaal tot hun recht laten komen. Moderne modellen zijn krachtiger en “slimmer” dan ooit, maar ze presteren het best als wij **de juiste aanwijzingen** geven.

Cruciaal is ook het besef dat prompt engineering een **iteratief proces** is – er is zelden in één keer een perfecte prompt. Experts raden aan om te experimenteren, kleine aanpassingen te testen, en successen te documenteren (bijvoorbeeld een library van goed werkende prompts binnen je team opbouwen)[[78]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=helps%20you%20get%20real%20value,bugs%20that%20actually%20impact%20users)[[79]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=documentation%2C%20and%20bug%20analysis,you%E2%80%99ll%20shift%20your%20focus%20from). Deel die kennis, leer van elkaar, en blijf verbeteren. Door AI niet als zwarte doos maar als samenwerkingspartner te benaderen – eentje die je instructies en context nodig heeft – verschuift jouw focus van repetitief werk naar creatief en kritisch werk: *het oplossen van de échte problemen*. Met de strategieën en technieken die we tot nu toe geleerd hebben, gewapend met data en voorbeelden, kun je prompt engineering inzetten om sneller te ontwikkelen, beter te testen en effectiever samen te werken. En misschien wel het mooiste: je houdt het model eerlijk en nuttig, zonder hallucinaties en met resultaten waar je op kunt bouwen.

**Bronnen:** De inzichten in dit verslag zijn gebaseerd op een reeks recente richtlijnen, onderzoeken en praktijkcases, waaronder de SUSE AI-prompt handleiding[[1]](https://documentation.suse.com/suse-ai/1.0/html/AI-preventing-hallucinations/index.html#:~:text=The%20most%20common%20causes%20of,hallucinations%20are)[[43]](https://documentation.suse.com/suse-ai/1.0/html/AI-preventing-hallucinations/index.html#:~:text=2.1%20Set%20clear%20expectations%20), PromptHub experimenten[[32][34]](https://www.prompthub.us/blog/three-prompt-engineering-methods-to-reduce-hallucinations#:~:text=%E2%80%8D), OpenAI’s en Anthropic’s modeldocumentatie[[15]](https://www.prompthub.us/blog/everything-you-need-to-know-about-claude-4-5#:~:text=For%20all%20the%20noise%20Anthropic,them%20in%20a%20meaningful%20way)[[16]](https://cookbook.openai.com/examples/gpt-5/gpt-5_prompting_guide#:~:text=many%20ideas%3A), en gespecialiseerde blogartikelen over prompt technieken voor testers[[6]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=Be%20Specific%20and%20Detailed)[[20]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=%2A%20Few,your%20field%20for%20your%20technical) en agile teams[[74]](https://dzone.com/articles/ai-prompts-for-agile-scrum-developers#:~:text=%3E%20Prompt%3A%20,Flag%20any%20missing%20requirements). Deze referenties leveren het feitelijke fundament voor de genoemde best practices en resultaten.

[[1]](https://documentation.suse.com/suse-ai/1.0/html/AI-preventing-hallucinations/index.html#:~:text=The%20most%20common%20causes%20of,hallucinations%20are) [[42]](https://documentation.suse.com/suse-ai/1.0/html/AI-preventing-hallucinations/index.html#:~:text=The%20most%20common%20causes%20of,hallucinations%20are) [[43]](https://documentation.suse.com/suse-ai/1.0/html/AI-preventing-hallucinations/index.html#:~:text=2.1%20Set%20clear%20expectations%20) [[44]](https://documentation.suse.com/suse-ai/1.0/html/AI-preventing-hallucinations/index.html#:~:text=,%E2%80%9D) [[46]](https://documentation.suse.com/suse-ai/1.0/html/AI-preventing-hallucinations/index.html#:~:text=2,) [[47]](https://documentation.suse.com/suse-ai/1.0/html/AI-preventing-hallucinations/index.html#:~:text=Techniques%20) [[48]](https://documentation.suse.com/suse-ai/1.0/html/AI-preventing-hallucinations/index.html#:~:text=%2A%20No%20retrieval,need%20to%20generate%20specific%20information) [[49]](https://documentation.suse.com/suse-ai/1.0/html/AI-preventing-hallucinations/index.html#:~:text=2.3%20Use%20retrieval,) [[51]](https://documentation.suse.com/suse-ai/1.0/html/AI-preventing-hallucinations/index.html#:~:text=,chance%20that%20hallucinations%20can%20happen) [[52]](https://documentation.suse.com/suse-ai/1.0/html/AI-preventing-hallucinations/index.html#:~:text=Limit%20the%20length%20or%20scope,topic%20or%20hallucinating%20extra%20details) [[53]](https://documentation.suse.com/suse-ai/1.0/html/AI-preventing-hallucinations/index.html#:~:text=,%E2%80%9D) [[54]](https://documentation.suse.com/suse-ai/1.0/html/AI-preventing-hallucinations/index.html#:~:text=2.5%20Prompt%20for%20verification%20) [[55]](https://documentation.suse.com/suse-ai/1.0/html/AI-preventing-hallucinations/index.html#:~:text=Examples%20) Preventing AI Hallucinations with Effective User Prompts

<https://documentation.suse.com/suse-ai/1.0/html/AI-preventing-hallucinations/index.html>

[[2]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=,different%20from%20what%20you%20desire) [[3]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=Before%20diving%20into%20specific%20prompts%2C,effective%20prompts%20share%20certain%20characteristics) [[4]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=When%20a%20prompt%20tester%20experiments,useful%20and%20detailed%20test%20cases) [[5]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=massively,useful%20and%20detailed%20test%20cases) [[6]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=Be%20Specific%20and%20Detailed) [[7]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=,to%20you%3F%20Don%E2%80%99t%20be%20lazy) [[18]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=%2A%20Chain,to%20not%20mix%20everything%20together) [[20]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=%2A%20Few,your%20field%20for%20your%20technical) [[21]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=a%2035%20years%20old%20security,to%20not%20mix%20everything%20together) [[56]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=ChatGPT%20lacks%20direct%20access%20to,application%2C%20which%20creates%20several%20challenges) [[57]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=Solution%3A%20Provide%20relevant%20code%20snippets%2C,behaviour%20when%20crafting%20your%20prompts) [[58]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=,to%20feed%20them%20into%20AI) [[59]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=Prompt%20engineering%20is%20already%20an,bugs%20that%20actually%20impact%20users) [[60]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=Let%E2%80%99s%20start%20with%20the%20core,efforts%2C%20test%20case%20generation%20prompts) [[61]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=1.%20Requirement,%E2%80%9D) [[62]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=,include%20relevant%20constraints%20or%20requirements) [[63]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=Bug%20Reporting%20Prompts) [[64]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=severity%20assessment%2C%20and%20potential%20impact) [[65]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=Test%20Data%20Generation) [[66]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=%28free%2Fpremium%2Fenterprise%29%2C%20and%20usage%20statistics) [[67]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=Risk%20Assessment%20Prompts) [[68]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=impact%20and%20probability%2C%20and%20suggest,%E2%80%9D) [[69]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=and,If%20you%20don%E2%80%99t%20have%20the) [[70]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=,to%20you%3F%20Don%E2%80%99t%20be%20lazy) [[71]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=Technical%20Accuracy%20Concerns) [[72]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=Solution%3A%20Always%20review%20and%20verify,refine%20rather%20than%20final%20products) [[73]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=Solution%3A%20Use%20AI%20as%20a,assistance%20and%20manual%20testing%20efforts) [[76]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=Privacy%20and%20Security%20Considerations) [[77]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=Solution%3A%20Sanitise%20sensitive%20information%20before,descriptions%20when%20discussing%20proprietary%20systems) [[78]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=helps%20you%20get%20real%20value,bugs%20that%20actually%20impact%20users) [[79]](https://aqua-cloud.io/prompt-engineering-for-testers/#:~:text=documentation%2C%20and%20bug%20analysis,you%E2%80%99ll%20shift%20your%20focus%20from) Prompt Engineering for Software Testers: Best Practices for 2025

<https://aqua-cloud.io/prompt-engineering-for-testers/>

[[8]](https://www.ssw.com.au/rules/ai-prompt-xml/#:~:text=understand) [[9]](https://www.ssw.com.au/rules/ai-prompt-xml/#:~:text=Why%20use%20Markdown%20or%20XML,tags) [[10]](https://www.ssw.com.au/rules/ai-prompt-xml/#:~:text=struggle%20too) [[11]](https://www.ssw.com.au/rules/ai-prompt-xml/#:~:text=%2A%20Clarity%20,prompt%20by%20having%20everything%20atomic) [[14]](https://www.ssw.com.au/rules/ai-prompt-xml/#:~:text=Using%20Markdown%20or%20XML%20tags,quality%20outputs) Do you use Markdown or XML tags to structure your AI prompts? | SSW.Rules

<https://www.ssw.com.au/rules/ai-prompt-xml/>

[[12]](https://medium.com/@isaiahdupree33/optimal-prompt-formats-for-llms-xml-vs-markdown-performance-insights-cef650b856db#:~:text=,negative%20score%20across%20the%20board) [[13]](https://medium.com/@isaiahdupree33/optimal-prompt-formats-for-llms-xml-vs-markdown-performance-insights-cef650b856db#:~:text=,negative%20score%20across%20the%20board) Optimal Prompt Formats for LLMs: XML vs Markdown Performance Insights | by Isaiah Dupree | Medium

<https://medium.com/@isaiahdupree33/optimal-prompt-formats-for-llms-xml-vs-markdown-performance-insights-cef650b856db>

[[15]](https://www.prompthub.us/blog/everything-you-need-to-know-about-claude-4-5#:~:text=For%20all%20the%20noise%20Anthropic,them%20in%20a%20meaningful%20way) [[38]](https://www.prompthub.us/blog/everything-you-need-to-know-about-claude-4-5#:~:text=Coding) [[41]](https://www.prompthub.us/blog/everything-you-need-to-know-about-claude-4-5#:~:text=Prompt%20injections) Everything You Need to Know about Claude 4.5

<https://www.prompthub.us/blog/everything-you-need-to-know-about-claude-4-5>

[[16]](https://cookbook.openai.com/examples/gpt-5/gpt-5_prompting_guide#:~:text=many%20ideas%3A) [[17]](https://cookbook.openai.com/examples/gpt-5/gpt-5_prompting_guide#:~:text=%3Cpersistence%3E%20,assumptions%2C%20as%20you%20can%20always) [[35]](https://cookbook.openai.com/examples/gpt-5/gpt-5_prompting_guide#:~:text=GPT,coding%2C%20raw%20intelligence%2C%20and%20steerability) [[36]](https://cookbook.openai.com/examples/gpt-5/gpt-5_prompting_guide#:~:text=We%E2%80%99ve%20seen%20significant%20gains%20from,remember%20that%20prompting%20is%20not) [[37]](https://cookbook.openai.com/examples/gpt-5/gpt-5_prompting_guide#:~:text=Agentic%20workflow%20predictability) GPT-5 prompting guide

<https://cookbook.openai.com/examples/gpt-5/gpt-5_prompting_guide>

[[19]](https://dzone.com/articles/ai-prompts-for-agile-scrum-developers#:~:text=%3E%20Prompt%3A%20,impact%2C%20risk%2C%20and%20delivery%20speed) [[74]](https://dzone.com/articles/ai-prompts-for-agile-scrum-developers#:~:text=%3E%20Prompt%3A%20,Flag%20any%20missing%20requirements) [[75]](https://dzone.com/articles/ai-prompts-for-agile-scrum-developers#:~:text=In%20this%20guide%2C%20we%20share,just%20for%20your%20Scrum%20Master) AI Prompts Every Developer Should Steal

<https://dzone.com/articles/ai-prompts-for-agile-scrum-developers>

[[22]](https://arxiv.org/html/2502.14255v1#:~:text=findings%20indicate%20that%20longer%20prompts,for%20a%20deeper%20understanding%20of) [[29]](https://arxiv.org/html/2502.14255v1#:~:text=Most%20existing%20work%20on%20prompt,Thoughts%20%28GoT%29%20prompting%C2%A0%5B43%5D%20leverages) Effects of Prompt Length on Domain-specific Tasks for Large Language Models

<https://arxiv.org/html/2502.14255v1>

[[23]](https://blog.promptlayer.com/disadvantage-of-long-prompt-for-llm/#:~:text=Reliability%20and%20Output%20Quality%20Degradation) [[24]](https://blog.promptlayer.com/disadvantage-of-long-prompt-for-llm/#:~:text=Recency%20bias%20emerges%20as%20a,2%2C000%20tokens%2C%20wasting%20resources%20while) [[25]](https://blog.promptlayer.com/disadvantage-of-long-prompt-for-llm/#:~:text=Recency%20bias%20emerges%20as%20a,resources%20while%20missing%20key%20context) [[26]](https://blog.promptlayer.com/disadvantage-of-long-prompt-for-llm/#:~:text=missing%20key%20context) [[27]](https://blog.promptlayer.com/disadvantage-of-long-prompt-for-llm/#:~:text=The%20most%20immediate%20impact%20of,sensitive%20applications) [[28]](https://blog.promptlayer.com/disadvantage-of-long-prompt-for-llm/#:~:text=,04%20one) Disadvantage of Long Prompt for LLM

<https://blog.promptlayer.com/disadvantage-of-long-prompt-for-llm/>

[[30]](https://www.prompthub.us/blog/three-prompt-engineering-methods-to-reduce-hallucinations#:~:text=Chain) [[31]](https://www.prompthub.us/blog/three-prompt-engineering-methods-to-reduce-hallucinations#:~:text=The%20researchers%20tested%20CoVe%20across,method%20and%20the%20experiments%20here) [[32]](https://www.prompthub.us/blog/three-prompt-engineering-methods-to-reduce-hallucinations#:~:text=%E2%80%8D) [[33]](https://www.prompthub.us/blog/three-prompt-engineering-methods-to-reduce-hallucinations#:~:text=Chain%20of%20thought%20reasoning%20is,the%20end%20of%20your%20prompt) [[34]](https://www.prompthub.us/blog/three-prompt-engineering-methods-to-reduce-hallucinations#:~:text=%E2%80%8D) [[45]](https://www.prompthub.us/blog/three-prompt-engineering-methods-to-reduce-hallucinations#:~:text=Researchers%20found%20this%20method%20was,method%20and%20the%20experiments%20here) [[50]](https://www.prompthub.us/blog/three-prompt-engineering-methods-to-reduce-hallucinations#:~:text=%E2%80%9CAccording%20to%E2%80%A6%E2%80%9D%20prompting) Three Prompt Engineering Methods to Reduce Hallucinations

<https://www.prompthub.us/blog/three-prompt-engineering-methods-to-reduce-hallucinations>

[[39]](https://www.reddit.com/r/PromptEngineering/comments/1j4ia54/2_prompt_engineering_techniques_that_actually/#:~:text=2%20Prompt%20Engineering%20Techniques%20That,prefers%20XML%20formatting%20for%20input) [[40]](https://www.reddit.com/r/PromptEngineering/comments/1j4ia54/2_prompt_engineering_techniques_that_actually/#:~:text=1%20is%20called%20markdown%20formatting%2C,prefers%20XML%20formatting%20for%20input) 2 Prompt Engineering Techniques That Actually Work (With Data)

<https://www.reddit.com/r/PromptEngineering/comments/1j4ia54/2_prompt_engineering_techniques_that_actually/>