Whitepaper

Vibecoding & Software Engineering.

Van snelle applicatie
naar betrouwbaar systeem.

SOLUTION STUDIO AI

POWERED BY AVISI

Management summary.

Vibecoding, het genereren van software met natuurlijke taal, stelt organisaties in staat om ideeën vrijwel direct tastbaar te maken in plaats van pas na een lang ontwikkeltraject. Die snelheid is waardevol: aannames kunnen al vroeg in het traject worden getoetst, waardoor het risico op het ontwikkelen van de verkeerde scope wordt verkleind.

Diezelfde snelheid brengt een risico met zich mee. Onderzoek laat zien dat 45 procent van AI gegenereerde code securitytests niet doorstaat, en een groeiend deel van software engineers wantrouwt de nauwkeurigheid van AI output. Een prototype dat overtuigt tijdens een demonstratie is nog geen software die jarenlang veilig, schaalbaar en onderhoudbaar functioneert. Zonder heldere kaders kan een experiment bovendien ongemerkt uitgroeien tot een productiesysteem zonder eigenaar, beheer of zicht op kosten.

De oplossing is niet om vibecoding te vermijden, maar om het te combineren met de juiste menselijke expertise: domeinkennis en requirements engineering vooraf, context engineering en technische guardrails tijdens het bouwen, en menselijke controle over kwaliteit en risico's. Avisi past deze aanpak toe in de Solution Studio, waarin vibecoding gericht wordt ingezet om in korte tijd een prototype én een onderbouwd requirementsdocument op te leveren, als basis voor veilige en schaalbare doorontwikkeling.

Dit whitepaper beschrijft wat vibecoding bestuurders oplevert, waar de risico's zitten en hoe Avisi organisaties helpt om de voordelen te benutten zonder grip op kwaliteit, veiligheid en verantwoordelijkheid te verliezen.

Inhoud.

> **“Van een idee op een scherm naar een systeem waar een organisatie op kan bouwen.”**

Inleiding.

Waar teams eerder al met design sprints binnen een week een eerste prototype konden tonen, kan een bestuurder, ondernemer of professional dankzij vibecoding tegenwoordig zelf, zonder tussenkomst van een ontwikkelteam, in zeer korte tijd een werkende digitale oplossing in handen hebben. Beschrijf in gewone taal wat een applicatie moet doen, voeg enkele documenten of een dataset toe, en generatieve AI levert binnen korte tijd een eerste prototype op.

Deze manier van werken staat tegenwoordig algemeen bekend als vibecoding. Andrej Karpathy introduceerde het begrip in februari 2025 voor een manier van programmeren waarbij iemand vooral beschrijft wat er moet gebeuren, de gegenereerde software uitprobeert en de AI op basis van het resultaat nieuwe opdrachten geeft. De gebruiker hoeft de onderliggende code niet noodzakelijk te lezen of volledig te begrijpen.

De aantrekkingskracht is vanzelfsprekend. Een idee dat voorheen alleen bestond uit gesprekken, moodboards of slidedecks, kan nu in korte tijd worden ervaren: gebruikers reageren op iets wat daadwerkelijk voor hen staat, en bestuurders zien direct wat een digitale oplossing kan betekenen. Aannames worden zichtbaar voordat er grote investeringen zijn gedaan.

Daarmee biedt vibecoding precies wat veel innovatietrajecten missen: snelheid, concreetheid en momentum.

Maar die snelheid kan ook tot een misverstand leiden. Dat software snel kan worden gegenereerd, betekent nog niet dat er een betrouwbaar softwaresysteem is ontstaan.

Een prototype dat tijdens een demonstratie overtuigt, is iets wezenlijk anders dan software die jarenlang moet standhouden: veilig functioneren, integreren met bestaande systemen, persoonsgegevens verwerken, piekbelasting aankunnen, en aanpasbaar blijven voor verschillende teams.

De relevante vraag voor bestuurders is daarom niet óf vibecoding waarde heeft; dat staat buiten kijf. De vraag is waarvoor organisaties het gebruiken, waar in de levenscyclus van software die waarde het grootst is en vanaf welk moment professionele softwareontwikkeling noodzakelijk wordt.

Eerst de begrippen.

Vibecoding

Vibecoding is het met natuurlijke taal laten genereren en aanpassen van software door generatieve AI. De term werd in februari 2025 geïntroduceerd door AI-onderzoeker Andrej Karpathy. Hij beschreef een manier van programmeren waarbij iemand zich grotendeels door het resultaat laat leiden: iets proberen, bekijken wat er gebeurt en de AI vervolgens vertellen wat anders moet.

AI-assisted engineering

AI-assisted engineering is het professioneel ontwikkelen en beheren van software waarbij ervaren engineers AI doelgericht inzetten binnen vastgestelde architectuur-, security- en kwaliteitskaders. AI kan daarbij helpen met onder meer analyse, codegeneratie, testen, documentatie en foutopsporing. Mensen en de organisatie nemen daarbij bewust en actief de verantwoordelijkheid voor de werking, veiligheid en onderhoudbaarheid van de software.

Solution Studio

De Solution Studio is de werkwijze waarin Avisi's requirements engineers, software-engineers en domeinexperts van de klant in korte tijd een prototype ontwikkelen. Dat gaat verder dan in natuurlijke taal instrueren en bijsturen op het zichtbare resultaat; de aanpak die doorgaans onder vibecoding wordt verstaan. Onder begeleiding van requirements engineers worden requirements vastgelegd volgens een vaste methodiek; deze gevalideerde requirements vormen mede de input voor de AI om code te genereren. Daarbij wordt bewust gewerkt met dezelfde taal, techniek en frameworks die ook in productiesoftware worden gebruikt, zodat het resultaat niet alleen conceptueel maar ook technisch aansluit op een eventueel vervolgtraject.

Het resultaat is dan ook niet alleen een klikbaar prototype, maar een technisch onderbouwde basis: een requirementsdocument waarmee het vraagstuk is doorgrond en aannames zijn getoetst, in code die al is voorbereid op veilige en schaalbare doorontwikkeling.

Vibecoding brengt nieuwe mogelijkheden.

AI geeft organisaties nieuwe mogelijkheden om software sneller en doelgerichter te (laten) ontwikkelen. Vibecoding is daarvan de meest zichtbare vorm: bestuurders kunnen er veel eerder mee leren wat werkt en wat niet. Zij moesten lange tijd beslissingen nemen over digitale oplossingen die nog niet bestonden. Zij kregen businesscases, architectuurplaten, ontwerpen en planningen voorgelegd, maar konden moeilijk ervaren hoe een oplossing in de praktijk zou werken. Vibecoding verandert dat. Een eerste prototype hoeft niet langer het resultaat te zijn van een omvangrijk ontwikkeltraject. Het kan juist worden ingezet vóórdat zo'n traject begint.

Dat biedt organisaties een aantal nieuwe mogelijkheden.

Innovatie krijgt meer snelheid en momentum

Een idee dat binnen korte tijd zichtbaar wordt, is gemakkelijker intern te bespreken, te verbeteren en verder te brengen. Dat voorkomt dat kansrijke initiatieven lang in analyses en besluitvorming blijven hangen.

Experimenteren wordt toegankelijker

Niet ieder prototype hoeft naar productie. Vibecoding kan ook worden gebruikt om AI, nieuwe interfaces of digitale werkwijzen te verkennen. Dat levert kennis, inspiratie en vaak ook enthousiasme op.

Minder risico op kostbare misinvesteringen

Een organisatie hoeft niet eerst een volledig ontwikkeltraject te financieren om te ontdekken dat een idee niet werkt. Doordat er veel eerder gevalideerd wordt, is veel sneller helder wát er eigenlijk gebouwd moet worden. Een vroeg prototype kan tijdig aantonen dat een initiatief moet worden bijgestuurd of gestopt, waardoor het vervolgtraject met minder risico en tegen lagere kosten kan worden uitgevoerd.

Experimenteren mag ook gewoon leuk zijn

Er is nog een voordeel dat in bestuurlijke discussies over efficiency en rendement gemakkelijk onderbelicht blijft: experimenteren met technologie kan ook gewoon leuk zijn. Bestuurders, medewerkers en domeinexperts kunnen zelf ervaren wat AI mogelijk maakt. Een idee dat tijdens een bijeenkomst ontstaat, kan dezelfde dag nog zichtbaar worden. Dat levert energie, nieuwsgierigheid en creativiteit op.

Deze voordelen gelden vooral in de vroege fase van een idee. Hoe die balans verschuift naarmate een idee volwassener wordt, komt in het volgende hoofdstuk aan bod.

Wanneer experimenteren gevolgen krijgt.

Doordat de drempel om AI in te zetten binnen het softwareontwikkelproces zo laag wordt, ontstaat er ook een nieuwe verantwoordelijkheid: niet iedere toepassing die eenvoudig kan worden gemaakt, kan zonder meer op dezelfde manier worden gebruikt.

De risico's hangen sterk af van het doel en de context van de software. Een prototype met fictieve gegevens dat na een middag weer wordt verwijderd, vraagt om andere waarborgen dan een klantportaal dat persoonsgegevens verwerkt. Een persoonlijk hulpmiddel voor één medewerker is iets anders dan een applicatie waarvan een heel bedrijfsproces afhankelijk wordt.

Vibecoding is dus niet per definitie onveilig of onverantwoord. Het risico ontstaat wanneer een experiment steeds serieuzer wordt gebruikt en daarmee ongemerkt de experimentfase ontgroeit, terwijl de software niet voor dat zwaardere doel is ontworpen. Die overgang kan snel plaatsvinden, maar ook geleidelijk, over een periode van maanden of zelfs jaren.

Een prototype valideert alleen wat je het vraagt te valideren

Vibecoding maakt het mogelijk om aannames vroeg te toetsen, en dat verkleint het risico op een verkeerde oplossingsrichting aanzienlijk. Maar dat voordeel geldt alleen voor de aannames die daadwerkelijk zijn getoetst. Een prototype bevestigt niet vanzelf dat het onderliggende probleem volledig en juist is geformuleerd; het laat zien of de gekozen aanpak werkt voor het scenario dat is voorgelegd.

Stel dat een organisatie een toepassing ontwikkelt die klantaanvragen automatisch beoordeelt en bepaalt welke dossiers direct kunnen worden afgehandeld en welke extra controle nodig hebben.

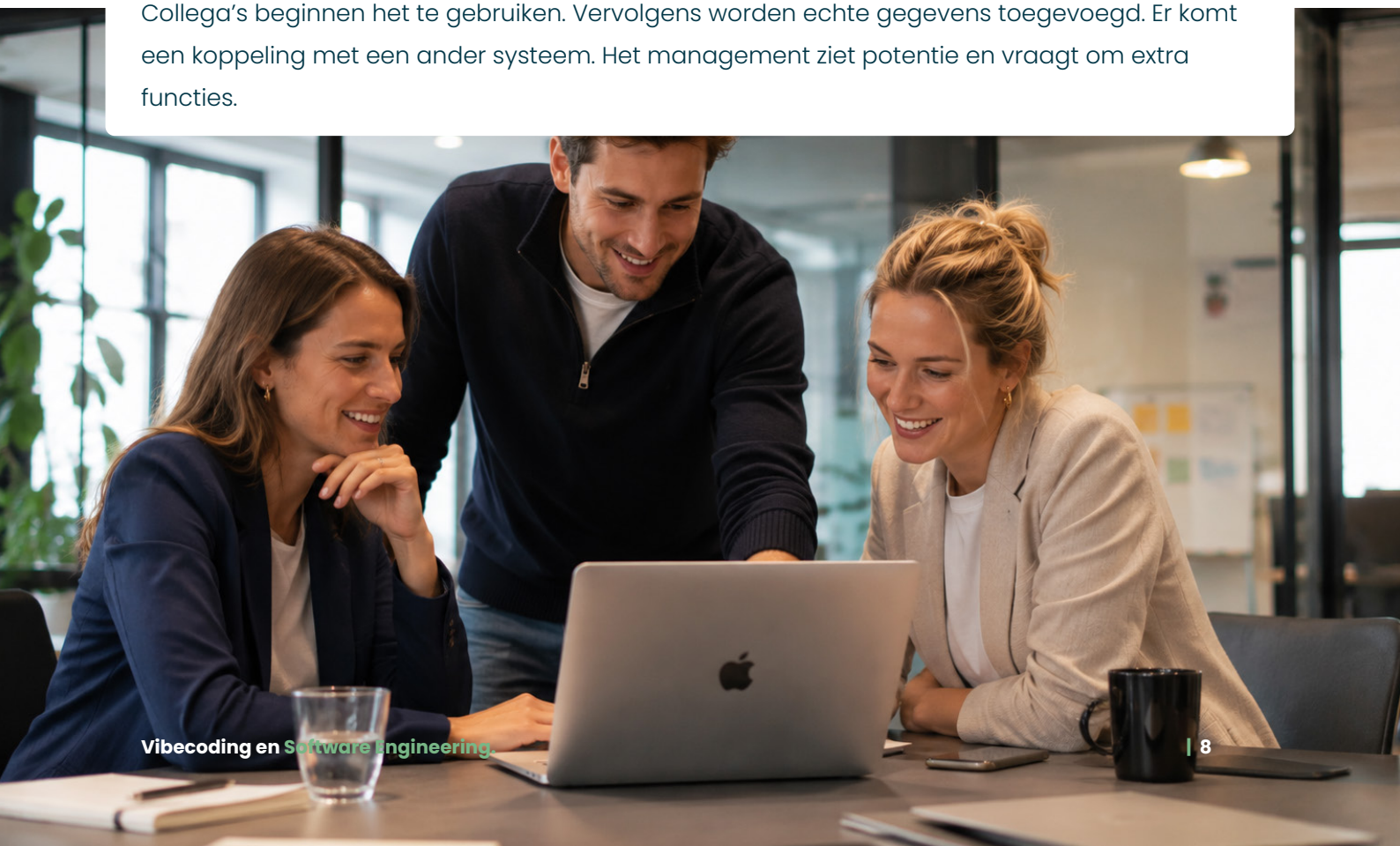
Wanneer de selectiecriteria onvolledig zijn, uitzonderingen ontbreken of historische data bestaande fouten bevat, kan een prototype dat er overtuigend uitziet toch de verkeerde dossiers goedkeuren of risicovolle gevallen missen. Juist omdat niemand die specifieke uitzondering had voorgelegd om te testen. Hoe verder zo'n toepassing verweven raakt met een bedrijfsproces, hoe moeilijker en kostbaarder het wordt om die fout later te herstellen.

Dit is precies waar requirements engineering en domeinkennis waarde toevoegen, ook wanneer een organisatie zelf al veel domeinkennis in huis heeft. Domeinkennis alleen garandeert niet dat alle aannames, uitzonderingen en randgevallen expliciet worden gemaakt voordat ze in een prototype worden vastgelegd. Dat vraagt om een aparte vaardigheid: gestructureerd doorvragen op wat er niet gezegd is. Een requirements engineer voegt dus niet toe wát het proces inhoudelijk moet doen, maar zorgt ervoor dat de juiste vragen zijn gesteld vóórdat het prototype als validatie wordt beschouwd. Zonder die stap valideert een organisatie mogelijk overtuigend het verkeerde scenario.

Een goed prototype is dan ook geen miniatuurversie van een al besloten eindproduct. Het is een gedeeld denkmodel waarmee gebruikers, bestuurders en engineers gezamenlijk onderzoeken of het probleem goed is begrepen, de processen kloppen, welke uitzonderingen bestaan en waar technische of organisatorische risico's ontstaan.

Wanneer een experiment ongemerkt een product wordt

De grens tussen experiment en productiesysteem wordt zelden tijdens één formeel besluit overschreden. Meestal gebeurt het geleidelijk. Een medewerker bouwt een intern hulpmiddel. Collega's beginnen het te gebruiken. Vervolgens worden echte gegevens toegevoegd. Er komt een koppeling met een ander systeem. Het management ziet potentie en vraagt om extra functies.



Enkele maanden later ondersteunt de tijdelijke applicatie een proces waarvan medewerkers of klanten afhankelijk zijn. Vanaf dat moment ontstaan vragen die tijdens het experiment waarschijnlijk niet zijn beantwoord. Wie is eigenaar van de applicatie? Wie mag bij de productiedata? Welke handelingen worden gelogd? Wat gebeurt er bij een storing? Kunnen wijzigingen veilig worden doorgevoerd? En zijn de gebruikte componenten nog wel veilig en worden ze nog ondersteund? De software kan op dat moment technisch prima werken en veilig zijn, maar is ontwikkeld voor een andere context dan waarin het inmiddels wordt gebruikt.

Daar ontstaat technical debt: toekomstige kosten en risico's die voortkomen uit keuzes die op het moment zelf snel en logisch leken, maar niet zijn gemaakt met langdurig gebruik in gedachten.

Zichtbaar functioneren is nog geen goede software

AI-modellen kunnen indrukwekkend snel code genereren die een gewenste functie uitvoert. Maar

geschiktheid voor daadwerkelijk in productie gaan, bestaat uit veel meer dan zichtbaar functioneren.

Software moet niet alleen werken, maar ook begrijpelijk, veilig, schaalbaar, aanpasbaar en testbaar zijn; kenmerken die ook terugkomen in de internationale kwaliteitsstandaard ISO/IEC 25010. Alleen dan kan een organisatie erop vertrouwen dat de toepassing ook bij groei, storingen en toekomstige wijzigingen betrouwbaar blijft functioneren.

Het Veracode 2025 GenAI Code Security Report onderzocht code gegenereerd door meer dan honderd AI-modellen op 80 coding tasks, getoetst tegen bekende kwetsbaarheidscategorieën (CWE) en de OWASP Top 10. Daaruit bleek dat 45 procent van de onderzochte codevoorbeelden niet door de securitytests kwam. De code kon syntactisch correct zijn en ogenschijnlijk doen wat was gevraagd, terwijl de beveiliging tekortschoot.

Ook software engineers vertrouwen AI-output niet blind. Onderzoek van Sonar onder ruim 1.100 professionele developers (State of Code Developer Survey 2026) laat een vergelijkbaar patroon zien: 96 procent vertrouwt niet volledig dat AI-gegenereerde code functioneel correct is, terwijl slechts 48 procent AI-code altijd controleert voordat deze wordt gecommit. Wantrouwen alleen leidt dus niet vanzelf tot verificatie; dat moet in het proces worden afgedwongen, niet aan individuele discipline worden overgelaten. De Stack Overflow Developer Survey 2025 bevestigt het beeld: onder meer dan 49.000 respondenten wantrouwt 46 procent de nauwkeurigheid van AI-tools, tegenover 33 procent die er wel op vertrouwt.

Die terughoudendheid is rationeel. Veel tekortkomingen zijn tijdens een demonstratie niet zichtbaar, zoals gebrekkige toegangscontrole, onveilig beheer van API-sleutels, of tests die alleen het ideale scenario controleren.

Het risico wordt groter wanneer de maker de gegenereerde code zelf niet kan beoordelen. Een ervaren ontwikkelteam kan AI output beoordelen, testen en corrigeren. Een niet technische vibecoder ziet vooral dat de applicatie lijkt te werken.

Architecturale wildgroei: de rekening bij latere wijzigingen

Een AI-model kan snel nieuwe functionaliteit toevoegen, maar bewaakt niet vanzelfsprekend hoe de codebase als geheel groeit. Zonder voldoende architectonische sturing kunnen structuren, afhankelijkheden en verantwoordelijkheden steeds verder versnipperen. Daardoor wordt de software na verloop van tijd moeilijker te begrijpen, testen, onderhouden en veranderen.

De praktijk in volwassen codebases laat dan ook vaak een genuanceerder beeld zien dan demonstraties met nieuwe, losstaande applicaties. In bestaande systemen moeten software engineers rekening houden met eerdere keuzes, architectuur, koppelingen, kwaliteitseisen en functionaliteit die niet verloren mag gaan. Meer gegenereerde code staat in zo'n omgeving niet automatisch gelijk aan meer geleverde waarde.

Wanneer AI-inzet zich daadwerkelijk terugbetaalt

Naast de architecturale kant is er een aparte, meer financiële vraag: wanneer levert het gebruik van AI in het ontwikkeltraject een organisatie daadwerkelijk iets op? Het gaat dan niet om het aantal regels gegenereerde code of alleen de lage kosten van een AI-model, maar om de waarde die daar tegenover staat: hoeveel ontwikkeltijd wordt bespaard, hoeveel aannames worden eerder getoetst, en hoeveel onnodig ontwikkelwerk wordt voorkomen?

Een prototype dat honderd euro aan AI-gebruik kost, maar voorkomt dat een team maanden aan de verkeerde oplossing werkt, is economisch zeer goedkoop. Een prototype dat nauwelijks iets kost, maar zonder duidelijk doel eindeloos wordt aangepast, kan juist verspilling zijn.

De juiste maatstaf is daarom niet hoeveel code een organisatie produceert of hoe goedkoop een eerste versie tot stand komt. Het gaat erom hoe snel zij waardevolle veranderingen veilig kan realiseren en de software daarna verantwoord kan blijven verbeteren.

Nieuwe mogelijkheden verantwoord benutten.

Het antwoord is niet om AI te vermijden of vibecoding te verbieden. De oplossing is om AI juist te integreren in de volledige ontwikkelcyclus, maar altijd in combinatie met de juiste menselijke expertise (human in the loop).

Wat dat in de praktijk betekent, valt terug te brengen tot een klein aantal principes. Niet als checklist, maar als samenhangende manier van werken waarin proces, techniek en organisatie elkaar versterken.

Eerst het probleem, dan de oplossing

Een AI-agent bouwt wat wordt gevraagd, niet noodzakelijk wat nodig is. Verantwoord gebruik begint daarom niet bij het genereren van code, maar bij het scherp krijgen van het probleem. Requirements engineers, domeinexperts en software engineers onderzoeken samen wat een organisatie werkelijk nodig heeft, welke uitzonderingen bestaan en waar technische of organisatorische risico's zitten, voordat dat wordt vertaald naar functionaliteit.

Context is de belangrijkste hefboom

Een generiek AI-model kent de organisatie niet. Het weet niet vanzelf welke wetgeving van toepassing is, welke architectuurkeuzes eerder zijn gemaakt, hoe bestaande systemen samenwerken of welke gegevens gevoelig zijn. Naarmate AI structureler wordt ingezet, verschuift de aandacht daarom van prompt engineering naar context engineering: het bewust beschikbaar maken van gevalideerde requirements, domeinkennis, architectuurprincipes, securitybeleid en eerdere besluiten. Dat maakt van AI geen alwetende engineer, maar het verkleint wel de ruimte waarin het model op basis van aannames moet improviseren.

Guardrails begrenzen wat een AI-agent zelfstandig mag doen

Context bepaalt wat een AI-agent weet; guardrails bepalen wat een AI-agent mag, onafhankelijk van wat er in de prompt of instructies staat. Een agent krijgt daarom ook technische grenzen mee: geen onbeperkte toegang tot data, systemen en productieomgevingen, maar werken binnen vooraf vastgestelde kaders. Architectuurregels, beveiligingsbeleid en ontwikkelstandaarden worden waar mogelijk automatisch getoetst of afgedwongen, niet eenmalig bij oplevering, maar bij elke wijziging opnieuw.

Samen zorgen de juiste context en de juiste grenzen ervoor dat een toepassing die aanvankelijk snel vooruitging, ook na de tiende of vijftigste aanpassing niet moeilijker en duurder wordt om te veranderen.

Kwaliteit blijft mensenwerk, ook als de uitvoering deels wordt geautomatiseerd

Omdat AI-output nooit twee keer precies hetzelfde is, kan kwaliteitsborging nooit afhangen van het model alleen. Geautomatiseerd testen, code review en statische analyse blijven het fundament, ook wanneer AI het grootste deel van de code genereert. Het uitvoeren daarvan is geen mensenwerk, maar het inrichten, interpreteren en verantwoordelijk zijn voor de uitkomst wel: engineers hoeven niet iedere regel zelf te schrijven, maar blijven verantwoordelijk voor de gemaakte keuzes, de samenhang van de oplossing en de kwaliteit van het uiteindelijke systeem. AI ondersteunt het vakmanschap, maar neemt die verantwoordelijkheid niet over.

Een expliciet moment, geen geleidelijke glijbaan

Zoals eerder beschreven ontstaat een experiment zelden bewust tot een productiesysteem, dat gebeurt meestal geleidelijk. Verantwoord gebruik van AI vraagt daarom om een bewust omslagpunt: een moment waarop een organisatie vaststelt dat een toepassing niet langer een vrijblijvend experiment is, en expliciet regelt wie eigenaar is, wie bij de gegevens mag, hoe wijzigingen veilig worden doorgevoerd, wat er gebeurt bij een storing, en wat de toepassing bij daadwerkelijk gebruik gaat kosten. Zonder dat moment schuift een organisatie ongemerkt van een goedkoop experiment naar een kostbare afhankelijkheid.

Hoe dit samenkomt in de praktijk

De voorgaande principes gelden voor het volledige traject van idee tot productiesysteem. De Solution Studio van Avisi is waar dat traject begint.

In de Solution Studio werken requirements engineers, software-engineers en domeinexperts van de klant samen om analyse, requirements, ontwerp en prototyping te versnellen. In enkele uren ontstaat niet alleen een eerste klikbaar prototype, maar ook een uitgewerkt requirementsdocument dat richting geeft aan het vervolg.

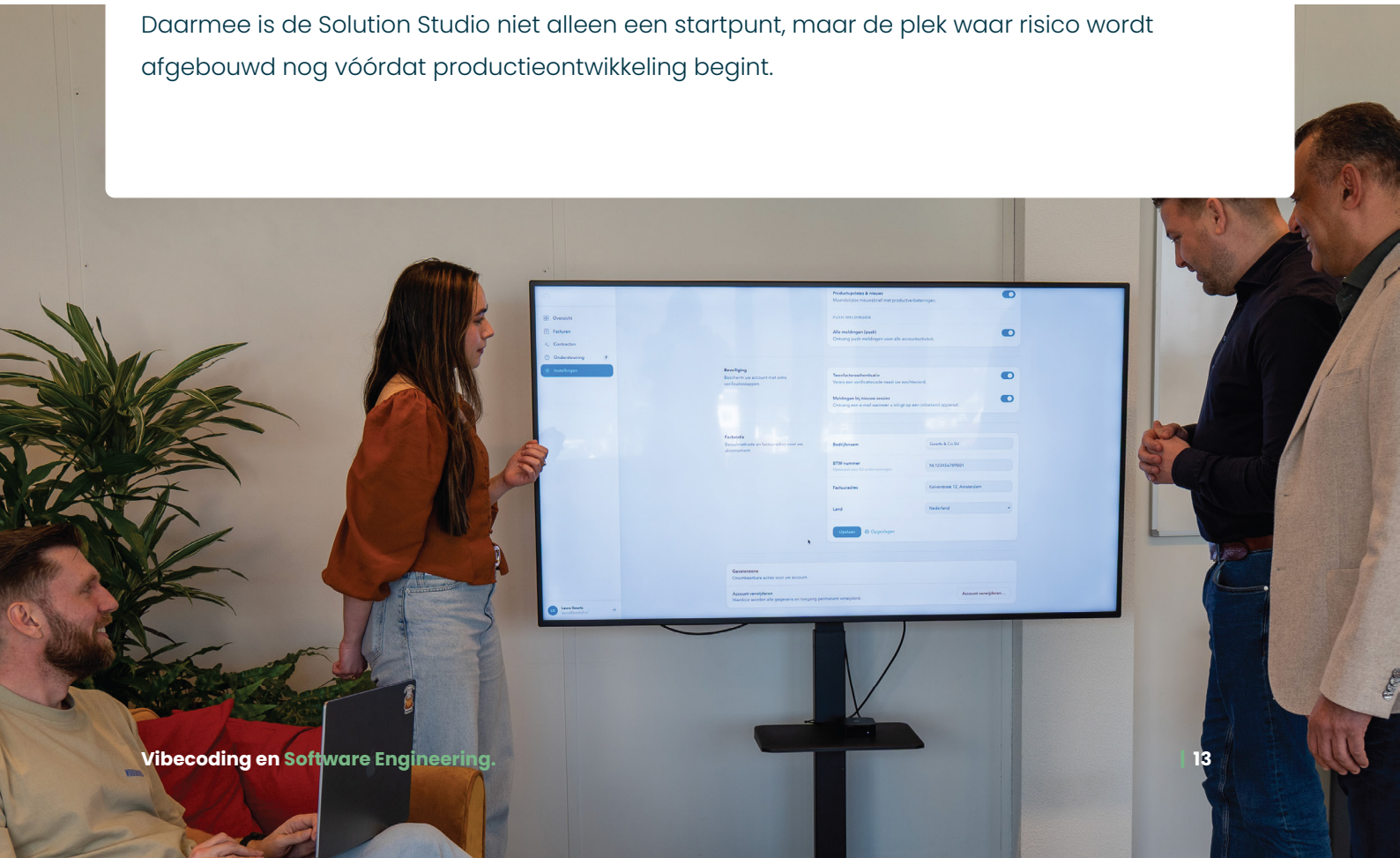
De waarde daarvan zit in vier opbrengsten:

1. Scherpere en gevalideerde requirements
2. Zicht op aannames, uitzonderingen en gebruikersbehoeften
3. Een tastbare oplossingsrichting
4. Inzicht in wat nodig is om de toepassing verantwoord te realiseren

Wanneer een organisatie na validatie wil doorgaan, begint de weg naar productie. AI verdwijnt dan niet uit het proces, maar de manier waarop het wordt ingezet, verandert wel: de gerichte, requirements-gedreven aanpak van de Solution Studio maakt plaats voor AI-assisted engineering, professionele softwareontwikkeling waarbij engineers AI doelgericht inzetten binnen diezelfde architectuur-, security- en kwaliteitskaders. AI helpt dan bijvoorbeeld bij het genereren en verbeteren van code, het schrijven van tests, het analyseren van bestaande systemen en het opsporen van kwetsbaarheden, terwijl engineers verantwoordelijk blijven voor de keuzes en de kwaliteit van het resultaat.

Een zorgvuldig uitgevoerde Solution Studio maakt deze volgende fase eenvoudiger: de gegenereerde code is al opgebouwd in dezelfde taal, techniek en frameworks als de latere productiesoftware, en het probleem, de requirements en de aannames zijn al doorgrond. De Solution Studio en AI-assisted engineering zijn daarmee geen concurrerende benaderingen, maar opeenvolgende fasen van hetzelfde traject: eerst ontdekken wat het waard is om te bouwen, en dat vervolgens veilig, schaalbaar en beheersbaar realiseren.

Daarmee is de Solution Studio niet alleen een startpunt, maar de plek waar risico wordt afgebouwd nog vóórdat productieontwikkeling begint.



Tot slot.

In de inleiding van dit whitepaper stond een belofte: vernieuwende ideëen kunnen tegenwoordig sneller dan ooit op een scherm zichtbaar worden. Die belofte blijft overeind. Vibecoding heeft daadwerkelijk veranderd hoe snel een organisatie kan ontdekken wat werkt en wat niet.

Maar snelheid en betrouwbaarheid zijn geen tegenpolen, mits een organisatie weet op welk moment de een moet plaatsmaken voor de ander. Bestuurders die dat onderscheid vroeg en bewust maken, hoeven niet te kiezen tussen tempo en zekerheid. Zij bepalen zelf wanneer welke aanpak nodig is, en houden daarmee grip op wat hun organisatie bouwt, ongeacht hoe snel dat gaat.

Avisi denkt daar graag in mee, vanaf het eerste prototype tot en met het systeem waar een organisatie jarenlang op kan bouwen. Benieuwd of de Solution Studio ook voor uw organisatie een waardevolle eerste stap is? In een vrijblijvende intake verkennen we samen welk vraagstuk zich daarvoor leent, en wat een eerste prototype voor uw organisatie kan opleveren.

John Mohamed.
Business Development Manager
+31 6 51 57 72 36
john.mohamed@avisi.nl

Meer informatie

www.avisi.nl/solutionstudio