

GENERATIEVE AI IN HET PROGRAMMEERONDERWIJS

KANSEN, UITDAGINGEN EN DE STAND VAN ZAKEN IN VLAANDEREN

Aantal woorden: 14.305

Maxime Debosschere

Studentennummer: 02413416

Promotor: Prof. dr. Kris Coolsaet

Masterpraktijkproef (9SP) voorgelegd tot het behalen van de graad van
Educatieve Master in de Wetenschappen en Technologie: Informatica

Academiejaar: 2025 – 2026, Educatieve Masteropleiding

Generatieve AI in het programmeeronderwijs

Kansen, uitdagingen en de stand van zaken in Vlaanderen

Abstract

Generatieve artificiële intelligentie heeft zich op korte tijd stevig verankerd in softwareontwikkeling en beïnvloedt ook steeds vaker het programmeeronderwijs. Deze masterpraktijkproef onderzoekt de implicaties hiervan voor het leren en onderwijzen van programmeren en de wijze waarop lesgevers hiermee omgaan in hun onderwijspraktijk.

Het eerste luik bestaat uit een literatuurstudie over de invloed van generatieve AI op het aanleren van programmeervaardigheden en over didactische strategieën die hierbij kunnen worden ingezet. De literatuur beschrijft kansen op het vlak van ondersteuning, evaluatie, differentiatie en motivatie. Tegelijk worden uitdagingen aangehaald op het vlak van academische integriteit, afhankelijkheid en bias.

Het tweede luik omvat een vragenlijstonderzoek bij programmeerlesgevers in Vlaanderen ($N = 37$), met aandacht voor hun gebruik, ervaringen en opvattingen met betrekking tot generatieve AI. De resultaten wijzen erop dat generatieve AI inmiddels zowel bij lesgevers als bij lerenden een vaste plaats heeft verworven. De visies op het gebruik ervan lopen echter sterk uiteen: sommige lesgevers benadrukken het belang van een stevige programmeerbasis en kiezen voor een restrictieve aanpak, terwijl anderen generatieve AI beschouwen als een onmisbaar onderdeel van hedendaags programmeren en deze bewust integreren in hun onderwijspraktijk. Respondenten benadrukken daarnaast het groeiende belang van vaardigheden zoals probleemanalyse, het kritisch beoordelen en debuggen van code en het doordacht aanwenden van AI-hulpmiddelen.

Deze studie concludeert dat generatieve AI het programmeeronderwijs niet overbodig maakt, maar wel vraagt om een heroriëntatie van leerdoelen, didactiek en evaluatie.

Inhoudsopgave

Abstract	i
1 Inleiding	1
1.1 Situering	1
1.1.1 Generatieve AI in softwareontwikkeling	1
1.1.2 Gevolgen voor softwareontwikkelaars	2
1.1.3 Gevolgen voor de arbeidsmarkt	3
1.2 Onderzoeksvragen	4
1.3 Begripsafbakening	5
2 Literatuurstudie	7
2.1 Kansen	7
2.1.1 Ondersteuning van het onderwijsproces	7
2.1.2 Ondersteuning van het evaluatieproces	9
2.1.3 Nieuwe didactische benaderingen	11
2.1.4 Betere differentiatie door inzicht in AI-gebruik	12
2.1.5 Meer motivatie voor lerenden	15
2.1.6 Betere leerresultaten door prompttraining	16
2.2 Uitdagingen	17
2.2.1 Fraude	17
2.2.2 Bias en onbetrouwbaarheid	21
2.2.3 Afhankelijkheid	22
3 Resultaten van de bevraging	25
3.1 Methodologie	25
3.2 Profiel van de respondenten	26
3.3 Gebruik van generatieve AI in het programmeeronderwijs	26
3.3.1 Gebruik door lesgevers	26
3.3.2 Gebruik door lerenden in de lescontext	27

3.3.3	Gebruik door lerenden buiten de lescontext	27
3.4	Toelating van generatieve AI in het programmeeronderwijs . .	28
3.4.1	Toelating in de lescontext	28
3.4.2	Toelating tijdens evaluatie	29
3.4.3	Toelating buiten de lescontext	30
3.5	Percepties over generatieve AI in het programmeeronderwijs .	31
3.5.1	Kansen	31
3.5.2	Uitdagingen	32
3.6	Implicaties voor het programmeeronderwijs	35
3.6.1	Vaardigheden van lerenden die belangrijker worden . .	35
3.6.2	Aanpassingen in de lescontext	36
3.6.3	Aanpassingen aan programmeeropdrachten	37
3.6.4	Aanpassingen aan evaluatie	38
3.6.5	Veranderende rol van lesgevers	39
3.6.6	Professionalisering van lesgevers	40
4	Discussie	41
4.1	Reflecties van de auteur	41
4.2	Synthese van de literatuur	44
4.3	Synthese van de vragenlijst	47
4.4	Beperkingen van het onderzoek	50
4.5	Verder onderzoek	50
5	Conclusie	53
5.1	Beantwoording van de onderzoeksvragen	53
5.2	Aanbevelingen	54
	Literatuurlijst	55
A	Vragenlijst: vragen	63
B	Vragenlijst: antwoorden	71
C	Verklaring omtrent het gebruik van generatieve AI	91

1. Inleiding

1.1 Situering

1.1.1 Generatieve AI in softwareontwikkeling

Lange tijd gold de algemene opvatting dat activiteiten zoals het ontwikkelen van software uitsluitend door mensen konden worden uitgevoerd. Recente ontwikkelingen in artificiële intelligentie hebben deze aanname echter ingrijpend veranderd: generatieve AI is vandaag in staat om nieuwe inhoud, zoals code, te creëren op een niveau dat moeilijk te onderscheiden is van menselijk werk (Feuerriegel et al., 2024).

Deze technologie heeft zich dan ook op korte tijd stevig verankerd in de softwareontwikkelingspraktijk. Zo gebruikt ongeveer de helft van de professionele softwareontwikkelaars dergelijke hulpmiddelen dagelijks in hun werkzaamheden, terwijl nog eens dertig procent ze wekelijks of minder frequent inzet (Stack Overflow, 2025). Deze technologie wordt daarbij voornamelijk gebruikt voor het schrijven van routinematige code, het opzoeken van ontwikkelgerelateerde informatie, het omzetten van code naar andere programmeertalen, het genereren van codecommentaar of documentatie en het samenvatten van recente codewijzigingen (Bedrina, 2025).

De ondersteuning van dergelijke taken kan ontwikkelaars toelaten zich sterker te richten op probleemoplossing en andere activiteiten op een hoger abstractieniveau, wat kan bijdragen aan een hogere productiviteit, verbeterde softwarekwaliteit, snellere prototyping en voortdurende innovatie (Banh & Strobel, 2023). Tegelijk brengt het gebruik van dergelijke hulpmiddelen ook risico's met zich mee, zoals een overschatting van de veiligheid van geschreven code (Perry et al., 2023), een daling in codekwaliteit en onderhoudbaarheid, vaardigheidsverlies en een afname van het plezier dat ontwikkelaars ontleen aan het zelfstandig oplossen van programmeerproblemen (Chang et al., 2025).

Chang et al. (2025) classificeren het gebruik van generatieve AI bij professionele programmeurs in drie gebruiksmodi: bij *AI-assisted coding* ondersteunt generatieve AI de ontwikkelaar bij specifieke programmeertaken, zoals codegeneratie, debugging, documentatie en informatieopzoeking, terwijl de programmeur zelf de controle behoudt over het ontwikkelproces; bij *vibe coding* verschuift de nadruk van het schrijven van code naar het formuleren van doelen in natuurlijke taal, waarbij generatieve AI de volledige implementatie genereert op basis van iteratieve interacties met de gebruiker; en bij *agentic coding* delegeren ontwikkelaars substantiële programmeertaken aan autonome AI-agenten, die zelfstandig ontwikkelactiviteiten plannen, coördineren en uitvoeren, waaronder het configureren van ontwikkelomgevingen, aanpassen van bestaande broncode, uitvoeren van tests en beheren van implementatieworkflows (Chang et al., 2025). Van deze drie gebruiksmodi is AI-assisted coding veruit het meest gangbaar in de praktijk (Stack Overflow, 2025). Volgens Sengul et al. (2024) wordt generatieve AI niet alleen ingezet voor programmeertaken, maar ook voor opleidingsdoeleinden, deskundig advies en onderhoudsactiviteiten.

1.1.2 Gevolgen voor softwareontwikkelaars

Het takenpakket van junior softwareontwikkelaars bestaat traditioneel uit relatief eenvoudige en vaak repetitieve programmeeropdrachten, zoals het oplossen van eenvoudige bugs, het testen van software, het doorvoeren van kleine functionele wijzigingen en onderhoudswerk (Osmani, 2025; Stokel-Walker, 2024). Dergelijke taken zijn essentieel voor beginnende ontwikkelaars om relevante vaardigheden op te bouwen en praktijkervaring te verwerven (Stokel-Walker, 2024). Tegelijk gaat het om werkzaamheden die aansluiten bij domeinen waarin generatieve AI vandaag vaak wordt ingezet. Hierdoor komen zowel de rol als de verwachtingen ten aanzien van (junior) ontwikkelaars onder druk te staan.

Wellicht zullen werkgevers van beginnende softwareontwikkelaars verwachten dat zij AI benutten om sneller en zelfstandiger kerncompetenties te ontwikkelen, mogelijk zonder daarbij op persoonlijke begeleiding van een mentor te kunnen rekenen (Stokel-Walker, 2024). Twee derde van de softwareontwikkelaars gaat er dan ook van uit dat werkgevers in de nabije toekomst competenties in het gebruik van AI-tools zullen vereisen (Bedrina, 2025). Bovendien zou mentorschap net in de omgekeerde richting kunnen

verlopen: volgens Miao (2026) ondersteunen de meeste junior ontwikkelaars in de Verenigde Staten nu al hun meer ervaren collega's bij het leren werken met AI-tools en AI-werkprocessen.

Becker et al. (2023) stellen dat softwareontwikkeling in toenemende mate zal worden gekenmerkt door automatisch gegenereerde code, wat wijst op een verschuiving van het schrijven van code naar het lezen en evalueren ervan. In dit verband beschrijft Yegge (2024) programmeren als een proces van iteratieve promptverfijning, waarbij ontwikkelaars herhaaldelijk hun probleemcontext verduidelijken en verfijnen om code te genereren die beantwoordt aan de vooropgestelde functionele vereisten, en deze vervolgens integreren in een bestaande codebase. De rol van de ontwikkelaar verschuift daarbij van het schrijven van code naar het beoordelen, valideren en verfijnen van AI-gegenereerde oplossingen. Yegge noemt deze werkwijze *Chat Oriented Programming* en verwacht dat zij zal uitgroeien tot een dominante ontwikkelpraktijk. Tegen deze achtergrond benadrukt Osmani (2025) het belang van vaardigheden die minder vatbaar zijn voor automatisering, zoals debuggen, systeemontwerp en doeltreffend samenwerken.

Verschillende auteurs formuleren verregaande toekomstverwachtingen omtrent deze evolutie. Volgens Yegge (2024) zal generatieve AI op termijn het merendeel van de wereldwijde broncode schrijven. Welsh (2023) stelt dat programmeren volledig overbodig wordt, omdat AI alle programma's zal schrijven terwijl mensen hoogstens een toezichhoudende rol vervullen. In het verlengde daarvan betoogt Rack (2026) dat code, net zoals assembly vandaag, voornamelijk een tussenrepresentatie wordt, waardoor mensen steeds minder routinematig code zullen schrijven en lezen.

1.1.3 Gevolgen voor de arbeidsmarkt

In Vlaanderen is het aantal vacatures voor softwareontwikkelaars de afgelopen jaren afgenomen. Ontwikkelaars die tot voor kort zeer gegeerd waren, merken tegenwoordig dat de vraag naar hun diensten terugloopt (Decock, 2025). Deze trend wordt op twee manieren verklaard: een correctie op de aanwervingspiek vlak na de coronapandemie (Deckmyn, 2025) en de opmars van AI (Decock, 2025). Steeds meer bedrijven kiezen namelijk voor een *AI-first*-strategie, waarbij taken in eerste instantie met behulp van AI worden benaderd en pas indien nodig door ontwikkelaars worden overgenomen (Kandaswamy, 2025; Peters, 2025; Roose, 2025).

Volgens vertegenwoordigers uit de Vlaamse IT-sector zijn door AI minder softwareontwikkelaars per analist nodig, maar verhoudingsgewijs wel meer hogere profielen (Deckmyn, 2025). Een gelijkaardig beeld is zichtbaar in de Verenigde Staten, waar de vraag naar ervaren ontwikkelaars stijgt terwijl de vraag naar juniorprofielen afneemt (Miao, 2026). Ook in Amerikaanse media wordt de druk op instapfuncties voor pas afgestudeerden in verband gebracht met AI, waarbij zelfs de term *AI job apocalypse* wordt gehanteerd (Roose, 2025).

Tegelijkertijd kan AI ook een argument vormen om net wél onervaren ontwikkelaars aan te werven, aangezien hun inwerktijd korter is dan voorheen (Miao, 2026) en zij vaak een voortrekkersrol opnemen bij de adoptie van AI binnen organisaties (Siu & Ghosal, 2025).

1.2 Onderzoeksvragen

De voorgaande situatieschets toont aan dat generatieve AI een ingrijpende impact heeft op het domein van softwareontwikkeling. Daarnaast is generatieve AI inmiddels ook doorgedrongen tot het onderwijs, waar zowel lesgevers als lerenden dergelijke hulpmiddelen gebruiken voor uiteenlopende doeleinden. Generatieve AI kan het leerproces ondersteunen en nieuwe mogelijkheden bieden voor feedback en begeleiding, maar roept tegelijk vragen op over evaluatie, afhankelijkheid en academische integriteit.

Deze masterpraktijkproef onderzoekt welke implicaties generatieve AI heeft voor het programmeeronderwijs en hoe lesgevers hiermee omgaan in hun onderwijspraktijk. Vanuit deze doelstelling worden de volgende onderzoeksvragen geformuleerd:

- OV1.** Hoe beïnvloedt generatieve AI het programmeeronderwijs, en welke didactische strategieën kunnen worden ingezet om de kansen ervan te benutten en de uitdagingen ervan te beheersen?
- OV2.** In welke mate en op welke manier wordt generatieve AI gebruikt binnen het programmeeronderwijs in Vlaanderen, en welke percepties en ervaringen hebben lesgevers met betrekking tot dit gebruik?

De eerste onderzoeksvraag wordt behandeld aan de hand van een literatuurstudie (hoofdstuk 2), terwijl de tweede onderzoeksvraag empirisch

wordt benaderd via een vragenlijstonderzoek bij programmeerlesgevers in Vlaanderen (hoofdstuk 3). De bevindingen worden vervolgens besproken (hoofdstuk 4), waarna de belangrijkste conclusies worden geformuleerd (hoofdstuk 5).

1.3 Begripsafbakening

In deze masterpraktijkproef worden enkele kernbegrippen in een specifieke betekenis gehanteerd. Met *lerende* wordt verwezen naar leerlingen, studenten en cursisten binnen het formeel onderwijs. De term *lesgever* verwijst naar personen die programmeren onderwijzen binnen het secundair onderwijs, het hoger onderwijs of het volwassenenonderwijs.

Daarnaast wordt een onderscheid gemaakt tussen artificiële intelligentie (AI) en generatieve artificiële intelligentie. Onder AI wordt het brede domein van systemen verstaan die taken uitvoeren waarvoor doorgaans menselijke intelligentie vereist is, zoals aanbevelingssystemen, beeldherkenning en spamfilters. Generatieve AI vormt een deelgebied van AI en verwijst naar systemen die nieuwe inhoud kunnen genereren, zoals tekst, code of afbeeldingen. Bekende toepassingen hiervan zijn onder meer ChatGPT, Gemini, Copilot en DALL-E.

2. Literatuurstudie

Deze literatuurstudie bespreekt het gebruik van generatieve AI in het programmeeronderwijs, met bijzondere aandacht voor de didactische kansen en uitdagingen die deze technologie met zich meebrengt. Het merendeel van de beschikbare literatuur situeert zich daarbij in het hoger onderwijs (Deriba et al., 2024). Een synthese van de belangrijkste inzichten volgt in paragraaf 4.2.

2.1 Kansen

2.1.1 Ondersteuning van het onderwijsproces

De didactische meerwaarde van *pair programming*, een gevestigde praktijk in softwareontwikkeling, werd reeds geruime tijd geleden aangetoond: lerenden die in paren programmeren leveren programma's van hogere kwaliteit af en behalen hogere slaagpercentages, ongeacht hun vaardigheidsniveau (McDowell et al., 2002). Generatieve AI kan in dit verband fungeren als een *pair programmer* en lerenden op diverse manieren ondersteunen (Malinka et al., 2023; Prather et al., 2023). Zo tonen recente onderzoeksresultaten aan dat deze aanpak de intrinsieke motivatie kan verhogen, programmeerangst kan verminderen en programmeerprestaties kan verbeteren ten opzichte van individueel programmeren (Fan et al., 2025).

Volgens Malinka et al. (2023) kan generatieve AI ook fungeren als een geavanceerde vorm van *rubber duck debugging*, doordat lerenden problemen en mogelijke oplossingsstrategieën met het systeem kunnen bespreken.

Daarnaast kan generatieve AI volgens Wieser et al. (2023) worden ingezet als persoonlijke tutor die lerenden ondersteunt met gerichte uitleg en aanvullende oefenkansen, al hangt de bruikbaarheid van dergelijke ondersteuning sterk af van duidelijk geformuleerde leerdoelen. Prather et al. (2023) wijzen

er bovendien op dat dergelijke gepersonaliseerde ondersteuning lerenden kan helpen om leerproblemen zelfstandig aan te pakken, in het bijzonder wanneer zij anders een aanzienlijke inspanning van de lesgever zouden vergen.

Generatieve AI kan modeloplossingen genereren ter referentie voor lerenden. Codegeneratiemodellen zijn bovendien in staat om meerdere correcte oplossingen voor eenzelfde probleem te genereren, waardoor lerenden inzicht verwerven in alternatieve aanpakken en de onderliggende concepten beter begrijpen. Deze variatie in oplossingsstrategieën wordt in de literatuur beschouwd als een belangrijk element in het leerproces (Becker et al., 2023; Finnie-Ansley et al., 2022).

Een studie naar AI-ondersteuning bij programmeertaken toont aan dat deelnemers generatieve AI als behulpzaam ervaren, omdat het een eerste aanzet biedt bij het oplossen van programmeertaken en zo de drempel van een lege pagina vermindert. Zelfs wanneer de gegenereerde code niet correct is, geeft deze steeds een richting aan van waaruit verder gewerkt kan worden (Vaithilingam et al., 2022). Verder rapporteren Vukojičić en Krstić (2023) dat ondersteuning door ChatGPT kan leiden tot uitgebreider gebruik van commentaar in code en een uniformere programmeerstijl, waardoor code leesbaarder en begrijpelijker wordt.

Het gebruik van generatieve AI tijdens programmeertaken kan een verschuiving vereisen van het schrijven van code naar het herschrijven, refactoren en debuggen ervan. Tegelijk biedt dit de mogelijkheid om lerenden te ondersteunen in het behouden van voortgang binnen een authentieke leeromgeving, waarin het evalueren, aanpassen en uitbreiden van bestaande code mogelijk belangrijker is dan het zelf schrijven van elke regel code (Becker et al., 2023).

Een veelvoorkomend obstakel bij beginnende lerenden zijn moeilijk interpreteerbare compilerfouten. Generatieve AI kan dergelijke foutmeldingen vertalen naar begrijpelijke natuurlijke taal en in veel gevallen ook suggesties voor oplossingen voorstellen (Leinonen et al., 2023; Pankiewicz & Baker, 2024). Hierdoor kunnen belangrijke leerbarrières mogelijk worden verlaagd (Becker et al., 2023).

Naast ondersteuning voor lerenden biedt generatieve AI ook mogelijkheden voor lesgevers. Voor hen vormt het namelijk een aanzienlijke uitdaging om een brede waaier aan kwalitatief hoogstaand leermateriaal te ontwikkelen dat aansluit bij de interesses van lerenden, en om daarbij gedetailleerde uitleg te bieden op verschillende abstractieniveaus (Denny et al., 2024). Bovendien

vergroot de snelle verspreiding van oplossingen voor bestaande opdrachten de behoefte aan nieuwe oefeningen en bijhorende evaluaties (Sarsa et al., 2022).

Generatieve AI kan worden ingezet om leermiddelen te genereren, zoals uitleg met bijhorende codevoorbeelden, oefeningen en geautomatiseerde tests voor het verifiëren van antwoorden (Becker et al., 2023; Denny et al., 2022; Sarsa et al., 2022). Prather et al. (2023) beschrijven nog andere toepassingen voor lesgevers, waaronder het afstemmen van lesmateriaal op uiteenlopende leerdoelen en moeilijkheidsgraden, verkennend literatuuronderzoek en het herschrijven en samenvatten van tekstfragmenten.

2.1.2 Ondersteuning van het evaluatieproces

In de onderzoeksliteratuur worden drie manieren onderscheiden om code te evalueren: traditionele evaluatie door de lesgever, onderlinge beoordeling door lerenden en geautomatiseerd testen. Traditionele evaluatie steunt rechtstreeks op de kennis en expertise van de lesgever, maar is een tijdrovend proces. Onderlinge beoordeling stimuleert de betrokkenheid van lerenden, maar is gevoelig voor subjectieve beoordelingen en kwaliteitsverschillen. Klassiek geautomatiseerd testen maakt onmiddellijke en objectieve evaluatie mogelijk, maar blijft beperkt tot het controleren van vooraf gedefinieerde uitkomsten en houdt minder rekening met aspecten zoals het creatieve proces of de kwaliteit van de code (Jukiewicz, 2024).

Op basis van een meta-analyse concluderen Prather et al. (2023) dat generatieve AI mogelijkheden biedt om beoordeling en feedback in het programmeeronderwijs te ondersteunen, onder meer door het analyseren van het werk van lerenden, het corrigeren van programmeerfouten en het aanbieden van gepersonaliseerde hulp.

In deze context onderzoekt Jukiewicz (2024) of generatieve AI niet alleen de correctheid van code kan beoordelen, maar ook subtielere aspecten zoals codekwaliteit, leesbaarheid, efficiëntie en het naleven van programmeerstandaarden. De resultaten suggereren dat generatieve AI potentieel heeft als hulpmiddel voor automatische beoordeling en feedback, maar wijzen ook op beperkingen zoals hallucinaties, lichte variaties tussen opeenvolgende beoordelingen, de financiële kost en een blijvende nood aan menselijke controle (Jukiewicz, 2024).

Meerdere studies rapporteren een sterke positieve correlatie tussen de

scores van generatieve AI en menselijke beoordelaars (Diamantopoulos et al., 2025; Jukiewicz, 2024; Wieser et al., 2023). Dit suggereert dat beide beoordelingsvormen opdrachten in grote mate op een vergelijkbare manier rangschikken. Hoewel de overeenstemming groot is, stellen sommige studies ook systematische verschillen vast. Bij Jukiewicz (2024) kent generatieve AI gemiddeld lagere scores toe dan de lesgever, waardoor het systeem in deze studie als een strengere beoordelaar naar voren komt. Volgens de auteur kunnen deze verschillen deels verklaard worden doordat lesgevers imperfecte maar werkende oplossingen soms hoger waarderen en bepaalde fouten over het hoofd zien. Diamantopoulos et al. (2025) rapporteren daarentegen dat generatieve AI de neiging heeft om zeer lage scores iets te verhogen en zeer hoge scores iets te verlagen. Deze bevinding wijst erop dat generatieve AI menselijke beoordelingen redelijk nauwkeurig benadert, maar mogelijk minder onderscheid maakt aan de uitersten van de beoordelingsschaal. Gezien de beperkte omvang van de besproken onderzoeken moeten deze bevindingen echter voorzichtig worden geïnterpreteerd.

Automatische code-evaluatie biedt volgens Jukiewicz (2024) verschillende potentiële voordelen. Zo kan deze aanpak lerenden onmiddellijk feedback geven, de werkdruk van lesgevers verminderen, grote groepen schaalbaar evalueren en een efficiënter leerproces ondersteunen. Daarnaast kan deze aanpak bijdragen aan het aanleren van een goede programmeerstijl en testvaardigheden, fraude helpen detecteren en gerichte terugkoppeling op programmeeropdrachten aanbieden. Vooral de mogelijkheid om gepersonaliseerde feedback te genereren onderscheidt generatieve AI van klassieke testgebaseerde evaluatie, omdat zij naast een score ook inhoudelijke toelichting en verbeteringsuggesties kan formuleren (Jukiewicz, 2024).

Een studie van Diamantopoulos et al. (2025) bevestigt dit potentieel. De auteurs tonen aan dat generatieve AI niet alleen kan worden ingezet voor beoordeling, maar ook voor het genereren van gedetailleerde formatieve feedback met concrete verbeterpunten en toelichting bij gemaakte fouten. Bovendien ontvingen lerenden deze feedback vrijwel onmiddellijk na het indienen van hun oplossing, waardoor de evaluatiecyclus aanzienlijk werd verkort ten opzichte van traditionele evaluatie. Zviel-Girshin (2024) rapporteert dat onmiddellijke respons en feedback tot de meest genoemde voordelen van AI-tools behoren.

Hoewel deze resultaten veelbelovend zijn, wijst experimenteel onderzoek

ook op de grenzen van AI-gegenereerde terugkoppeling. Er et al. (2025) vergeleken in een programmeercursus feedback van een lesgever met AI-gegenereerde feedback. Lerenden beoordeelden de terugkoppeling van de lesgever als nuttiger en boekten grotere verbeteringen dan lerenden die AI-gegenereerde feedback ontvingen. Volgens de auteurs pleit dit voor hybride feedbackmodellen waarin de schaalbaarheid van AI wordt gecombineerd met de inhoudelijke expertise van lesgevers.

Daarnaast kan generatieve AI lesgevers ondersteunen door individuele beoordelingen te aggregeren tot een overzicht van de prestaties van een volledige groep. Veelvoorkomende fouten, misconcepties en aandachtspunten kunnen automatisch worden geïdentificeerd en samengevat in een klasrapport. Dergelijke inzichten kunnen lesgevers helpen om gerichte remediëring te voorzien en hun onderwijs af te stemmen op de noden van de groep (Diamantopoulos et al., 2025).

2.1.3 Nieuwe didactische benaderingen

Generatieve AI biedt mogelijkheden om het programmeeronderwijs anders vorm te geven en de nadruk te verschuiven van syntactische implementatie naar probleemoplossing, analyse en ontwerp.

Programmeeronderwijs voor beginners richt zich traditioneel op syntaxis en fundamentele programmeerconcepten, waarbij het vaak geruime tijd vergt voordat lerenden deze basisvaardigheden beheersen. Wanneer generatieve AI wordt aangewend voor implementatie op laag niveau, ontstaat de mogelijkheid om de focus te verleggen naar algoritmisch denken en probleemoplossing op een hoger abstractieniveau. Dit kan ertoe leiden dat programmeeronderwijs zich in een vroeg stadium meer richt op conceptueel begrip, terwijl een diepgaande behandeling van syntaxis en implementatiedetails pas later wordt geïntroduceerd (Becker et al., 2023).

Om lerenden op hogere niveaus van de taxonomie van Bloom te betrekken, kunnen bijvoorbeeld codefragmenten van wisselende kwaliteit worden gegenereerd, die vervolgens door lerenden worden geanalyseerd en verbeterd. In dezelfde lijn stelt Beale (2025) dat generatieve AI vooral leerzaam wordt wanneer opdrachten lerenden verplichten om AI-gegenereerde oplossingen uit te leggen, te bekritisieren en te wijzigen. Zonder dergelijke didactische inbedding bestaat volgens de auteur het risico dat lerenden essentiële stappen in het leerproces overslaan.

Ook Chang et al. (2025) beschrijven concrete didactische werkvormen waarmee deze benadering kan worden gerealiseerd. De auteurs bevelen aan om lerenden AI-gegenereerde oplossingen kritisch te laten beoordelen, logische fouten te laten opsporen en code te laten debuggen. Verder kunnen lesgevers lerenden vragen om dergelijke oplossingen toe te lichten, ontwerpkeuzes te verantwoorden of oplossingen zonder AI-hulp te reconstrueren. Een mogelijke aanpak bestaat erin om generatieve AI aanvankelijk gecontroleerd in te zetten als leerondersteuning, gevolgd door voldoende oefenkansen zonder AI-ondersteuning om het begrip en de zelfstandigheid van lerenden te versterken.

Daarnaast wijzen de auteurs erop dat generatieve AI ook mogelijkheden creëert om binnen het programmeeronderwijs meer aandacht te besteden aan vaardigheden op een hoger abstractieniveau. Dergelijke vaardigheden kunnen onder meer worden ontwikkeld via opdrachten rond het analyseren van vereisten, het uitwerken van specificaties en het vergelijken van alternatieve oplossingsstrategieën. Dit kan bijdragen aan de ontwikkeling van inzicht in softwarearchitectuur en het systematisch opdelen van complexe problemen. In dat verband kan volgens de auteurs ook projectgebaseerd leren baat hebben bij de inzet van generatieve AI. Wanneer codegeneratiemodellen kleine implementatietaken overnemen, doen lerenden vroeger ervaring op met grotere en meer realistische softwaresystemen. Dergelijke projectcontexten bieden kansen om vaardigheden zoals systeemdecompositie, code review, testen, integratie en deployment te oefenen, naast professionele competenties zoals samenwerken, communiceren en projectmanagement. Bovendien vormen zij een natuurlijke omgeving om AI-gegenereerde code kritisch te evalueren en te valideren (Chang et al., 2025).

2.1.4 Betere differentiatie door inzicht in AI-gebruik

Inzicht in de manier waarop lerenden met generatieve AI interageren, kan lesgevers ondersteunen bij het afstemmen van didactische strategieën op individuele leerbehoeften (Güner & Er, 2025).

Malinka et al. (2023) onderscheiden drie gebruiksmodi op basis van de voorkennis van lerenden: *copy&paste*, waarbij de benodigde kennis ontbreekt en antwoorden rechtstreeks worden overgenomen; *interpretatie*, waarbij lerenden de gegenereerde output trachten te begrijpen en selectief aanpassen; en *assistentie*, waarbij zij actief interageren met AI, fouten corrigeren en

gerichte vragen stellen om de output te verfijnen.

Deze gebruiksmodi tonen aan dat generatieve AI op uiteenlopende manieren kan worden ingezet. Die diversiteit komt ook tot uiting in de concrete taken waarvoor AI wordt geraadpleegd. Zo blijkt uit Lepp en Kaimre (2025) dat lerenden AI-assistenten onder meer gebruiken voor debugging, het begrijpen van codevoorbeelden, syntaxisvragen, conceptuele uitleg, taakdecompositie en het omzetten van code naar andere programmeertalen. Ook Novak en Maček (2026) rapporteren dat codegeneratie, debugging en code-uitleg tot de meest voorkomende toepassingen behoren.

Groothuijsen et al. (2024) identificeren zes redenen waarom lerenden een beroep doen op generatieve AI bij het oplossen van programmeeropdrachten, die hieronder in dalende volgorde van frequentie worden besproken. Lerenden gebruiken generatieve AI vooral om *code uit te leggen* in gewone taal, met als doel de functionaliteit of werking van een codefragment beter te begrijpen. Daarnaast wordt AI ingezet om *code te genereren*. Deze code wordt doorgaans niet integraal overgenomen, maar gebruikt als startpunt of inspiratiebron. Volgens de lerenden komt dit doordat ChatGPT geen correcte oplossing kan genereren,¹ terwijl de lesgever dit toeschrijft aan een terughoudendheid om dergelijke code over te nemen, of een gebrek aan inzicht in de output ervan.

Verder gebruiken lerenden generatieve AI om *fouten op te sporen en code te debuggen*. Daarbij hanteren lerenden twee strategieën: enerzijds wordt gevraagd hoe fouten zelf kunnen worden opgespoord, anderzijds wordt AI ingezet om fouten of bugs te identificeren. Lerenden geven aan dat zij hun code uiteindelijk meestal zelf moeten aanpassen, omdat AI fouten of bugs doorgaans niet volledig kan verhelpen.²

Een ander belangrijk gebruik van generatieve AI is ter *ondersteuning van het conceptueel begrip* van programmeren. Lerenden vragen uitleg over programmeerconcepten en -technieken, of onderzoeken de voor- en nadelen van verschillende manieren om code te schrijven. Ten slotte wordt generatieve AI ook aangewend om *code te optimaliseren*, door bestaande code te laten herschrijven met het oog op vereenvoudiging of efficiëntieverhoging, en om *wiskundige problemen op te lossen*. Los van deze gebruiksdoelen raadplegen sommige lerenden ook andere digitale informatiebronnen, zoals online fora

¹Aangezien generatieve AI zich zeer snel ontwikkelt, is dit argument mogelijk niet langer van toepassing.

²Idem.

en professionele samenwerkingsplatformen, om de output van ChatGPT te verifiëren (Groothuijsen et al., 2024).

De bovenstaande bevindingen tonen aan dat het gebruik van generatieve AI door lerenden niet uniform is. Güner en Er (2025) benaderen dit vraagstuk vanuit een ander perspectief en onderscheiden vijf AI-interactieprofielen bij lerenden tijdens programmeeropdrachten: *AI-afhankelijke codegeneratoren* plakken volledige of gedeeltelijke instructies van een programmeeropdracht rechtstreeks in een AI-chatbot en streven naar snelle, directe antwoorden, met weinig betrokkenheid bij het eigenlijke probleemoplossingsproces; *AI-afhankelijke codegeneratoren en -verfijners* plakken volledige of gedeeltelijke instructies van een programmeeropdracht rechtstreeks in AI, maar formuleren vervolgens prompts om antwoorden te verfijnen in functie van de opdracht, of voor bijkomende uitleg of verificatie; *AI-collaboratieve programmeurs* kopiëren niets uit de opdracht, maar stellen gerichte vragen over specifieke programmeerkennis die ze vervolgens toepassen op de opdracht; *AI-ondersteunende codeverfijners* werken eerst zelfstandig aan de programmeeropdracht en raadplegen daarna AI voor revisie, feedback, foutopsporing of optimalisatie; en *AI-onafhankelijke programmeurs* tonen een hoge mate van zelfstandigheid en probleemoplossend vermogen door opdrachten te voltooien zonder daarbij van AI afhankelijk te zijn.

De auteurs beargumenteren verder dat voorkennis van programmeren een significante invloed heeft op deze profielen: lerenden met meer voorkennis gebruiken AI vaker aanvullend en reflectief, terwijl beginners sneller geneigd zijn tot direct en minder betrokken gebruik. Dit onderstreept het belang van het vroegtijdig herkennen van verschillen in voorkennis en AI-interactie, aangezien deze inzichten lesgevers kunnen ondersteunen bij het maken van didactische keuzes rond de inzet van generatieve AI (Güner & Er, 2025).

Daarnaast wijzen Novak en Maček (2026) erop dat lerenden niet alleen verschillen in de manier waarop zij generatieve AI gebruiken, maar ook in hun opvattingen over de plaats ervan binnen het programmeeronderwijs. Studenten die een gedeeltelijke toelating van generatieve AI verkiezen, verwijzen vaker naar het belang van leerkansen en een goed begrip van code, terwijl voorstanders van een ruimere toelating het gebruik van generatieve AI vaker motiveren vanuit de relevantie ervan voor de beroepspraktijk. Ook de context waarin programmeren wordt onderwezen lijkt een rol te spelen: Keuning et al. (2024) stellen vast dat studenten het gebruik van generatieve AI minder

aanvaardbaar achten wanneer het verwerven van programmeervaardigheden een centraal leerdoel vormt, dan wanneer programmeren voornamelijk wordt ingezet ter ondersteuning van andere leerdoelen.

2.1.5 Meer motivatie voor lerenden

Boguslawski et al. (2025) onderzoeken de relatie tussen generatieve AI en leermotivatie in het programmeeronderwijs. Daarbij maken ze een onderscheid tussen intrinsieke motivatie (zoals nieuwsgierigheid naar programmeren en de voldoening die voortkomt uit het voltooien van programmeeropdrachten) en extrinsieke motivatie (die wordt aangestuurd door externe factoren zoals deadlines, evaluaties of het vooruitzicht op carrièremogelijkheden). Beide vormen van motivatie spelen een belangrijke rol bij het leren programmeren.

De auteurs stellen dat generatieve AI niet veel kan betekenen voor extrinsieke motivatie. Deze vorm van motivatie komt namelijk best tot stand door sociale interactie met mensen, zoals andere lerenden en vertrouwde experts. Daarentegen kan generatieve AI wel bijdragen aan intrinsieke motivatie door: *autonomie te vergroten* (lerenden laten werken in de programmeertaal en het project van hun keuze, ondersteuning op maat bieden afgestemd op individuele interesses, vaardigheden en leervoorkeuren); *competentie en zelf-effectiviteit te vergroten* (moeilijke concepten uitleggen, vragen beantwoorden die lerenden misschien gênant vinden om te stellen, lerenden sneller laten doorgroeien naar het schrijven van complexe programma's); en *negatieve gevoelens zoals verveling en frustratie te verminderen* (tijd minimaliseren die besteed wordt aan saaie en repetitieve taken, het aantal momenten beperken waarop lerenden vastlopen op moeilijke concepten of leeruitdagingen).

Voor lesgevers formuleren Boguslawski et al. (2025) enkele concrete aanbevelingen. De motivatie van lerenden in het programmeeronderwijs kan versterkt worden door: generatieve AI te kaderen als hulpmiddel om sneller vooruitgang te boeken en beter voorbereid te zijn op de verwachtingen van de hedendaagse industrie; lerenden aan te leren hoe zij kwalitatief hoogwaardige antwoorden uit generatieve AI kunnen verkrijgen (zie paragraaf 2.1.6); lerenden te instrueren in het lezen, begrijpen en debuggen van AI-gegenereerde code; lerenden te trainen in het kritisch controleren van de accurateheid van AI-gegenereerde antwoorden; lerenden met een gemiddeld niveau te begeleiden bij het kiezen van een programmeertaal en project dat aansluit bij hun interesses, waarbij generatieve AI vervolgens ondersteuning op maat kan

bieden; kansen te creëren voor samenwerking in duo's of groepen ter bevordering van sociaal leren; en evaluatievormen aan te bieden en te begeleiden die afgestemd zijn op het niveau en de interesses van lerenden (opdrachten, examens en groepsleren).

2.1.6 Betere leerresultaten door prompttraining

Prompts zijn instructies of vragen die aan een generatief AI-model (zoals ChatGPT) worden gegeven om een specifiek type antwoord te genereren (Jukiewicz, 2024). Goed geformuleerde prompts kunnen bijdragen aan een effectiever leerproces, doordat generatieve AI onderliggende concepten duidelijker kan toelichten en relevante voorbeeldcode kan aanbieden ter aanvulling van de officiële documentatie (Malinka et al., 2023). Dit principe vormt de basis van het concept *prompt engineering*, waarbij prompts doelgericht worden geformuleerd om zo nauwkeurig, coherent en relevant mogelijke antwoorden te verkrijgen (Lo, 2023). Door deze vaardigheid te ontwikkelen, kunnen gebruikers AI-taalmodellen effectiever inzetten (Lo, 2023; Yilmaz & Yilmaz, 2023).

Prompt engineering kan worden ontwikkeld via gerichte prompttraining. Een gestructureerde aanpak hiervoor wordt geboden door het *CLEAR Framework for Prompt Engineering* (Lo, 2023), dat een standaardmethode aanreikt voor het formuleren van doeltreffende vragen aan AI-taalmodellen. Het raamwerk steunt daarbij op vijf kernprincipes, namelijk: *concise* (kort en duidelijk, zonder overbodige informatie en met nauwkeurige instructies), *logical* (gestructureerd en coherent, met een logische opbouw en samenhang tussen ideeën), *explicit* (voorzien van duidelijke outputcriteria, met precieze instructies over de gewenste inhoud en het uitvoerformaat), *adaptive* (gericht op aanpassingsvermogen, waarbij via experimenten met formuleringen, zinsbouw en instellingen een evenwicht wordt gezocht tussen creativiteit en focus) en *reflective* (gericht op continue evaluatie, waarbij AI-output kritisch wordt beoordeeld op basis van gebruikersfeedback en eigen evaluaties, met als doel verbeterpunten te identificeren en prompts iteratief verder te verfijnen).

Het belang van dergelijke gestructureerde prompttraining wordt empirisch ondersteund door Garg et al. (2025), die aantonen dat prompttraining positief samenhangt met betere leerresultaten, een hogere betrokkenheid en een hogere kwaliteit van AI-interacties. Lerenden die voorafgaand prompttraining ontvingen, presteren significant beter bij data-analyse- en programmeertaken

dan lerenden zonder dergelijke training, en dit over verschillende cognitieve niveaus van de taxonomie van Bloom heen. Daarnaast vertonen lerenden die generatieve AI gebruiken zonder voorafgaande prompttraining betere prestaties dan lerenden zonder AI-ondersteuning.

Naast algemene principes voor prompttraining formuleren Manley et al. (2024) ook concrete richtlijnen die de leerwaarde van interacties met generatieve AI kunnen verhogen. Zo bevelen zij aan om eigen oplossingspogingen in prompts op te nemen, vervolgvragen te stellen over onbegrepen onderdelen en AI voornamelijk in te zetten voor afgebakende deelproblemen in plaats van volledige oplossingen te laten genereren.

2.2 Uitdagingen

2.2.1 Fraude

Onder fraude in de context van programmeeronderwijs wordt onder meer verstaan: plagiaat (het kopiëren van andermans werk zonder bronvermelding, ongeacht of de oorspronkelijke auteur zich daarvan bewust is), zelfplagiaat (het indienen van eerder werk als nieuw werk), samenspanning (het ongeoorloofd samenwerken bij individueel ingediende opdrachten), uitbesteding (het laten uitvoeren van werk door een andere persoon of dienst), vervalsing (het verzinnen of onjuist voorstellen van beoordeeld werk), en het gebruik van niet-toegestane hulpmiddelen (Fraser, 2014; Prather et al., 2023; Simon et al., 2016).

Het programmeeronderwijs is erg vatbaar voor al deze vormen van fraude. Nog voor de opkomst van generatieve AI stelden onderzoekers vast dat informatica-opleidingen, en dan vooral programmeren, een disproportioneel groot aandeel van plagiaatgevalen in het hoger onderwijs vertegenwoordigen (Fraser, 2014). Dit kan deels worden verklaard door de aard van programmeeropdrachten, waarbij praktijken zoals samenwerking en hergebruik van code, die essentieel zijn in de professionele context, in een academische omgeving vaak als ongeoorloofd worden beschouwd (Simon et al., 2016). Daarnaast hebben programmeeropdrachten doorgaans een beperkt aantal correcte oplossingen, waardoor sterke gelijkenissen tussen antwoorden ontstaan en lerenden zich bewust zijn van de mogelijkheid om bestaande oplossingen over te nemen (Fraser, 2014).

Generatieve AI is steeds beter in staat om autonoom hoogwaardige output, waaronder code, te genereren (Savelka et al., 2023). De kwaliteit ervan is al enkele jaren vergelijkbaar met die van de gemiddelde universiteitsstudent (Finnie-Ansley et al., 2022; Malinka et al., 2023). Bijgevolg kan generatieve AI relatief eenvoudig worden misbruikt in een educatieve context, wat er uiteindelijk toe kan leiden dat lerenden slagen voor vakken zonder de onderliggende leerstof grondig te begrijpen (Malinka et al., 2023). Dit wordt geïllustreerd door Stutz et al. (2023), die een fictieve student succesvol lieten slagen voor een programmeervak in het afstandsonderwijs door uitsluitend gebruik te maken van generatieve AI. Om te voorkomen dat fraude de kwaliteit van afgestudeerden ondermijnt, beschouwen Malinka et al. (2023) aanpassingen aan het onderwijsmodel als onvermijdbaar en hoogdringend. Delphino (2025) stelt vast dat fraude veelvuldig voorkomt en ziet generatieve AI zelfs als een existentiële dreiging voor het informaticaonderwijs.

Preventie

In de onderzoeksliteratuur worden diverse strategieën beschreven om fraude met generatieve AI te voorkomen.

Een eerste preventiestrategie bestaat uit bewustmaking en duidelijke communicatie. Malinka et al. (2023) benadrukken dat lerenden bewust moeten worden gemaakt van de impact van generatieve AI en van het belang van academische integriteit. Daarnaast stellen Stutz et al. (2023) dat universiteiten duidelijke regels moeten opstellen om het gebruik van generatieve AI te reguleren, wat ook relevant lijkt voor andere onderwijsinstellingen. Ook Manley et al. (2024) wijzen op het belang van transparantie en stellen voor om lerenden een eenvoudige manier te bieden om AI-gebruik te documenteren, bijvoorbeeld door vanuit de code te verwijzen naar een transcript van de interactie met het AI-systeem. Volgens de auteurs kan deze aanpak drempels voor het eerlijk rapporteren van AI-gebruik wegnemen.

Een andere belangrijke preventiestrategie betreft het herzien van opdrachten. Traditionele programmeeropdrachten komen onder druk te staan, aangezien generatieve AI-systemen veel typische programmeeropgaven uit inleidende cursussen kunnen oplossen. Daardoor wordt het moeilijker om op basis van dergelijke opdrachten de individuele bijdrage van lerenden te beoordelen (Denny et al., 2024). Volgens Stutz et al. (2023) is het daarom noodzakelijk om opdrachten te herzien die momenteel volledig door genera-

tieve AI kunnen worden opgelost. Dit wijst op het belang van opdrachten waarin niet alleen de geproduceerde code, maar ook begrip, werkwijze en kritische beoordeling van oplossingen centraal staan. Een voorbeeld hiervan wordt beschreven door Becker et al. (2023), waarbij lerenden meerdere correcte oplossingen vergelijken en beoordelen op het vlak van werkwijze en kwaliteit.

Naast de aard van opdrachten staan ook evaluatiepraktijken ter discussie, in het bijzonder de weging en organisatie ervan. Volgens Prather et al. (2023) kan de opkomst van generatieve AI ertoe leiden dat opdrachten die buiten toezicht worden uitgevoerd minder gewicht krijgen binnen de evaluatie, omdat lesgevers minder vertrouwen hebben dat het ingediende werk volledig door de lerenden zelf werd gerealiseerd.

Daarnaast wordt in de praktijk geëxperimenteerd met alternatieve of meer gecontroleerde evaluatievormen. Zo pleiten Malinka et al. (2023) ervoor om het gewicht van huiswerkopdrachten op het eindcijfer te beperken en papieren evaluaties in te zetten als mogelijke maatregel tegen fraude. Hampton (2023) beschrijft hoe een programmeercursus overschakelde naar papieren examens nadat werd vastgesteld dat sommige lerenden generatieve AI gebruikten voor programmeeropdrachten. Huang (2023) rapporteert eveneens dat verschillende universiteiten als reactie op generatieve AI meer inzetten op mondelinge examens, groepswork en evaluaties die in de les plaatsvinden. Stutz et al. (2023) suggereren daarnaast om lerenden via videoconferentie hun ingediende opdrachten mondeling te laten toelichten aan de lesgever. Ten slotte wijzen Avouris et al. (2025) op de mogelijkheden van procesgerichte evaluatie, waarbij niet alleen het eindresultaat maar ook het ontwikkelingsproces, de ondervonden uitdagingen en de gevolgde oplossingsstrategieën worden beoordeeld. Volgens de auteurs kan dit de betrokkenheid van lerenden vergroten en oneerlijk gebruik van generatieve AI bemoeilijken.

Sommige onderzoekers gaan nog een stap verder en stellen voor om generatieve AI expliciet te integreren in het onderwijsproces, waarbij lerenden achteraf moeten aantonen dat zij de gegenereerde oplossingen begrijpen. Chung (2026) beschrijft bijvoorbeeld een model waarbij generatieve AI mag worden gebruikt voor programmeeropdrachten, op voorwaarde dat lerenden nadien via gesloten schriftelijke kennistoetsen aantonen dat zij de algoritmen, structuur en implementatiedetails van hun oplossing begrijpen. Ook Stutz

et al. (2023) roepen op tot het verkennen van manieren om generatieve AI in het onderwijsproces te integreren, in plaats van het gebruik ervan volledig te verbieden. Malinka et al. (2023) waarschuwen eveneens voor een louter restrictieve aanpak, omdat die een zinvolle aanwending van generatieve AI kan belemmeren.

Ondanks deze oproepen tot een doordachte integratie van generatieve AI werd op sommige plaatsen aanvankelijk gekozen voor een meer restrictieve aanpak. Zo werd de toegang tot ChatGPT op alle apparaten en netwerken van de openbare scholen van New York City geblokkeerd om fraude en afhankelijkheid tegen te gaan (Rosenblatt, 2023a). Deze maatregel werd echter reeds na enkele maanden teruggedraaid, omdat de aanvankelijke focus op risico's het potentieel en het toekomstige belang van generatieve AI onderschatte (Rosenblatt, 2023b).

Detectie

Reeds vóór de opkomst van generatieve AI werden in het programmeer-onderwijs gespecialiseerde detectiesystemen gebruikt om overeenkomsten tussen oplossingen op te sporen (Sağlam & Schmid, 2025). Generatieve AI maakt het voor lesgevers echter complexer om de individuele bijdrage van lerenden te beoordelen en mogelijk ongeoorloofd gebruik van AI-systemen te herkennen (Bommasani et al., 2021). AI-gegenereerde oplossingen vertonen namelijk vaak zodanige variatie dat zij moeilijk van origineel werk te onderscheiden zijn. Recent onderzoek wijst erop dat geautomatiseerde obfuscatietechnieken een bijkomende uitdaging vormen voor plagiaatdetectie. Waar het verhullen van plagiaat vroeger tijd, programmeerkennis en een goed begrip van de oorspronkelijke code vereiste, kunnen dergelijke technieken tegenwoordig grotendeels automatisch worden toegepast. Daardoor wordt het voor lerenden mogelijk om de detectie van overgenomen code te bemoeilijken zonder zelf ingrijpende wijzigingen aan te brengen (Sağlam & Schmid, 2025). AI-gegenereerde oplossingen vereisen bovendien geen interactie met derden, in tegenstelling tot klassieke vormen van uitbesteding, waarbij communicatie kan worden blootgelegd en oplossingen kunnen worden hergebruikt (Becker et al., 2023).

In de literatuur wordt daarom ook gekeken naar technische hulpmiddelen voor de detectie van AI-gegenereerde inhoud. Malinka et al. (2023) onderzochten een AI-detectietool die tekst analyseert op basis van statis-

tische kenmerken zoals voorspelbaarheid en variatie. Zij stellen vast dat deze tool in hun onderzoek vooral geschikt was voor Engelstalige teksten, terwijl de bruikbaarheid voor niet-Engelstalige teksten beperkt bleek. Stutz et al. (2023) onderzochten een andere AI-detectietool en rapporteren dat deze AI-gegenereerde programmeeropdrachten correct identificeerde. Tegelijk waarschuwen zij dat dergelijke tools niet foutloos zijn en dat hun resultaten voorzichtig moeten worden geïnterpreteerd. Deze voorzichtigheid wordt ondersteund door de bevindingen van Sağlam en Schmid (2025), die aantonen dat AI-detectietools vaak moeite hebben om te bepalen of code al dan niet door AI werd gegenereerd. Hun prestaties nemen verder af wanneer code slechts gedeeltelijk door AI is geschreven, wanneer lerenden samenwerken of wanneer gegenereerde output nadien wordt aangepast.

Denny et al. (2024) stellen vast dat gangbare plagiaatdetectietools vaak tekortschieten bij het detecteren van AI-gegenereerde oplossingen, waardoor technische detectie momenteel geen sluitende oplossing biedt. Ook Sağlam en Schmid (2025) benadrukken dat detectietools hoogstens verdachte patronen kunnen signaleren, maar niet zelfstandig kunnen bepalen of er sprake is van fraude. De uiteindelijke beoordeling blijft afhankelijk van menselijke interpretatie en pedagogische afwegingen. Deze conclusie sluit aan bij de empirische bevindingen van Suh et al. (2025), die vaststellen dat bestaande detectoren voor AI-gegenereerde code momenteel onvoldoende presteren en verdere ontwikkeling vereisen.

2.2.2 Bias en onbetrouwbaarheid

Bias (vertekening) in generatieve AI manifesteert zich ook in de context van codegeneratie. Zo kan de samenstelling van de trainingsdata ertoe leiden dat gegenereerde oplossingen minder goed aansluiten bij de behoeften van beginnende lerenden. Codegeneratiemodellen worden grotendeels getraind op publiek beschikbare code, die niet noodzakelijk representatief is voor het niveau van deze doelgroep. Dergelijke code is namelijk vaak afkomstig van ervaren ontwikkelaars, waardoor gegenereerde oplossingen complexer kunnen zijn dan didactisch wenselijk en minder goed aansluiten bij de beoogde leerdoelen van een cursus (Becker et al., 2023).

Naast bias bestaan er ook bezorgdheden over de kwaliteit van AI-gegenereerde code. Volgens Chen et al. (2021) kunnen codegeneratiemodellen oplossingen voorstellen die ogenschijnlijk correct zijn, maar in werkelijkheid

niet voldoen aan de beoogde functionaliteit. Dit kan onder meer veiligheidsrisico's veroorzaken wanneer onveilige code wordt gegenereerd.

Een recente systematische literatuurstudie van He et al. (2026) toont aan dat AI-gegenereerde code kwaliteitsproblemen kan vertonen op het vlak van onder meer correctheid, beveiliging, robuustheid, onderhoudbaarheid, begrijpelijkheid en efficiëntie. Daarnaast stellen Li et al. (2025) empirisch vast dat een aanzienlijk deel van AI-gegenereerde code beduidend minder robuust is dan vergelijkbare door mensen geschreven code, onder meer door ontbrekende inputvalidatie en gebrekkige foutafhandeling. Verder tonen Fu et al. (2025) aan dat AI-gegenereerde code regelmatig beveiligingskwetsbaarheden bevat.

Deze bevindingen onderstrepen dat de evaluatie van AI-gegenereerde code niet mag worden beperkt tot functionele correctheid, aangezien dergelijke code nog steeds tekortkomingen op het vlak van betrouwbaarheid kan vertonen. Volgens Chen et al. (2021) blijven daarom menselijke controle en kritische evaluatie essentieel bij het gebruik van generatieve AI.

2.2.3 Afhankelijkheid

Een belangrijk risico bij het gebruik van generatieve AI in programmeeronderwijs is een toenemende afhankelijkheid van AI-gegenereerde oplossingen. Lerenden die dergelijke systemen gebruiken om programmeeropdrachten op te lossen, ontwikkelen mogelijk minder diepgaande kennis en probleemoplossingsvaardigheden dan wanneer zij zelf actief naar oplossingen zoeken. Dit is vooral problematisch voor beginnende lerenden. Doordat generatieve AI eenvoudige opdrachten vaak probleemloos kan oplossen, bestaat het risico dat fundamentele programmeerconcepten onvoldoende worden verworven, waardoor zij later moeilijkheden ondervinden bij complexere opdrachten (Malinka et al., 2023).

Ook bij tools die ondersteuning integreren in ontwikkelomgevingen kunnen beginnende lerenden snel gewend raken aan automatisch voorgestelde oplossingen. Daardoor lezen zij probleemstellingen mogelijk minder aandachtig of zelfs helemaal niet en denken zij onvoldoende na over de computationele stappen die nodig zijn om een probleem op te lossen (Becker et al., 2023).

Een mechanisme dat aan afhankelijkheid kan bijdragen, is *cognitive offloading* (cognitieve uitbesteding), waarbij lerenden een deel van hun denken redeneerprocessen uitbesteden aan generatieve AI. Hierdoor kunnen zij sneller tot een oplossing komen zonder alle onderliggende redeneerstappen zelf

uit te voeren. Zo rapporteert Vivian (2025) dat lerenden in AI-ondersteunde programmeertaken hogere prestaties op korte termijn behalen, maar dat dit niet noodzakelijk gepaard gaat met betere resultaten in situaties waarin dergelijke ondersteuning ontbreekt. Deze observatie suggereert dat hogere prestaties niet steeds het gevolg zijn van een overeenkomstige kennis- of vaardigheidsontwikkeling.

Daarnaast stellen Lepp en Kaimre (2025) in een cursus objectgeoriënteerd programmeren vast dat frequenter gebruik van AI-assistenten negatief samenhangt met academische prestaties. Hoewel dit niet noodzakelijk een causaal verband aantoont, roept het vragen op over de mate waarin ongestructureerd AI-gebruik daadwerkelijk bijdraagt aan duurzame leerwinst.

Lerenden kunnen bovendien ten onrechte de indruk krijgen dat zij een concept begrijpen, omdat AI-gegenereerde uitleg en oplossingen vaak coherent en overtuigend overkomen. Dit kan ertoe leiden dat zij hun eigen competenties overschatten en minder geneigd zijn om oplossingen kritisch te analyseren of zelfstandig te doorgronden (Vivian, 2025).

Deze bevindingen suggereren dat generatieve AI in het programmeeronderwijs vooral effectief is wanneer het wordt ingezet als ondersteuning van het leerproces. Een evenwichtige inzet van deze hulpmiddelen, waarbij traditionele probleemoplossingsactiviteiten worden aangevuld in plaats van vervangen, blijft daarom noodzakelijk om programmeervaardigheden en conceptueel begrip op te bouwen (da Silva et al., 2024; Lepp & Kaimre, 2025; Vivian, 2025; Zvieli-Girshin, 2024).

3. Resultaten van de bevraging

Dit hoofdstuk bespreekt de resultaten van een vragenlijstonderzoek bij programmeerlesgevers in Vlaanderen, met als doel inzicht te verkrijgen in hun gebruik van generatieve AI binnen het programmeeronderwijs, alsook in hun percepties en ervaringen. Een synthese van de belangrijkste bevindingen volgt in paragraaf 4.3.

3.1 Methodologie

De vragenlijst werd opgesteld en gepubliceerd via het online enquêteplatform Qualtrics van de Universiteit Gent en was beschikbaar van 8 januari 2026 tot en met 8 mei 2026, gericht op programmeerlesgevers in het Vlaamse onderwijs. Van de 52 inzendingen werden enkel de 37 volledig doorlopen vragenlijsten opgenomen in de analyse ($N = 37$).

Het onderzoeksinstrument bestaat uit 38 vragen, waaronder 7 meerkeuzevragen met meerdere antwoordopties, 3 meerkeuzevragen met één antwoordoptie, 9 symmetrische en gebalanceerde Likertschalen, waarvan 8 met een optioneel vrij invoerveld ter toelichting, en 11 optionele open vragen voor kwalitatieve inzichten.

De bevraging werd anoniem ingevuld, met de vrijblijvende mogelijkheid voor respondenten om hun contactgegevens achter te laten voor verdere opvolging.

In het kader van dit onderzoek werd tevens een Data Management Plan (DMP) opgesteld volgens de richtlijnen van de Universiteit Gent, waarin de verwerking en opslag van de onderzoeksdata worden beschreven.³

De vragenlijst zoals die effectief aan respondenten werd gepresenteerd is opgenomen in bijlage A, en een volledig overzicht van alle antwoorden is terug te vinden in bijlage B.

³DMP ID: 219765

3.2 Profiel van de respondenten

Het merendeel van de respondenten is actief in het secundair onderwijs (89%; $n = 33$). Slechts één respondent is actief in het hoger onderwijs en vier in het volwassenenonderwijs. Eén respondent is actief in twee onderwijscontexten. Gezien deze beperkte spreiding is het niet mogelijk om afzonderlijke conclusies te trekken over verschillen tussen onderwijscontexten.

Er is wel een duidelijke spreiding in de hoeveelheid ervaring met het onderrichten van programmeeronderwijs. Zowel beginnende lesgevers (minder dan 1 jaar: 5%; 1–3 jaar: 24%) als ervaren lesgevers (13–20 jaar: 19%; meer dan 20 jaar: 19%) zijn goed vertegenwoordigd.

Python is met ruime voorsprong de meest onderrichte programmeertaal (73%), gevolgd door C# (35%) en Java en JavaScript (beide 19%).

3.3 Gebruik van generatieve AI in het programmeeronderwijs

3.3.1 Gebruik door lesgevers

Wat het eigen gebruik van generatieve AI door lesgevers betreft, blijkt dat het merendeel deze technologie minstens occasioneel inzet binnen de lespraktijk. Een aanzienlijke groep gebruikt generatieve AI zeer vaak (11%) of regelmatig (32%), terwijl 27% aangeeft dit af en toe te doen. Daartegenover staat dat 8% het zelden gebruikt en 22% nooit. Deze resultaten wijzen op een brede adoptie van deze technologie onder lesgevers, al bestaan er duidelijke verschillen in gebruiksfrequentie.

Bij lesgevers die generatieve AI minstens sporadisch gebruiken ($n = 29$), is ChatGPT veruit de populairste toepassing (83%). Op ruime afstand volgen Gemini (41%), Claude en Copilot (beide 31%). Andere tools, zoals Replit AI (10%), worden slechts beperkt gebruikt.

Binnen dezelfde groep werd ook onderzocht voor welke doeleinden generatieve AI wordt ingezet. De technologie wordt vooral gebruikt voor het genereren of herwerken van toets- en opdrachtmateriaal (90%) en voor het genereren van inspiratie of voorbeeldcode (86%). Daarnaast wordt de technologie ingezet bij de voorbereiding van lesmateriaal of cursusnota's (57%) en voor het verbeteren of optimaliseren van code (48%). Het gebruik voor

het geven van feedback aan lerenden blijft eerder beperkt (14%), evenals andere toepassingen (7%).

3.3.2 Gebruik door lerenden in de lescontext

Wat het gebruik van generatieve AI door lerenden tijdens programmeerlessen betreft, lopen de inschattingen van lesgevers sterk uiteen. Ongeveer een derde van de respondenten (32%) geeft aan dat (bijna) niemand generatieve AI gebruikt tijdens de lessen, terwijl 27% inschat dat (bijna) iedereen hiervan gebruikmaakt. Daarnaast schat 14% dat ongeveer een kwart van de lerenden generatieve AI gebruikt, 16% ongeveer de helft en 11% ongeveer driekwart.

Deze spreiding in het waargenomen gebruik lijkt gedeeltelijk samen te hangen met de visie van de lesgever en de mate waarin het gebruik van deze technologie toegelaten of geïntegreerd wordt binnen het programmeeronderwijs. Uit de toelichtingen blijkt namelijk dat een beperkt gebruik van generatieve AI vaak samenhangt met restrictieve maatregelen, zoals het uitschakelen van internettoegang, het actief monitoren van computerschermen of het expliciet verbieden van generatieve AI.

Bij een hoger waargenomen gebruik lijkt generatieve AI ook niet altijd toegelaten te zijn, maar gebeurt het gebruik soms verborgen en ontbreekt het aan controle. Daarentegen geeft een respondent aan dat deze technologie tijdens lessen en oefeningen bewust wordt toegelaten, maar dat lerenden via functionele en technische demonstraties hun oplossingen moeten kunnen toelichten en verdedigen.

Lerenden die gebruikmaken van generatieve AI tijdens programmeerlessen, doen dit volgens de respondenten regelmatig (40%), af en toe (28%), zeer vaak (20%) en zelden (12%). Twee respondenten merken op dat lerenden sneller naar deze hulpmiddelen grijpen wanneer zij opdrachten niet zelfstandig kunnen oplossen of wanneer zij ergens over twijfelen.

3.3.3 Gebruik door lerenden buiten de lescontext

Volgens de respondenten is het gebruik van generatieve AI buiten de lescontext sterk ingeburgerd bij lerenden: 39% schat dit gebruik als zeer frequent in en nog eens 39% als regelmatig. Daarnaast beschouwt 11% het gebruik als occasioneel, terwijl telkens 6% het gebruik als zelden of nooit inschat.

Een groot aantal respondenten geeft aan geen of aanzienlijk minder

programmeeropdrachten buiten de lescontext te voorzien. Dit hangt samen met de inschatting dat lerenden generatieve AI vrijwel altijd gebruiken bij dergelijke opdrachten, waardoor de authenticiteit van het ingediende werk moeilijk te beoordelen wordt. Daartegenover geven andere respondenten aan dat gebruik buiten de les nauwelijks voorkomt, omdat er eenvoudigweg geen opdrachten meer worden gegeven.

Verder merkt ongeveer een derde van de respondenten op dat het gebruik van generatieve AI vaak herkenbaar is in code of tijdens presentaties. Een lesgever vermoedt echter dat dit gebruik mogelijk overschat wordt, omdat sommige lerenden oplossingen van medeleerlingen overnemen in plaats van zelf hulpmiddelen te gebruiken.

Tot slot merkt een respondent op dat het gebruik van generatieve AI onder lerenden genormaliseerd lijkt.

3.4 Toelating van generatieve AI in het programmeeronderwijs

3.4.1 Toelating in de lescontext

De mate waarin generatieve AI tijdens programmeerlessen wordt toegestaan, blijkt sterk uiteen te lopen. Zo laat 22% van de respondenten dit nooit toe en nog eens 22% slechts zelden. Daarnaast geeft 35% aan generatieve AI occasioneel in te zetten tijdens de lessen, terwijl telkens 11% het gebruik regelmatig of zeer vaak toestaat.

Respondenten bij wie generatieve AI nooit of zelden wordt toegelaten binnen de lescontext, benadrukken voornamelijk het belang van zelfstandig leren programmeren en nadenken over oplossingen. Volgens hen moeten lerenden eerst een voldoende sterke programmeerbasis ontwikkelen alvorens hulpmiddelen zoals generatieve AI te gebruiken. Een respondent vergelijkt dit met het leren hoofdrekenen en cijferen vóór het gebruik van een rekenmachine. Bijkomend wordt aangehaald dat het beperkte aantal lessen onvoldoende ruimte laat om een niveau te bereiken waarbij generatieve AI noodzakelijk zou zijn.

Bij lesgevers die generatieve AI occasioneel toelaten, blijkt het gebruik doorgaans sterk afhankelijk van de context, de opdracht of het leerjaar. Sommige lesgevers geven aan dat het gebruik van generatieve AI moeilijk

volledig te vermijden is, maar beschouwen dit tegelijk als een opportuniteit om lerenden kritisch en doordacht met deze technologie te leren omgaan. Daarbij worden soms expliciete afspraken gemaakt over het gebruik van generatieve AI of worden gerichte prompts voorzien om lerenden redeneringen te laten opbouwen zonder hen onmiddellijk uitgewerkte oplossingen aan te reiken. Verder wordt generatieve AI in sommige gevallen gebruikt om oefeningen te controleren, verbeterde uitwerkingen te verkrijgen of toepassingen verder uit te werken.

Respondenten die generatieve AI regelmatig of zeer vaak toelaten, benaderen deze technologie eerder als een inherent onderdeel van het programmeren. Één lesgever spreekt over een zekere berusting in deze evolutie, terwijl een andere respondent generatieve AI beschouwt als één van de verschillende beschikbare hulpmiddelen naast leerstof, fora en begeleiding door de lesgever. Bij sommige respondenten lijkt bovendien een sterk vertrouwen in de verantwoordelijkheid van lerenden aanwezig: een lesgever verwacht dat lerenden expliciet aangeven wat zij niet zelfstandig konden oplossen, terwijl een andere respondent aangeeft dat lerenden weten dat inzicht belangrijker is dan enkel het verkrijgen van het juiste antwoord.

3.4.2 Toelating tijdens evaluatie

Een grote meerderheid van de respondenten laat het gebruik van generatieve AI tijdens toetsen en examens niet toe: 73% geeft aan dit nooit toe te laten en nog eens 5% slechts zelden. Daarnaast laat 3% het gebruik occasioneel toe tijdens evaluatiemomenten, tegenover respectievelijk 14% en 5% die dit regelmatig en zeer vaak toestaan.

Respondenten die generatieve AI niet toelaten tijdens toetsen of examens, benadrukken vooral de functie van evaluatie als controle op zelfstandige beheersing. Evaluaties moeten volgens hen nagaan in welke mate lerenden zelf opgaven kunnen oplossen en inzicht hebben in fundamentele programmeerlogica, algoritmes en syntax. Één respondent voegt daaraan toe dat het gebruik van generatieve AI de objectiviteit van evaluaties kan ondermijnen, omdat identieke prompts verschillende antwoorden kunnen opleveren.

Bij lesgevers die generatieve AI slechts zelden toelaten, wordt vaak gezocht naar manieren om lerenden zelfstandig te laten redeneren. Zo worden soms bijkomende evaluatie-elementen toegevoegd of examens op papier afgenomen. Tegelijk wordt opgemerkt dat deze technologie moeilijk volledig te weren

blijft, onder meer door geïntegreerde hulpmiddelen zoals Copilot.

De beperkte groep respondenten die generatieve AI regelmatig of zeer vaak toelaat tijdens evaluaties, beschouwt deze technologie eerder als een onderdeel van het programmeerproces. In sommige gevallen worden bijkomende voorwaarden opgelegd, zoals een mondelinge verdediging van de code of het uitvoeren van kleine wijzigingen op verzoek van de lesgever. Hierbij verschuift de nadruk gedeeltelijk van het louter schrijven van code naar het kunnen begrijpen, toelichten en aanpassen ervan.

3.4.3 Toelating buiten de lescontext

De meningen over het toelaten van generatieve AI bij programmeeropdrachten buiten de les blijken sterker verdeeld dan bij evaluaties. Zo laat 15% van de respondenten dit nooit toe en 11% slechts zelden. Daarnaast staat 30% het gebruik occasioneel toe, terwijl telkens 22% aangeeft dit regelmatig of zeer vaak toe te laten. Voor sommige lesgevers was de vraag niet van toepassing omdat ze buiten de les geen programmeeropdrachten geven.

In de toelichtingen staat vooral de beperkte controleerbaarheid centraal. Verschillende respondenten achten een verbod op generatieve AI bij opdrachten buiten de lescontext weinig zinvol, omdat lerenden ook buiten de schoolomgeving toegang hebben tot deze technologie. Bij lesgevers die het gebruik nooit of zelden toelaten, blijft de nadruk vooral liggen op het zelfstandig verwerven van basisvaardigheden. Tegelijk erkennen sommigen dat lerenden generatieve AI alsnog gebruiken bij opdrachten buiten de les.

Respondenten die generatieve AI vaker toelaten buiten de lescontext, beschouwen deze technologie doorgaans als moeilijk te vermijden binnen het programmeeronderwijs. Daarbij wordt benadrukt dat lerenden op een doordachte manier met generatieve AI moeten leren omgaan en dat het gebruik ervan vooral ondersteunend bedoeld is.

Één respondent geeft aan dat generatieve AI bij volledig online onderwijs niet zinvol te verbieden valt, waardoor het gebruik ervan bewust geïntegreerd wordt in de onderwijspraktijk.

3.5 Percepties over generatieve AI in het programmeeronderwijs

3.5.1 Kansen

De antwoorden tonen een genuanceerd beeld van de kansen die generatieve AI kan bieden binnen het programmeeronderwijs. Een terugkerend standpunt is dat generatieve AI een potentieel waardevol hulpmiddel vormt, mits het gebruik ervan voldoende doordacht en begeleid gebeurt.

Meerdere respondenten benadrukken dat generatieve AI het leerproces kan versterken, maar pas nadat lerenden een voldoende sterke programmeerbasis verworven hebben. Een kleine minderheid van lesgevers ziet daarom weinig tot geen kansen voor een zinvolle inzet van deze technologie binnen het secundair onderwijs, maar eerder in het hoger onderwijs of later op de werkvloer.

Veel lesgevers benaderen generatieve AI in de eerste plaats als een ondersteunend leermiddel. Volgens hen kunnen lerenden sneller hulp krijgen, bijkomende uitleg vragen, voorbeelden raadplegen wanneer zij vastlopen en verschillende oplossingswegen verkennen. Ook kan generatieve AI concepten uitleggen in een eenvoudigere of andere taal, wat vooral voor lerenden met een andere thuistaal als waardevol wordt beschouwd. Aanvullend kan deze technologie ondersteuning bieden bij het zelfstandig verwerken van leerstof, bijvoorbeeld na afwezigheid, of oefeningen met foutieve code genereren die lerenden zelf moeten corrigeren.

Verder wijzen verschillende respondenten erop dat generatieve AI kan bijdragen aan meer zelfstandigheid en een versterking van het leerproces. Lerenden zouden sneller kunnen groeien doordat zij gemakkelijker toegang krijgen tot bijkomende informatie en ondersteuning. Generatieve AI kan lerenden helpen om redeneringen op te bouwen bij moeilijkere oefeningen en programmeerconcepten te verkennen die nog niet expliciet aan bod kwamen. Een respondent merkt bovendien op dat generatieve AI de drempel tot programmeervaardigheden kan verlagen, waardoor programmeren mogelijk ook in meer studierichtingen geïntegreerd kan worden.

Andere respondenten leggen de nadruk eerder op efficiëntie en kwaliteit. Volgens hen kan generatieve AI lerenden helpen om sneller oplossingen te ontwikkelen en code te produceren die korter of efficiënter is. Daardoor

verschuift de aandacht gedeeltelijk van het zelf schrijven van code naar het begrijpen, evalueren, debuggen en aanpassen van programma's. Ook zou generatieve AI het mogelijk maken om extra uitdagende elementen aan projecten toe te voegen en grotere of complexere projecten uit te werken. Dit kan bijdragen aan meer voldoening over het bereikte eindresultaat. Daarnaast wordt vermeld dat generatieve AI lerenden kan ondersteunen bij het afstemmen van hun vaardigheden op actuele ontwikkelingen binnen het vakgebied.

Vanuit het perspectief van lesgevers worden eveneens verschillende kansen genoemd. Generatieve AI kan ondersteuning bieden bij het ontwikkelen van lesmateriaal, het bedenken van opgaven, het genereren van voorbeeldcode of data, en bij evaluatie en begeleiding.

3.5.2 Uitdagingen

De uitdagingen die respondenten ervaren bij het gebruik van generatieve AI in het programmeeronderwijs situeren zich voornamelijk rond het behoud van fundamentele programmeervaardigheden, het kritisch omgaan met generatieve AI, het behoud van motivatie en betrouwbare evaluatie.

Verschillende respondenten geven aan dat lerenden geneigd zijn gegeneerde code zonder diepgaand inzicht over te nemen. Volgens hen doen lerenden vaak te weinig moeite om zelf oplossingen uit te werken en grijpen zij al vanaf de eerste stap naar generatieve AI, waardoor het volledige denkproces wordt overgeslagen. Daardoor worden vaardigheden zoals zelfstandig nadenken, logisch redeneren, debuggen en het opbouwen van oplossingen minder geoefend.

Meerdere lesgevers merken bovendien op dat lerenden vaak onvoldoende begrijpen wat generatieve AI precies genereert, mede omdat de code soms instructies of technieken bevat die nog niet aan bod kwamen in de leerstof. Lerenden hebben daardoor moeite om code kritisch te beoordelen of aan te passen, verliezen houvast en haken soms af. Daarom wordt herhaaldelijk gewezen op de noodzaak om lerenden op een correcte en kritische manier met generatieve AI te leren omgaan. Een respondent beschouwt het efficiënt gebruiken van generatieve AI zelfs als een toekomstige basiscompetentie.

Door de brede beschikbaarheid van generatieve AI zien meerdere lesgevers het ook als een belangrijke uitdaging om lerenden gemotiveerd te houden om zelf programmeervaardigheden te blijven ontwikkelen. Volgens

een respondent bestaat anders het risico dat lerenden slagen in hun opleiding zonder voldoende inzicht in programmeren te verwerven.

Verder ervaren verschillende respondenten moeilijkheden met evaluatie en opvolging, vooral buiten de lescontext en in het afstandsonderwijs, waar het vaak moeilijk blijft in te schatten in welke mate werk daadwerkelijk zelfstandig gerealiseerd werd.

Ten slotte geven enkele respondenten aan weinig of geen uitdagingen met het gebruik van generatieve AI te ervaren.

Authenticiteit van ingediende oplossingen

Respondenten tonen zich overwegend bezorgd over de authenticiteit van het werk van lerenden in een context waarin mogelijk generatieve AI gebruikt wordt: 43% geeft aan eerder bezorgd te zijn en 16% zeer bezorgd. Daarnaast blijft 24% neutraal, terwijl 16% zich eerder niet bezorgd toont. Niemand geeft aan helemaal onbezorgd te zijn.

De toelichtingen benaderen deze problematiek vanuit verschillende invalshoeken: detectie, preventie en integratie van generatieve AI.

Vier lesgevers hebben het expliciet over detectie van gebruik van generatieve AI. Drie van hen geven aan dat ongeoorloofd gebruik vaak herkenbaar is, bijvoorbeeld via schriftelijke toetsen, demonstraties of kenmerken van de ingediende code. De vierde lesgever merkt daarentegen op dat het niet altijd duidelijk is of lerenden van generatieve AI gebruik maken.

Ook preventieve maatregelen komen aan bod. Zo vermeldt een respondent dat tijdens evaluaties getracht wordt achterpoortjes voor misbruik zoveel mogelijk af te sluiten. Iemand anders geeft aan dat programmeeropdrachten die thuis worden gemaakt niet meetellen voor evaluaties.

Ten slotte beschouwen twee lesgevers AI als een technologie die geïntegreerd moet worden binnen het leerproces. Volgens hen moeten lerenden deze hulpmiddelen geschikt kunnen aanwenden om kwaliteitsvolle output te verkrijgen. Generatieve AI wordt daarbij beschouwd als een inherent onderdeel van het ontwikkelen van programma's, vergelijkbaar met het gebruik van spellingscontrole in een tekstverwerker. Daarnaast merkt een respondent op dat lerenden beseffen dat generatieve AI ondersteunend kan zijn, maar niet bedoeld is om oefeningen volledig te laten oplossen.

Invloed op programmeervaardigheden

Ongeveer twee derde van de respondenten beschouwt de invloed van generatieve AI op de programmeervaardigheden van lerenden als enigszins negatief (43%) of zeer negatief (22%). Verder geeft 14% aan geen duidelijk effect waar te nemen, terwijl 22% de invloed als enigszins positief beoordeelt. Geen enkele respondent beschouwt de impact als zeer positief.

Uit de toelichtingen blijkt dat de negatieve perceptie voornamelijk verband houdt met het uitbesteden van denkwerk aan generatieve AI. Lerenden nemen volgens respondenten geregeld gegenereerde code over zonder voldoende inzicht, waardoor basisvaardigheden zoals algoritmisch denken, zelfstandig probleemoplossend leren, het kritisch evalueren van oplossingen (testen en debuggen) en het analyseren van opdrachten minder geoefend worden. Ook wordt meermaals opgemerkt dat lerenden sneller naar generatieve AI grijpen uit onzekerheid of wanneer zij vastlopen, waardoor zelfstandig oefenen en het doorlopen van de leercurve onder druk komen te staan.

Verschillende respondenten benadrukken bovendien dat lerenden de gegenereerde code vaak onvoldoende begrijpen en minder kritisch omgaan met de correctheid of geschiktheid ervan. Een respondent wijst daarnaast op een mogelijke afname van creativiteit, inspanning en retentie van leerstof. Ook wordt vermeld dat lerenden die sterk afhankelijk worden van generatieve AI tijdens evaluaties soms onvoldoende inzicht blijken te hebben in programmacode of in het vertalen van opdrachten naar concrete oplossingen.

Tegelijk noemt een minderheid van de respondenten ook enkele positieve aspecten. Volgens sommigen kan generatieve AI ondersteunend werken wanneer lerenden de basisvaardigheden reeds beheersen, bijvoorbeeld bij het controleren van oplossingen of het sneller ontwikkelen van uitgebreidere software. Daarbij wordt wel benadrukt dat generatieve AI een hulpmiddel moet blijven en het eigen denkproces niet mag vervangen.

Ten slotte legt een respondent een verband tussen de impact van generatieve AI en de motivatie van de lerende. Volgens deze lesgever kunnen intrinsiek gemotiveerde lerenden generatieve AI zinvol inzetten ter ondersteuning van oefeningen, terwijl ongemotiveerde lerenden het risico lopen het programmeerproces grotendeels uit te besteden.

Andere bezorgdheden

Naast de eerder besproken thema's formuleren respondenten ook enkele bredere bezorgdheden over het gebruik van generatieve AI binnen het programmeeronderwijs.

Wat begeleiding en ondersteuning betreft, merkt een respondent op dat het gebruik van generatieve AI zo goed mogelijk begeleid wordt binnen de lespraktijk. Tegelijk wordt aangehaald dat lesgevers onvoldoende worden opgeleid om snel evoluerende technologieën zoals generatieve AI aan te leren, en dat in heel wat scholen een degelijk beleid hieromtrent ontbreekt.

Een respondent formuleert daarnaast een uitgesproken kritische houding ten opzichte van generatieve AI door te stellen dat deze technologie “*een afstompend effect op alles en iedereen*” heeft.

Ten slotte komt volgens twee lesgevers de relevantie van programmeervaardigheden onder druk te staan, omdat generatieve AI programmeertaken ogenschijnlijk kan overnemen. In het verlengde daarvan vreest een lesgever dat programmeeronderwijs op termijn zou kunnen verdwijnen.

3.6 Implicaties voor het programmeeronderwijs

3.6.1 Vaardigheden van lerenden die belangrijker worden

Probleemanalyse wordt door 89% van de respondenten aangeduid als vaardigheid die belangrijker wordt door de beschikbaarheid van generatieve AI, bijvoorbeeld het begrijpen en ontleden van programmeervraagstukken vóór het programmeren. Eveneens 89% benadrukt het belang van het begrijpen en beoordelen van code, terwijl 73% debugging noemt als vaardigheid die extra aandacht vraagt.

Daarnaast worden het documenteren van oplossingen en het kunnen redeneren hierover door 62% van de lesgevers als belangrijk beschouwd. Verder duidt 59% prompttraining aan als competentie die meer aandacht verdient. In de open antwoorden wordt ook gewezen op het belang van het beheren van code in plaats van louter code te schrijven.

Niemand geeft aan dat bijkomende aandacht voor bepaalde vaardigheden niet nodig is.

3.6.2 Aanpassingen in de lescontext

Een meerderheid van de respondenten geeft aan de lespraktijk minstens in zekere mate aangepast te hebben om rekening te houden met generatieve AI: 24% stelt dit altijd te doen en 35% soms. Daarnaast overweegt 24% nog aanpassingen door te voeren, terwijl 16% dit niet overweegt.

Uit de toelichtingen komt een duidelijk spanningsveld naar voren tussen integratie en afscherming. Een eerste groep lesgevers kiest ervoor generatieve AI bewust op te nemen in de lespraktijk. Zij integreren deze technologie in oefeningen, projecten of zelfs de volledige cursusinhoud, maar koppelen dit doorgaans aan voorwaarden: lerenden mogen gegenereerde code gebruiken, maar moeten deze vervolgens kunnen uitleggen, analyseren, verbeteren of kritisch beoordelen. Ook worden soms gerichte prompts aangereikt, zodat lerenden ondersteuning krijgen zonder onmiddellijk uitgewerkte oplossingen te ontvangen.

Een tweede groep aanpassingen vertrekt eerder vanuit controle en afscherming. Daarbij wordt geprobeerd om het inzicht en de redenering van lerenden beter zichtbaar en toetsbaar te maken. Zo worden schermen nauwer opgevolgd, worden vragen zo geformuleerd dat het gebruik van generatieve AI herkenbaar wordt, en verlopen toetsen of examens vaker op papier. Ook worden voorbereidingen buiten de klas soms geschrapt of niet langer geëvalueerd.

Tussen beide benaderingen bestaan ook tussenvormen. Verschillende lesgevers combineren het toelaten of integreren van generatieve AI met aangepaste evaluatievormen, zoals demonstraties, mondelinge verdedigingen en presentaties. Daarbij moeten lerenden hun code regel per regel toelichten, of gerichte vragen over hun aanpak beantwoorden. De nadruk verschuift zo van het louter produceren van code naar het kunnen begrijpen en verantwoorden ervan.

Tot slot blijkt dat generatieve AI niet alleen de activiteiten van lerenden beïnvloedt, maar ook de voorbereiding door lesgevers. Zo wordt deze technologie gebruikt voor het genereren van presentaties, het ontwikkelen van cursusmateriaal of het mogelijk maken van meer hedendaagse softwaretoepassingen. De aanpassingen raken daardoor zowel aan inhoud, begeleiding, evaluatie als materiaalontwikkeling.

3.6.3 Aanpassingen aan programmeeropdrachten

Lesgevers werden gevraagd in welke mate zij programmeeropdrachten aanpassen in functie van generatieve AI. De antwoorden tonen een verdeeld beeld: 11% van de respondenten past opdrachten altijd aan, terwijl 35% dit soms doet. Daarnaast overweegt eveneens 35% nog aanpassingen door te voeren, tegenover 19% die dit niet overweegt. Respondenten lijken terughoudender om programmeeropdrachten aan te passen dan om wijzigingen door te voeren in de lespraktijk: meer lesgevers overwegen nog aanpassingen, terwijl minder lesgevers opdrachten systematisch aanpassen.

Uit de toelichtingen blijkt dat veel aanpassingen gericht zijn op een verschuiving van focus. Waar vroeger vooral het functioneren van code centraal stond, wordt nu vaker nadruk gelegd op analyse, begrip, optimalisatie en reflectie. Lerenden mogen in sommige gevallen gegenereerde code gebruiken, maar moeten deze vervolgens kunnen uitleggen, analyseren, verbeteren of verder uitwerken. Ook worden vaker reflectievragen, mondelinge toelichtingen of stapsgewijze besprekingen van code ingezet om na te gaan of lerenden hun oplossing daadwerkelijk begrijpen.

Die nadruk op begrip komt ook terug in de strategieën die respondenten het meest effectief achten. Mondelinge toelichtingen of stapsgewijze besprekingen van code worden het vaakst genoemd (81%), gevolgd door reflectievragen (65%). Daarnaast legt 43% meer nadruk op het werkproces, bijvoorbeeld via versiegeschiedenis of tussentijdse opvolging. Het documenteren van AI-gebruik wordt door 32% aangeduid als mogelijke strategie. Dit suggereert dat sommige lesgevers minder inzetten op het volledig verbieden van generatieve AI, en eerder streven naar transparantie over hoe dergelijke hulpmiddelen gebruikt worden.

Programmeeropdrachten worden ook inhoudelijk aangepast. Ze bevatten soms expliciete leerdoelen rond het gebruik van generatieve AI, bijvoorbeeld als ondersteuning binnen het programmeerproces of voor het genereren van voorbeelddata. Daarnaast worden opdrachten soms omvangrijker of complexer gemaakt, onder meer via grotere projecten of uitdagendere programmeerproblemen. Hierdoor wordt het eenvoudig overnemen van gegenereerde oplossingen minder evident. Binnen dergelijke opdrachten ligt meer nadruk op analyse, structuur en het kunnen toelichten van gemaakte keuzes.

Daarnaast besteden sommige lesgevers meer aandacht aan het werkproces,

bijvoorbeeld via versiegeschiedenis, tussentijdse opvolging, het documenteren van AI-gebruik of het zelfstandig laten uitwerken van oplossingen, bijvoorbeeld via pseudocode op papier.

Andere strategieën hebben betrekking op de vormgeving en uitvoering van opdrachten. Zo duidt 27% samenwerkend leren als mogelijke strategie aan, terwijl een even grote groep kiest voor meer open of praktijkgerichte taken. In de open antwoorden worden daarnaast ook opdrachten op papier, het gebruik van educatieve omgevingen met afgesloten internettoegang en gepersonaliseerde datasets of opdrachtspecifieke coderichtlijnen vermeld. Enkele respondenten kiezen er bovendien voor programmeeropdrachten uitsluitend in de klas te laten uitvoeren, mogelijk om meer zicht te behouden op het werkproces van lerenden en om het gebruik van generatieve AI beter te kunnen opvolgen. Een beperkte minderheid ziet geen noodzaak (8%) of geen mogelijkheden (3%) om opdrachten aan te passen aan generatieve AI.

3.6.4 Aanpassingen aan evaluatie

Bij de evaluatie van programmeeropdrachten waarbij generatieve AI gebruikt wordt, vraagt 54% van de respondenten een mondelinge toelichting of een stapsgewijze bespreking van de code om na te gaan of lerenden hun oplossing daadwerkelijk begrijpen. Dit wijst erop dat de beoordeling niet uitsluitend gericht is op het eindproduct, maar ook op de mate waarin lerenden hun code kunnen uitleggen en onderbouwen.

Daarnaast past 30% de beoordeling aan wanneer AI-hulpmiddelen gebruikt worden, bijvoorbeeld via strengere criteria, bijkomende controle of een grotere nadruk op begrip.

Een kleinere groep lesgevers (16%) beoordeelt werk waarbij dergelijke hulpmiddelen gebruikt worden op dezelfde manier als ander werk. Dit suggereert dat sommige lesgevers ervan uitgaan dat dezelfde beoordelingscriteria volstaan, zolang het ingediende werk aan de vooropgestelde verwachtingen voldoet.

Een deel van de respondenten interpreteert evaluatie eerder in termen van toelating dan van beoordeling. Zo geven sommigen aan dat generatieve AI niet toegelaten is tijdens evaluaties, of dat enkel werk beoordeeld wordt dat zonder dergelijke hulpmiddelen tot stand kwam.

3.6.5 Veranderende rol van lesgevers

Volgens een minderheid van de respondenten leidt generatieve AI tot beperkte of geen veranderingen in de rol van programmeerlesgevers. Daarbij wordt onder meer verwezen naar de blijvende noodzaak om technieken en concepten aan te leren. Een respondent maakt de vergelijking met de rekenmachine, die de rol van de wiskundeleraar evenmin fundamenteel veranderde. In deze visie blijft de lesgever expert en fungeert generatieve AI vooral als aanvullend hulpmiddel.

Enkele respondenten drukken onzekerheid uit over de toekomstige impact van generatieve AI. De verdere evolutie is moeilijk in te schatten, onder meer omdat het om een relatief recent fenomeen gaat. Tegelijk stelt één respondent zonder verdere toelichting dat generatieve AI de rol van de programmeerleerkracht in sterk negatieve zin verandert.

Bij respondenten die een duidelijke verandering verwachten, komt vooral een verschuiving naar voren van kennisoverdrager naar begeleider. Één respondent merkt daarbij op dat de rol van de lesgever niet kleiner wordt, maar inhoudelijk rijker en didactisch veelzijdiger. Meer algemeen wordt een verschuiving verwacht van het stapsgewijs uitleggen van code naar het begeleiden van het leer- en denkproces van lerenden. Daarbij wordt het belangrijker om lerenden kritisch te leren omgaan met generatieve AI, hen inzicht te laten verwerven in de werking van gegenereerde code, en te vermijden dat zij oplossingen louter overnemen. Verder worden projectmatig werken, foutopsporing, begeleiding van het realisatieproces en het aanleren van nieuwe technologieën genoemd. Over de mate waarin deze rolverandering ingrijpend zal zijn, bestaat echter geen eensgezindheid: sommige respondenten spreken over een fundamentele verandering, terwijl anderen eerder een geleidelijke evolutie beschrijven.

Volgens meerdere respondenten moeten lesgevers zowel het denkproces van lerenden als hun gebruik van generatieve AI nauwer opvolgen en sterker sturen. Daarnaast zouden meer individuele leergesprekken, sterkere differentiatie en aangepaste evaluatievormen noodzakelijk worden. Sommige lesgevers ervaren hierdoor een verschuiving naar een meer controlerende rol.

Ook op inhoudelijk vlak wordt een verschuiving verwacht. De focus zou minder liggen op syntax en het uitvoerig schrijven van code, en meer op computationeel denken, ideeënvorming en het herkennen en toepassen van

grotere structuren en technieken. Tot slot worden ook softwarearchitectuur, security en het formuleren van prompts voor generatieve AI genoemd als thema's die aan belang kunnen winnen binnen het programmeeronderwijs.

3.6.6 Professionalisering van lesgevers

Van de respondenten geeft 43% aan zich voldoende voorbereid te voelen om generatieve AI zinvol te integreren in het programmeeronderwijs. Tegelijk rapporteert 35% nood aan training, wenst 27% richtlijnen voor een didactisch verantwoorde inzet van dergelijke hulpmiddelen en geeft 11% aan behoefte te hebben aan meer ondersteuning vanuit de school of instelling.

In de toelichtingen verwijzen respondenten die zich voldoende voorbereid voelen onder meer naar eigen initiatief, zelfredzaamheid, ondersteuning binnen de schoolcontext en het volgen van een opleiding. Verder wordt aangegeven dat bijkomende vorming en kennisdeling, ondanks een zekere basisvertrouwdheid, nog steeds een meerwaarde kunnen bieden.

Een respondent die aangeeft nood te hebben aan training verwijst naar het snelle tempo waarmee de technologie evolueert. Een andere lesgever geeft aan dat expertise momenteel vooral gebaseerd is op eigen initiatief en dat meer structurele ondersteuning wenselijk is. Aanpakstrategieën voor het programmeeronderwijs met generatieve AI zouden meer aandacht mogen krijgen binnen opleidingen en nascholingen.

4. Discussie

4.1 Reflecties van de auteur

Doorheen deze masterpraktijkproef streef ik naar een zo objectief mogelijke bespreking van de literatuur en onderzoeksbevindingen. In deze paragraaf formuleer ik echter enkele persoonlijke reflecties bij de bevindingen van dit onderzoek. Om die reden hanteer ik hier uitzonderlijk de ik-vorm.

Generatieve AI in de lescontext

In de lescontext zie ik een spanningsveld tussen een integrerende en een beschermende houding ten aanzien van generatieve AI. Welke aanpak het meest aangewezen is, lijkt sterk afhankelijk van de context, waaronder het aantal beschikbare lessen, de didactische accenten van de lesgever en de kenmerken van de leerstof en klasgroep.

Toch zou ik in omvangrijkere programmeervakken generatieve AI minstens gedeeltelijk integreren, gezien het toenemende belang ervan in het werkveld en de mogelijkheden die deze technologie biedt voor het leerproces. Hoewel dit het meest vanzelfsprekend lijkt bij gevorderde programmeurs, is een doordachte integratie volgens mij ook bij beginners mogelijk.

Zo lijkt het me waardevol om lerenden al vroeg te begeleiden in het doelgericht en kritisch gebruik van generatieve AI. Daarbij moet de nadruk volgens mij liggen op de ondersteuning van het leerproces. Generatieve AI kan hierbij functioneren als een vorm van gepersonaliseerde begeleiding, bijvoorbeeld door gerichte hints te geven, alternatieve uitleg aan te bieden of feedback te verstrekken op tussenstappen. Op die manier blijft de actieve betrokkenheid van de lerende centraal staan. Daarnaast lijkt het belangrijk om niet alleen aandacht te besteden aan effectieve prompting, maar ook aan het kritisch evalueren van de gegenereerde output.

Tegelijk lijkt het mij aangewezen om erop te waken dat lerenden fundamentele programmeervaardigheden zelfstandig beheersen, zonder daarbij terug te vallen op generatieve AI.

Generatieve AI buiten de lescontext

Het gebruik van generatieve AI buiten de lescontext valt nauwelijks volledig te controleren. Om die reden ben ik geen voorstander van een algemeen verbod op het gebruik ervan bij programmeeropdrachten. Persoonlijk zou ik opdrachten buiten de lescontext enkel evalueren wanneer dit nodig is om de betrokkenheid van lerenden te verhogen. Wanneer er toch wordt geëvalueerd, lijkt het mij belangrijk om niet zozeer het eindresultaat te beoordelen, maar vooral de manier waarop lerenden hun oplossing toelichten en verantwoorden.

Meerdere respondenten gaven aan geen programmeertaken meer buiten de lescontext te voorzien. Ik beschouw dit niet als een wenselijke evolutie. Programmeervaardigheden ontwikkelen zich immers grotendeels door oefening, terwijl de beschikbare lestijd daarvoor niet volstaat.

Bovendien lieten meerdere ervaren programmeerleerkrachten mij, los van dit onderzoek, weten dat leerlingen buiten de klas minder inspanningen lijken te leveren dan vroeger en minder intrinsieke motivatie vertonen. Ik zie mogelijkheden voor generatieve AI om de betrokkenheid van lerenden bij programmeren buiten de klas te versterken. Mits een doordachte inzet kan deze technologie bijvoorbeeld lerenden ondersteunen bij persoonlijke projecten waarin ruimte is voor eigen keuzes en interesses.

Preventie en detectie van fraude

Versillende respondenten geven aan het gebruik van generatieve AI relatief gemakkelijk te kunnen herkennen. Welke kenmerken daarbij precies als aanwijzing worden beschouwd, werd niet verder toegelicht. Mogelijke signalen zijn het gebruik van technieken die nog niet in de les aan bod kwamen of code die aanzienlijk afwijkt van wat de lerende doorgaans produceert. Dergelijke aanwijzingen bieden echter geen zekerheid. Ik sluit niet uit dat een deel van de leerlingenpopulatie generatieve AI op een subtielere manier inzet, bijvoorbeeld door gegenereerde oplossingen aan te passen aan het niveau van de leerstof, waardoor het gebruik minder zichtbaar blijft. Deze onzekerheid sluit aan bij bevindingen uit de literatuur, die erop wijzen dat het gebruik

van generatieve AI moeilijk betrouwbaar te detecteren is.

Om die reden beschouw ik preventie als een kansrijkere strategie dan detectie. Binnen de lescontext kunnen verschillende preventieve maatregelen worden toegepast, zoals schermmonitoring, internetbeperkingen of evaluaties op papier. De resultaten van de vragenlijst tonen bovendien een samenhang tussen dergelijke maatregelen en een lagere inschatting van de mate waarin lerenden generatieve AI gebruiken. Hoewel hieruit geen oorzakelijk verband kan worden afgeleid, zie ik hierin wel een aanwijzing dat preventieve maatregelen het gebruik van generatieve AI binnen de klas kunnen beperken.

De mogelijkheden van preventie blijven echter grotendeels beperkt tot de lescontext. Zoals eerder aangehaald beschouw ik controle buiten de lescontext als vrijwel onmogelijk.

De toekomst van generatieve AI

Tijdens deze masterpraktijkproef viel mij op hoe sterk het onderwerp van generatieve AI leeft, zowel in de academische wereld als in de onderwijspraktijk. De wetenschappelijke literatuur hierover is reeds zeer omvangrijk en breidt zich nog steeds snel uit. Ook in de media wordt regelmatig bericht over nieuwe ontwikkelingen.

Daarnaast kwam generatieve AI tijdens mijn stage meermaals spontaan ter sprake, zowel bij lesgevers als bij lerenden. Ook de vragenlijst kende een relatief hoge responsgraad, waarbij veel respondenten uitgebreide en doordachte antwoorden formuleerden. Bovendien gaf een opmerkelijk groot aandeel van de respondenten vrijwillig hun e-mailadres op om de resultaten van dit onderzoek te ontvangen. Dit alles wijst erop dat generatieve AI door veel betrokkenen niet wordt beschouwd als een tijdelijke trend, maar als een ontwikkeling met een belangrijke impact op het onderwijs en het werkveld.

Tegelijk maakt de snelheid waarmee deze technologie evolueert het moeilijk om verregaande uitspraken te doen over de toekomst. Verschillende voorspellingen die vandaag plausibel lijken, kunnen binnenkort al achterhaald zijn. Daarom beschouw ik deze masterpraktijkproef vooral als een momentopname binnen een snel veranderend domein. Ik verwacht dan ook dat verder onderzoek naar generatieve AI in het programmeeronderwijs de komende jaren noodzakelijk zal blijven.

4.2 Synthese van de literatuur

Deze paragraaf biedt een synthese van de bevindingen uit de literatuur over het gebruik van generatieve AI in het programmeeronderwijs, met aandacht voor zowel didactische kansen als uitdagingen. Raadpleeg hoofdstuk 2 voor de volledige literatuurstudie.

Kansen

Generatieve AI kan lerenden ondersteunen door op te treden als *pair programmer*, persoonlijke tutor of hulpmiddel bij programmeertaken, waardoor zij zelfstandiger kunnen werken, verschillende oplossingsstrategieën verkennen en ondersteuning krijgen bij het begrijpen en aanpassen van bestaande code. Dit kan bijdragen aan betere programmeerprestaties, minder programmeerangst en een dieper inzicht in programmeerconcepten. Voor lesgevers biedt generatieve AI mogelijkheden bij het ontwikkelen en differentiëren van didactisch materiaal en bij taken zoals literatuuronderzoek en tekstverwerking.

Ook evaluatie en feedback in het programmeeronderwijs kunnen ondersteund worden door generatieve AI. In tegenstelling tot klassieke testgebaseerde evaluatie kan deze technologie niet alleen de correctheid van code beoordelen, maar ook feedback geven over aspecten zoals codekwaliteit, leesbaarheid, efficiëntie en het naleven van programmeerstandaarden. Daarnaast kan generatieve AI lesgevers ondersteunen door individuele beoordelingen te aggregeren tot een overzicht van de prestaties van een volledige groep, inclusief veelvoorkomende fouten, misconcepties en aandachtspunten. Tegelijk wijzen verschillende studies op beperkingen, waaronder hallucinaties en variatie tussen beoordelingen, waardoor menselijke controle noodzakelijk blijft.

Generatieve AI creëert ook mogelijkheden voor nieuwe didactische benaderingen in het programmeeronderwijs. Het aanwenden ervan voor implementatietaken maakt het mogelijk de nadruk te verschuiven van het louter schrijven van code naar analyse, evaluatie, verbetering en ontwerp. Daardoor kunnen lerenden meer aandacht besteden aan hogere cognitieve vaardigheden en kennismaken met complexere en realistischere programmeercontexten, op voorwaarde dat zij AI-gegenereerde output kritisch beoordelen.

Verder blijkt uit de literatuur dat inzicht in het AI-gebruik van lerenden kansen biedt voor differentiatie. Lerenden gebruiken generatieve AI immers

op uiteenlopende manieren, waarbij verschillen in gebruik onder meer samenhangen met voorkennis van programmeren en de wijze waarop zij met AI interageren. Door dergelijke gebruikspatronen te herkennen, kunnen lesgevers hun begeleiding mogelijk beter afstemmen op individuele leerbehoeften.

Generatieve AI kan bovendien bijdragen aan de intrinsieke motivatie van lerenden door hun autonomie, competentiegevoel en zelfeffectiviteit te versterken en negatieve gevoelens zoals frustratie en verveling te verminderen.

Verschillende studies suggereren dat prompttraining kan bijdragen aan een effectiever gebruik van generatieve AI en mogelijk ook samenhangt met hogere betrokkenheid en betere leerresultaten. Door lerenden aan te leren hoe zij doelgerichte en goed gestructureerde prompts formuleren, kunnen zij generatieve AI doelgerichter benutten en beter presteren bij data-analyse- en programmeertaken.

Uitdagingen

De voornaamste uitdaging bij het gebruik van generatieve AI in het programmeeronderwijs betreft academische integriteit. Daaronder vallen onder meer plagiaat, zelfplagiat, ongeoorloofde samenwerking, uitbesteding, vervalsing en het gebruik van niet-toegestane hulpmiddelen. Deze problematiek wordt versterkt door verschillende kenmerken van het programmeeronderwijs. Zo zijn samenwerking en hergebruik van code gangbare praktijken in professionele softwareontwikkeling. Daarnaast laten programmeeropdrachten vaak slechts een beperkt aantal correcte oplossingen toe, wat de detectie van plagiaat bemoeilijkt. Bovendien wordt generatieve AI steeds beter in het genereren van hoogwaardige code. Om die redenen beschouwen verschillende onderzoekers aanpassingen aan het onderwijsmodel als noodzakelijk en urgent.

De literatuur wijst op het belang van bewustmaking rond generatieve AI en academische integriteit, duidelijke regels en transparante documentatie van AI-gebruik. Omdat generatieve AI het moeilijker maakt om de individuele bijdrage en het begrip van lerenden te beoordelen, dient de focus volgens verschillende auteurs te verschuiven van het louter produceren van code naar begrip, werkwijze en kritische omgang met gegenereerde output. Mogelijke antwoorden hierop zijn meer gecontroleerde evaluatievormen, zoals evaluatie op papier, mondelinge examens en opdrachten binnen de lescontext. Sommige auteurs pleiten daarnaast voor een expliciete integratie van generatieve

AI, waarbij lerenden het gebruik ervan moeten kunnen verantwoorden en de gegenereerde code moeten begrijpen. Een te restrictieve aanpak kan daarentegen een zinvolle inzet van generatieve AI belemmeren.

Louter technische detectie biedt geen sluitende oplossing om fraude tegen te gaan. AI-detectietools kunnen verdachte patronen signaleren, maar blijven gevoelig voor fouten, taalverschillen, gedeeltelijk AI-gebruik, samenwerking en aanpassingen aan gegenereerde output. De uiteindelijke beoordeling blijft daardoor afhankelijk van menselijke interpretatie en didactische afwegingen.

Een tweede uitdaging betreft bias en de kwaliteit en betrouwbaarheid van AI-gegenereerde code. Omdat trainingsdata vaak afkomstig zijn van ervaren ontwikkelaars, kunnen gegenereerde oplossingen complexer zijn dan didactisch wenselijk en minder goed aansluiten bij de beoogde leerdoelen. Daarnaast kunnen codegeneratiemodellen oplossingen voorstellen die ogenschijnlijk correct zijn, maar niet voldoen aan de beoogde functionaliteit of aan kwaliteitscriteria zoals robuustheid, onderhoudbaarheid, begrijpelijkheid en efficiëntie. Ook beveiliging vormt een aandachtspunt, aangezien AI-gegenereerde code kwetsbaarheden kan bevatten. Daarom mag de evaluatie van gegenereerde code niet beperkt blijven tot functionele correctheid: menselijke controle en kritische beoordeling blijven essentieel.

Een derde belangrijke uitdaging is afhankelijkheid. Wanneer lerenden generatieve AI gebruiken om programmeeropdrachten op te lossen, bestaat het risico dat zij minder diepgaande kennis en probleemoplossingsvaardigheden ontwikkelen dan bij zelfstandig probleemoplossen. Deze afhankelijkheid kan worden versterkt door *cognitive offloading* (cognitieve uitbesteding), waarbij denk- en redeneerprocessen aan generatieve AI worden uitbesteed. Daardoor kunnen betere prestaties op korte termijn optreden zonder overeenkomstige leerwinst. Bovendien kan AI-gegenereerde uitleg ook bijdragen aan een vals gevoel van begrip.

Een evenwichtige inzet van generatieve AI blijft daarom noodzakelijk. De technologie lijkt vooral waardevol wanneer zij het leerproces ondersteunt, maar niet wanneer zij de cognitieve activiteiten vervangt die nodig zijn om programmeervaardigheden en conceptueel begrip op te bouwen.

4.3 Synthese van de vragenlijst

Deze paragraaf biedt een synthese van de bevindingen uit het empirisch onderzoek naar het gebruik van generatieve AI binnen het Vlaamse programmeeronderwijs en naar de percepties en ervaringen van lesgevers. Raadpleeg hoofdstuk 3 voor de volledige analyse.

Gebruik door lerenden en didactische visies

Het gebruik van generatieve AI door lerenden vertoont een grote spreiding. Tijdens programmeerlessen hangt het gebruik sterk samen met de mate waarin lesgevers generatieve AI verbieden, beperken of expliciet integreren. Buiten de lescontext lijkt het gebruik sterker ingeburgerd en moeilijker controleerbaar, waardoor sommige lesgevers minder of geen programmeeropdrachten meer buiten de lescontext voorzien.

De didactische visie van lesgevers varieert eveneens. Sommige respondenten benadrukken het belang van een stevige programmeerbasis voordat generatieve AI wordt ingezet, terwijl anderen de technologie beschouwen als een inherent onderdeel van het programmeren. Over de uiteenlopende visies heen bestaat wel brede overeenstemming dat generatieve AI het belang doet toenemen van vaardigheden zoals probleemanalyse, het begrijpen en beoordelen van code, debugging en het kritisch omgaan met gegenereerde output.

Kansen

Lesgevers zien generatieve AI vooral als een ondersteunend leermiddel, op voorwaarde dat het gebruik ervan voldoende begeleid wordt. De technologie kan lerenden helpen bij het verkrijgen van uitleg, het begrijpen van foutmeldingen, het raadplegen van voorbeelden, het controleren of verfijnen van oplossingen en het verkennen van nieuwe programmeerconcepten. Daarnaast kan generatieve AI bijdragen aan snellere codeontwikkeling en aan een verschuiving van louter code schrijven naar het begrijpen, evalueren, debuggen en aanpassen van code.

Ook voor lesgevers biedt generatieve AI kansen. De technologie wordt onder meer ingezet voor het ontwikkelen van toets- en opdrachtmateriaal, het genereren van inspiratie of voorbeeldcode en de voorbereiding van lesma-

teriaal. Daarnaast kan generatieve AI bijdragen aan grotere en complexere projecten, aan opdrachten die beter aansluiten bij de leefwereld van lerenden en aan een verhoogde motivatie door sneller zichtbare resultaten.

Uitdagingen

De respondenten wijzen tegelijk op belangrijke risico's. Een terugkerende bezorgdheid is dat lerenden gegenereerde code overnemen zonder voldoende inzicht, waardoor zelfstandig redeneren, debugging en probleemoplossende vaardigheden minder worden geoefend. Daarnaast vrezen verschillende respondenten dat de brede beschikbaarheid van generatieve AI de motivatie van lerenden kan verminderen om zelfstandig programmeervaardigheden te ontwikkelen.

Daarnaast ervaren lesgevers moeilijkheden met evaluatie, opvolging en de authenticiteit van ingediende oplossingen, vooral buiten de lescontext. Respondenten benaderen deze problematiek uiteenlopend: sommigen zetten in op detectie of beperking van misbruik, terwijl anderen generatieve AI juist willen integreren en lerenden willen leren om deze hulpmiddelen doordacht te gebruiken. Verder wijzen respondenten op het ontbreken van een duidelijk beleid en op de nood aan verdere professionalisering.

Aanpassingen in de lespraktijk

Generatieve AI wordt door veel lesgevers minstens occasioneel gebruikt, vooral bij het ontwikkelen van toets- en opdrachtmateriaal en het genereren van voorbeeldcode. Daarnaast wordt de technologie in mindere mate ingezet voor het voorbereiden van lesmateriaal, terwijl het gebruik voor feedbackdoeleinden beperkter blijft.

Een meerderheid van de lesgevers past de lespraktijk in zekere mate aan. Deze aanpassingen situeren zich op een spectrum van integratie tot afscherming. Bij integratie wordt generatieve AI bewust opgenomen in oefeningen, projecten of cursussen, waarbij lerenden doorgaans moeten aantonen dat zij gegenereerde code kunnen uitleggen, analyseren of verbeteren. Bij een afschermende aanpak kiezen lesgevers eerder voor sterker toezicht, papieren toetsen, beperkingen op internetgebruik of het verminderen van opdrachten buiten de lescontext.

Aanpassingen aan programmeeropdrachten en evaluatie

Bijna de helft van de respondenten past programmeeropdrachten aan in functie van generatieve AI. Daarbij verschuift de focus gedeeltelijk van het louter laten functioneren van code naar analyse, begrip, optimalisatie en reflectie. Sommige opdrachten worden uitgebreid met expliciete leerdoelen rond AI-gebruik, tussentijdse opvolging, documentatie van het werkproces of meer open en praktijkgerichte taken.

Ook evaluatievormen worden aangepast. Een kleine meerderheid vraagt lerenden om hun code mondeling of stapsgewijs toe te lichten, terwijl ongeveer een derde de beoordeling aanpast via strengere criteria of bijkomende controle. Deze aanpassingen zijn vooral gericht op het nagaan of lerenden hun oplossing daadwerkelijk begrijpen.

Veranderende rol en professionalisering van lesgevers

Respondenten verschillen in hun inschatting van de impact van generatieve AI op de rol van programmeerlesgevers. Sommigen beschouwen de technologie vooral als een aanvullend hulpmiddel, terwijl anderen een verschuiving zien van kennisoverdracht naar begeleiding van het leer- en denkproces. Daarbij komen computationeel denken, kritische omgang met gegenereerde code, foutopsporing, projectmatig werken en het opvolgen van AI-gebruik sterker op de voorgrond.

Deze veranderende verwachtingen stellen ook nieuwe eisen aan de professionalisering van lesgevers. Hoewel een deel van de respondenten zich voldoende voorbereid acht, geven velen aan nood te hebben aan bijkomende training, didactische richtlijnen en ondersteuning. De aanwezige expertise lijkt daardoor vaak voort te komen uit eigen initiatief en zelfredzaamheid.

Samenvattend vertonen de resultaten uit de vragenlijst in grote lijnen overeenkomsten met de bevindingen uit de literatuurstudie. Zowel de literatuur als de respondenten wijzen op een verschuiving van de nadruk op codeproductie naar vaardigheden zoals analyse, evaluatie en debugging. Daarnaast worden zowel kansen als uitdagingen van generatieve AI benadrukt. Kansen situeren zich onder meer op het vlak van ondersteuning bij programmeren, feedback en efficiëntie, terwijl bezorgdheden vooral verband houden met fraude, afhankelijkheid, evaluatie en betrouwbaarheid. Deze overeenkomsten

suggesteren dat de ervaringen van Vlaamse programmeerleerkrachten in grote lijnen aansluiten bij de inzichten uit de literatuur.

4.4 Beperkingen van het onderzoek

Dit onderzoek kent enkele beperkingen waarmee rekening moet worden gehouden bij de interpretatie van de resultaten. De belangrijkste beperking is het relatief lage aantal respondenten dat deelnam aan de vragenlijst. Hierdoor zijn de resultaten slechts in beperkte mate generaliseerbaar naar alle programmeerlesgevers binnen het Vlaamse onderwijs.

Daarnaast zijn de gegevens uit de vragenlijst gebaseerd op anonieme zelf-rapportage en weerspiegelen zij daardoor mogelijk niet volledig de feitelijke opvattingen en onderwijspraktijken van lesgevers. Ook kan het vrijwillige online karakter van de bevraging geleid hebben tot een vorm van zelfselectiebias, waarbij voornamelijk lesgevers met een uitgesproken interesse in of mening over generatieve AI deelnamen aan het onderzoek.

Een bijkomend aandachtspunt betreft het verschil tussen de context van de literatuurstudie en die van het empirisch onderzoek. De beschikbare wetenschappelijke literatuur over generatieve AI in het onderwijs richt zich voornamelijk op het hoger onderwijs, terwijl het merendeel van de respondenten uit de vragenlijst actief is binnen het secundair onderwijs. Bijgevolg is voorzichtigheid geboden bij het formuleren van parallellen en contrasten tussen beide onderzoeksluiken.

4.5 Verder onderzoek

Deze masterpraktijkproef opent verschillende pistes voor toekomstig onderzoek. Zo lijkt het zinvol om verschillen tussen onderwijsniveaus en onderwijscontexten uitgebreider te onderzoeken. Daarbij kan ook gekeken worden naar verschillen tussen finaliteiten binnen het secundair onderwijs en tussen instellingen in het hoger- en volwassenenonderwijs.

Een andere vraag betreft de invloed van de beschikbare lestijd op het gebruik en de perceptie van generatieve AI. Een respondent uit de vragenlijst merkt namelijk op dat één lesuur programmeren per week onvoldoende is om een minimumniveau te bereiken voor het didactisch verantwoord gebruik van generatieve AI. Aangezien de vragenlijst geen gegevens bevat over het

aantal programmeeruren per week, kon dit binnen het huidige onderzoek niet verder worden onderzocht.

Daarnaast lijkt verder onderzoek naar de relatie tussen de ervaring van lesgevers en hun opvattingen over generatieve AI relevant. Hoewel de vragenlijst peilde naar het aantal jaren ervaring van lesgevers en hierin een duidelijke spreiding aanwezig was, liet het beperkte aantal respondenten geen statistisch onderbouwde analyses toe.

Ten slotte kan vervolgonderzoek nagaan hoe het gebruik en de perceptie van generatieve AI binnen het programmeeronderwijs evolueren over de tijd. Aangezien zowel generatieve AI als de bijbehorende onderwijspraktijken zich nog snel ontwikkelen, vormen de bevindingen van deze studie slechts een momentopname. Hoewel het anonieme karakter van de vragenlijst longitudinaal onderzoek op individueel niveau onmogelijk maakt, kan deze studie wel dienen als een eerste nulmeting, waarmee toekomstige bevestigingen op groepsniveau kunnen worden vergeleken.

5. Conclusie

5.1 Beantwoording van de onderzoeksvragen

De eerste onderzoeksvraag onderzocht hoe generatieve AI het programmeeronderwijs beïnvloedt en welke didactische strategieën kunnen worden ingezet om de kansen ervan te benutten en de uitdagingen ervan te beheersen. Uit de literatuurstudie blijkt dat generatieve AI mogelijkheden biedt voor onder meer gepersonaliseerde ondersteuning, snellere feedback, differentiatie en motivatie. Tegelijk brengt het gebruik ervan uitdagingen met zich mee, waaronder een risico op fraude, afhankelijkheid en verminderde ontwikkeling van probleemoplossende vaardigheden. De literatuur wijst erop dat deze kansen en uitdagingen in belangrijke mate worden beïnvloed door de wijze waarop generatieve AI didactisch wordt geïntegreerd, onder meer via aanpassingen aan opdrachten en evaluatievormen, en door een kritische en doelgerichte inzet ervan.

De tweede onderzoeksvraag onderzocht in welke mate en op welke manier generatieve AI wordt gebruikt binnen het Vlaamse programmeeronderwijs en welke percepties en ervaringen lesgevers hierover hebben. De resultaten tonen aan dat generatieve AI reeds door een aanzienlijk deel van de bevraagde lesgevers wordt ingezet, zowel ter ondersteuning van het lesgeven als binnen leeractiviteiten. De percepties van lesgevers zijn gemengd en omvatten zowel positieve verwachtingen met betrekking tot ondersteuning bij het programmeren en lesvoorbereiding, als bezorgdheden omtrent afhankelijkheid, het behoud van fundamentele programmeervaardigheden, evaluatie en de authenticiteit van ingediende oplossingen. De bevindingen sluiten in grote lijnen aan bij de inzichten uit de literatuur.

5.2 Aanbevelingen

Generatieve AI heeft in korte tijd haar intrede gedaan in het programmeeronderwijs en lijkt daarin een blijvende rol te zullen spelen. Een centrale opdracht bestaat er daarom in lerenden niet alleen programmeervaardigheden bij te brengen, maar hen ook te leren beoordelen hoe, wanneer en onder welke voorwaarden generatieve AI zinvol kan worden ingezet. Tegen deze achtergrond worden hieronder enkele aanbevelingen geformuleerd voor de verschillende aggregatieniveaus binnen het Vlaamse onderwijs.

Op het microniveau kan deze studie lesgevers ondersteunen bij het ontwikkelen of verfijnen van hun visie en aanpak met betrekking tot generatieve AI in het programmeeronderwijs. De combinatie van inzichten uit de literatuur en ervaringen uit de onderwijspraktijk biedt daarbij aanknopingspunten voor verdere professionele ontwikkeling en verdieping.

Op het mesoniveau wijzen de resultaten niet op een uitgesproken vraag naar bijkomende ondersteuning. Desondanks kunnen onderwijsinstellingen een rol spelen door duidelijke richtlijnen aan te reiken en de integratie van generatieve AI binnen het onderwijs te ondersteunen.

Op het macroniveau is een belangrijkere rol weggelegd voor beleidsmakers en ondersteunende organisaties. Zij kunnen investeren in professionalisering rond generatieve AI, richtlijnen en ondersteuningsinitiatieven uitwerken, en platformen faciliteren die samenwerking, kennisuitwisseling en de verspreiding van didactisch materiaal en praktijkervaringen bevorderen.

De toekomst van het programmeeronderwijs zal niet uitsluitend worden bepaald door technologische ontwikkelingen, maar ook door de keuzes die lesgevers, onderwijsinstellingen en beleidsmakers vandaag maken.

Literatuurlijst

- Avouris, N., Sgarbas, K., Caridakis, G., & Sintoris, C. (2025). Teaching Introduction to Programming in the times of AI: A case study of a course re-design. *Proceedings of the 12th Panhellenic Conference on Computer Science Education*, 158–167. <https://doi.org/10.12681/cetpe.9223>
- Banh, L., & Strobel, G. (2023). Generative artificial intelligence. *Electronic Markets*, 33(1), 63. <https://doi.org/10.1007/s12525-023-00680-1>
- Beale, R. (2025). Computer Science Education in the Age of Generative AI. *arXiv preprint*. <https://doi.org/10.48550/arXiv.2507.02183>
- Becker, B. A., Denny, P., Finnie-Ansley, J., Luxton-Reilly, A., Prather, J., & Santos, E. A. (2023). Programming Is Hard – Or at Least It Used to Be: Educational Opportunities and Challenges of AI Code Generation. *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, 500–506. <https://doi.org/10.1145/3545945.3569759>
- Bedrina, O. (2025, 15 oktober). *The State of Developer Ecosystem 2025: Coding in the Age of AI, New Productivity Metrics, and Changing Realities*. JetBrains Research. <https://blog.jetbrains.com/research/2025/10/state-of-developer-ecosystem-2025>
- Boguslawski, S., Deer, R., & Dawson, M. G. (2025). Programming education and learner motivation in the age of generative AI: student and educator perspectives. *Information and Learning Sciences*, 126(1-2), 91–109. <https://doi.org/10.1108/ILS-10-2023-0163>
- Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., Brynjolfsson, E., Buch, S., Card, D., Castellon, R., Chatterji, N., Chen, A., Creel, K., Davis, J. Q., Demszky, D., ... Liang, P. (2021). On the Opportunities

- and Risks of Foundation Models. *arXiv preprint*. <https://doi.org/10.48550/arXiv.2108.07258>
- Chang, H.-F., Shirazi, M. S., Cao, L., & Mobasser, S. K. (2025). Coding With AI: From a Reflection on Industrial Practices to Future Computer Science and Software Engineering Education. *arXiv preprint*. <https://doi.org/10.48550/arXiv.2512.23982>
- Chen, M., Tworek, J., Jun, H., Yuan, Q., Ponde de Oliveira Pinto, H., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., . . . Zaremba, W. (2021). Evaluating Large Language Models Trained on Code. *arXiv preprint*. <https://doi.org/10.48550/arXiv.2107.03374>
- Chung, C.-J. (2026). Ensuring Computer Science Learning in the AI Era: Open Generative AI Policies and Assignment-Driven Written Quizzes. *arXiv preprint*. <https://doi.org/10.48550/arXiv.2601.17024>
- da Silva, C. A. G., Ramos, F. N., de Moraes, R. V., & dos Santos, E. L. (2024). ChatGPT: Challenges and Benefits in Software Programming for Higher Education. *Sustainability*, 16(3), 1245. <https://doi.org/10.3390/su16031245>
- Deckmyn, D. (2025, 2 augustus). *Is Vlaanderen klaar voor de impact van AI op de jobmarkt?* De Standaard. <https://standaard.be/78818097.html>
- Decock, S. (2025, 28 december). *Waarom de arbeidsmarkt slabakt: "Het vet is van de soep"*. De Standaard. <https://standaard.be/119689051.html>
- Delphino, K. (2025). Assessing the Prevalence of AI-assisted Cheating in Programming Courses: A Pilot Study. *arXiv preprint*. <https://doi.org/10.48550/arXiv.2507.06438>
- Denny, P., Prather, J., Becker, B. A., Finnie-Ansley, J., Hellas, A., Leinonen, J., Luxton-Reilly, A., Reeves, B. N., Santos, E. A., & Sarsa, S. (2024). Computing Education in the Era of Generative AI. *Communications of the ACM*, 67(2), 56–67. <https://doi.org/10.1145/3624720>
- Denny, P., Sarsa, S., Hellas, A., & Leinonen, J. (2022). Robosourcing Educational Resources – Leveraging Large Language Models for Learner-sourcing. *arXiv preprint*. <https://doi.org/10.48550/arXiv.2211.04715>
- Deriba, F. G., Sanusi, I. T., Campbell, O. O., & Oyelere, S. S. (2024). Computer Programming Education in the Age of Generative AI:

- Insights from Empirical Research. <https://doi.org/10.2139/ssrn.4891302>
- Diamantopoulos, A., Sintoris, C., Demetriadis, S., & Avouris, N. (2025). Feedback to Students and Instructors of a Programming Course Through a Large Language Model. *Proceedings of the 14th Panhellenic Conference on ICT in Education*, 1159–1168. <https://doi.org/10.12681/cetpe.9388>
- Er, E., Akçapınar, G., Bayazit, A., Noroozi, O., & Banihashem, S. K. (2025). Assessing student perceptions and use of instructor versus AI-generated feedback. *British Journal of Educational Technology*, 56(3), 1074–1091. <https://doi.org/10.1111/bjet.13558>
- Fan, G., Liu, D., Zhang, R., & Pan, L. (2025). The impact of AI-assisted pair programming on student motivation, programming anxiety, collaborative learning, and programming performance: A comparative study with traditional pair programming and individual approaches. *International Journal of STEM Education*, 12, 16. <https://doi.org/10.1186/s40594-025-00537-3>
- Feuerriegel, S., Hartmann, J., Janiesch, C., & Zschech, P. (2024). Generative AI. *Business & Information Systems Engineering*, 66(1), 111–126. <https://doi.org/10.1007/s12599-023-00834-7>
- Finnie-Ansley, J., Denny, P., Becker, B. A., Luxton-Reilly, A., & Prather, J. (2022). The Robots Are Coming: Exploring the Implications of OpenAI Codex on Introductory Programming. *Proceedings of the 24th Australasian Computing Education Conference*, 10–19. <https://doi.org/10.1145/3511861.3511863>
- Fraser, R. (2014). Collaboration, Collusion and Plagiarism in Computer Science Coursework. *Informatics in Education*, 13(2), 179–195. <https://doi.org/10.15388/infedu.2014.10>
- Fu, Y., Liang, P., Tahir, A., Li, Z., Shahin, M., Yu, J., & Chen, J. (2025). Security Weaknesses of Copilot-Generated Code in GitHub Projects: An Empirical Study. *ACM Transactions on Software Engineering and Methodology*, 34(8), 218. <https://doi.org/10.1145/3716848>
- Garg, A., Nisumba Soodhani, K., & Rajendran, R. (2025). Enhancing data analysis and programming skills through structured prompt training: The impact of generative AI in engineering education. *Computers*

- and *Education: Artificial Intelligence*, 8, 100380. <https://doi.org/10.1016/j.caeai.2025.100380>
- Groothuijsen, S., van den Beemt, A., Remmers, J. C., & van Meeuwen, L. W. (2024). AI chatbots in programming education: Students' use in a scientific computing course and consequences for learning. *Computers and Education: Artificial Intelligence*, 7, 100290. <https://doi.org/10.1016/j.caeai.2024.100290>
- Güner, H., & Er, E. (2025). AI in the classroom: Exploring students' interaction with ChatGPT in programming learning. *Education and Information Technologies*, 30, 12681–12707. <https://doi.org/10.1007/s10639-025-13337-7>
- Hampton, C. (2023, 7 februari). *'I'll just stop grading': Students, faculty reflect on ChatGPT after computer science course turns to paper exams*. The Dartmouth. <https://thedartmouth.com/article/2023/02/hampton-ill-just-stop-grading-students-faculty-reflect-on-chatgpt-after-computer-science-course-turns-to-paper-exams>
- He, K., Zhang, X., Cai, P., Liu, M., Wang, Y., Wang, C., Huang, K., Chen, B., Peng, X., & Zheng, Z. (2026). Bridging Generation and Training: A Systematic Review of Quality Issues in LLMs for Code. *arXiv preprint*. <https://doi.org/10.48550/arXiv.2605.05267>
- Huang, K. (2023, 16 januari). *Alarmed by A.I. Chatbots, Universities Start Re-vamping How They Teach*. The New York Times. <http://nytimes.com/2023/01/16/technology/chatgpt-artificial-intelligence-universities.html>
- Jukiewicz, M. (2024). The future of grading programming assignments in education: The role of ChatGPT in automating the assessment and feedback process. *Thinking Skills and Creativity*, 52, 101522. <https://doi.org/10.1016/j.tsc.2024.101522>
- Kandaswamy, R. (2025, 13 november). *Be AI-First to Outperform Your Competition by 2028*. Gartner. <https://gartner.com/en/articles/ai-first>
- Keuning, H., Alpizar-Chacon, I., Lykourentzou, I., Beehler, L., Köppe, C., de Jong, I., & Sosnovsky, S. (2024). Students' Perceptions and Use of Generative AI Tools for Programming Across Different Computing Courses. *Proceedings of the 24th Koli Calling International Conference*

- on *Computing Education Research*, 14. <https://doi.org/10.1145/3699538.3699546>
- Leinonen, J., Hellas, A., Sarsa, S., Reeves, B., Denny, P., Prather, J., & Becker, B. A. (2023). Using Large Language Models to Enhance Programming Error Messages. *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, 563–569. <https://doi.org/10.1145/3545945.3569770>
- Lepp, M., & Kaimre, J. (2025). Does generative AI help in learning programming: Students’ perceptions, reported use and relation to performance. *Computers in Human Behavior Reports*, 18, 100642. <https://doi.org/10.1016/j.chbr.2025.100642>
- Li, Z., Liu, M., Li, A., He, K., Wang, Y., Peng, X., & Zheng, Z. (2025). A Preliminary Study on the Robustness of Code Generation by Large Language Models. *arXiv preprint*. <https://doi.org/10.48550/arXiv.2503.20197>
- Lo, L. S. (2023). The CLEAR path: A framework for enhancing information literacy through prompt engineering. *The Journal of Academic Librarianship*, 49(4), 102720. <https://doi.org/10.1016/j.acalib.2023.102720>
- Malinka, K., Perešíni, M., Firc, A., Hujňák, O., & Januš, F. (2023). On the Educational Impact of ChatGPT: Is Artificial Intelligence Ready to Obtain a University Degree? *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1*, 47–53. <https://doi.org/10.1145/3587102.3588827>
- Manley, E. D., Urness, T., Migunov, A., & Reza, M. A. (2024). Examining Student Use of AI in CS1 and CS2. *Journal of Computing Sciences in Colleges*, 39(6), 41–51. <https://dl.acm.org/doi/abs/10.5555/3665464.3665469>
- McDowell, C., Werner, L., Bullock, H., & Fernald, J. (2002). The Effects of Pair-Programming on Performance in an Introductory Programming Course. *Proceedings of the 33rd SIGCSE technical symposium on Computer science education*, 38–42. <https://doi.org/10.1145/563340.563353>
- Miao, C. (2026, 14 januari). *Junior Developers in the Age of AI*. Thoughtful Engineering. <http://thoughtfuleng.substack.com/p/junior-developers-in-the-age-of-ai>

- Novak, M., & Maček, M. (2026). AI Tools in Programming Education: Student Perspectives and Usage Trends. *Journal of Information and Organizational Sciences*, 50(1), 83–104. <https://doi.org/10.31341/jios.50.1.5>
- Osmani, A. (2025, 10 mei). *AI Won't Kill Junior Devs - But Your Hiring Strategy Might*. Elevate. <https://addyo.substack.com/p/ai-wont-kill-junior-devs-but-your>
- Pankiewicz, M., & Baker, R. S. (2024). Navigating Compiler Errors with AI Assistance - A Study of GPT Hints in an Introductory Programming Course. *Proceedings of the 2024 Conference on Innovation and Technology in Computer Science Education V. 1*, 94–100. <https://doi.org/10.1145/3649217.3653608>
- Perry, N., Srivastava, M., Kumar, D., & Boneh, D. (2023). Do Users Write More Insecure Code with AI Assistants? *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, 2785–2799. <https://doi.org/10.1145/3576915.3623157>
- Peters, J. (2025, 8 april). *Shopify CEO says no new hires without proof AI can't do the job*. The Verge. <https://theverge.com/news/644943/shopify-ceo-memo-ai-hires-job>
- Prather, J., Denny, P., Leinonen, J., Becker, B. A., Albluwi, I., Craig, M., Keuning, H., Kiesler, N., Kohn, T., Luxton-Reilly, A., MacNeil, S., Petersen, A., Pettit, R., Reeves, B. N., & Savelka, J. (2023). The Robots Are Here: Navigating the Generative AI Revolution in Computing Education. *Proceedings of the 2023 Working Group Reports on Innovation and Technology in Computer Science Education*, 108–159. <https://doi.org/10.1145/3623762.3633499>
- Rack, C. (2026). Programming Languages as Intermediate Representations in the AI-Generated-Code Era [Manuscript in voorbereiding]. <https://constantin-rack.com/papers/#ir-ai-generated-code-era>
- Roose, K. (2025, 30 maart). *For Some Recent Graduates, the A.I. Job Apocalypse May Already Be Here*. The New York Times. <https://nytimes.com/2025/05/30/technology/ai-jobs-college-graduates.html>
- Rosenblatt, K. (2023a, 5 januari). *New York City public schools ban ChatGPT on devices and networks*. NBC News. <https://nbcnews.com/tech/tech-news/rcna64446>

- Rosenblatt, K. (2023b, 18 mei). *New York City public schools remove Chat-GPT ban*. NBC News. <https://nbcnews.com/tech/tech-news/rcna85089>
- Sağlam, T., & Schmid, L. (2025). Evaluating Software Plagiarism Detection in the Age of AI: Automated Obfuscation and Lessons for Academic Integrity. *arXiv preprint*. <https://doi.org/10.48550/arXiv.2505.20158>
- Sarsa, S., Denny, P., Hellas, A., & Leinonen, J. (2022). Automatic Generation of Programming Exercises and Code Explanations Using Large Language Models. *Proceedings of the 2022 ACM Conference on International Computing Education Research - Volume 1*, 27–43. <https://doi.org/10.1145/3501385.3543957>
- Savelka, J., Agarwal, A., Bogart, C., Song, Y., & Sakr, M. (2023). Can Generative Pre-trained Transformers (GPT) Pass Assessments in Higher Education Programming Courses? *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1*, 117–123. <https://doi.org/10.1145/3587102.3588792>
- Sengul, C., Neykova, R., & Destefanis, G. (2024). Software engineering education in the era of conversational AI: current trends and future directions. *Frontiers in Artificial Intelligence*, 7, 1436350. <https://doi.org/10.3389/frai.2024.1436350>
- Simon, Sheard, J., Morgan, M., Petersen, A., Settle, A., Sinclair, J., Cross, G., & Riedesel, C. (2016). Negotiating the Maze of Academic Integrity in Computing Education. *Proceedings of the 2016 ITiCSE working group reports*, 57–80. <https://doi.org/10.1145/3024906.3024910>
- Siu, E., & Ghosal, S. (2025, 12 september). *Gen Z workers are teaching older colleagues AI – and reshaping office culture, survey shows*. CNBC. <https://cnbc.com/2025/09/12/iwg-report-gen-z-employees-are-coaching-older-colleagues-to-use-ai.html>
- Stack Overflow. (2025). *Stack Overflow Developer Survey 2025 – AI*. Stack Overflow. <https://survey.stackoverflow.co/2025/ai>
- Stokel-Walker, C. (2024, 31 juli). *Why it sucks to be a junior developer right now*. LeadDev. <https://leaddev.com/hiring/why-it-sucks-be-junior-developer-right-now>
- Stutz, P., Elixhauser, M., Grubinger-Preiner, J., Linner, V., Reibersdorfer-Adelsberger, E., Traun, C., Wallentin, G., Wöhs, K., & Zuberbühler, T. (2023). Ch(e)atGPT? An Anecdotal Approach Addressing the

- Impact of ChatGPT on Teaching and Learning GIScience. *GI_Forum*, 11(1), 140–147. https://doi.org/10.1553/giscience2023_01_s140
- Suh, H., Tafreshipour, M., Li, J., Bhattiprolu, A., & Ahmed, I. (2025). An Empirical Study on Automatically Detecting AI-Generated Source Code: How Far Are We? *Proceedings of the IEEE/ACM 47th International Conference on Software Engineering*, 859–871. <https://doi.org/10.1109/ICSE55347.2025.00064>
- Vaithilingam, P., Zhang, T., & Glassman, E. L. (2022). Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models. *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems*, Article 332. <https://doi.org/10.1145/3491101.3519665>
- Vivian, R. (2025). Coding with ChatGPT: Empirical Evidence of Cognitive Offloading in Computer Science Education. *Clareus Scientific Science and Engineering*, 2(10), 10–19. <https://doi.org/10.70012/CSSE.02.062>
- Vukojić, M., & Krstić, J. (2023). ChatGPT in programming education: ChatGPT as a programming assistant. *InspirED Teachers' Voice*, 2023(1), 7–13.
- Welsh, M. (2023). The End of Programming. *Communications of the ACM*, 66(1), 34–35. <https://doi.org/10.1145/3570220>
- Wieser, M., Schöffmann, K., Stefanics, D., Bollin, A., & Pasterk, S. (2023). Investigating the Role of ChatGPT in Supporting Text-Based Programming Education for Students and Teachers. In *Informatics in Schools. Beyond Bits and Bytes: Nurturing Informatics Intelligence in Education* (pp. 40–53). https://doi.org/10.1007/978-3-031-44900-0_4
- Yegge, S. (2024, 24 juni). *The death of the junior developer*. Sourcegraph. <https://sourcegraph.com/blog/the-death-of-the-junior-developer>
- Yilmaz, R., & Yilmaz, F. G. K. (2023). The effect of generative artificial intelligence (AI)-based tool use on students' computational thinking skills, programming self-efficacy and motivation. *Computers and education: Artificial intelligence*, 4, 100147. <https://doi.org/10.1016/j.caeai.2023.100147>
- Zviel-Girshin, R. (2024). The Good and Bad of AI Tools in Novice Programming Education. *Education Sciences*, 14(10), 1089. <https://doi.org/10.3390/educsci14101089>

A. Vragenlijst: vragen

Deze bijlage bevat de volledige vragenlijst, met als titel *Generatieve AI in het Vlaamse programmeeronderwijs*, zoals die via het enquêteplatform Qualtrics van de Universiteit Gent aan de deelnemers werd gepresenteerd.⁴

Vragen en antwoordopties aangeduid met * zijn verplicht in te vullen. Vragen aangeduid met $\mathbb{A}$ maken gebruik van een symmetrische en gebalanceerde Likertschaal. Antwoordopties aangeduid met $\mathbb{Q}$ zijn exclusief en kunnen niet gecombineerd worden met andere antwoordopties. Vragen aangeduid met $\mathbb{Z}$ presenteren de antwoordopties in willekeurige volgorde, met uitzondering van verplichte of exclusieve opties, die hun positie behouden.

Introductie

Wat is de impact van Generatieve AI op programmeeronderwijs, en welke aanbevelingen kunnen daaruit worden afgeleid?

Als student in de Educatieve Master (wetenschappen en technologie: informatica) aan de Universiteit Gent onderzoek ik in het kader van mijn masterpraktijkproef de impact van Generatieve AI (GenAI) op het programmeeronderwijs. Het doel van deze vragenlijst is in kaart te brengen hoe GenAI momenteel wordt gebruikt en welke gevolgen dit heeft voor lessen, opdrachten, en evaluaties. Op basis van uw antwoorden en wetenschappelijke literatuur wil ik praktische adviezen formuleren voor leerkrachten.

Privacy en toestemming

U wordt uitgenodigd om deel te nemen aan een onderzoek over de impact van generatieve AI op het Vlaamse programmeeronderwijs.

Verantwoordelijke voor de verwerking ...

⁴Voor de leesbaarheid werden in deze bijlage paragrafen toegevoegd en werden delen van de privacydisclaimer weggelaten.

Welke gegevens worden verzameld? ...

Hoe worden de gegevens verwerkt? ...

Rechtsgrond voor de verwerking ...

Bewaartermijn ...

Uw rechten ...

Door hieronder “Ja” te kiezen, geeft u expliciet toestemming voor deelname aan dit onderzoek en de verwerking van uw persoonsgegevens zoals hierboven beschreven. *

- ☐ Ja, ik geef toestemming om deel te nemen aan dit onderzoek
- ☐ Nee, ik geef geen toestemming⁵

Screening

Bevestigt u dat u op dit moment actief bent als leerkracht programmeeronderwijs binnen het Vlaamse onderwijs? *

- ☐ Ja
- ☐ Nee⁶

Achtergrond en context

Q1. In welke Vlaamse onderwijscontext geeft u programmeeronderwijs? *

- ☐ Secundair onderwijs
- ☐ Hoger onderwijs
- ☐ Volwassenenonderwijs
- ☐ * Andere:

Q2. Hoeveel jaar ervaring heeft u met het geven van programmeeronderwijs? *

- ☐ Minder dan 1 jaar
- ☐ 1–3 jaar
- ☐ 4–7 jaar
- ☐ 8–12 jaar
- ☐ 13–20 jaar
- ☐ Meer dan 20 jaar

Q3. Met welke programmeertalen werkt u vooral in uw lessen?

.....

⁵Deze antwoordoptie leidt naar het einde van de vragenlijst.

⁶Idem.

Gebruik van generatieve AI door lesgevers

Q4. Hoe vaak gebruikt u zelf GenAI als leerkracht in het programmeeronderwijs? * 41

- ☐ Nooit
- ☐ Zelden
- ☐ Af en toe
- ☐ Regelmatig
- ☐ Zeer vaak

Q5. Welke GenAI-tools gebruikt u zelf als leerkracht in het programmeeronderwijs?⁷ * 23

- ☐ ChatGPT
- ☐ Gemini
- ☐ Claude
- ☐ Copilot
- ☐ Replit AI
- ☐ DeepSeek
- ☐ * Andere:

Q6. Voor welke doeleinden gebruikt u zelf GenAI in uw programmeeronderwijspraktijk?⁸ 23

- ☐ Toets- of opdrachtmateriaal genereren of herwerken
- ☐ Inspiratie of voorbeeldcode genereren
- ☐ Lesmateriaal of cursusnota's voorbereiden
- ☐ Code verbeteren of optimaliseren
- ☐ Feedback geven aan leerlingen
- ☐ * Andere:

Gebruik van generatieve AI door lerenden

Q7. Hoeveel leerlingen gebruiken naar schatting GenAI tijdens de programmeerlessen? * 41

- ☐ (Bijna) niemand
- ☐ Ongeveer een kwart
- ☐ Ongeveer de helft
- ☐ Ongeveer driekwart
- ☐ (Bijna) iedereen

⁷Deze vraag verschijnt enkel bij andere antwoorden dan "Nooit" op Q4.

⁸Idem.

Toelichting bij uw antwoord (optioneel)

.....

Q8. Over de leerlingen die GenAI gebruiken tijdens de programmeerlessen: hoe vaak gebruikt een typische leerling GenAI, naar schatting?⁹ 

- ☐ Nooit
- ☐ Zelden
- ☐ Af en toe
- ☐ Regelmatig
- ☐ Zeer vaak

Toelichting bij uw antwoord (optioneel)

.....

Q9. Indien u programmeeropdrachten buiten de klas geeft: hoe vaak vermoedt u dat leerlingen GenAI gebruiken voor deze taken? 

- ☐ Nooit
- ☐ Zelden
- ☐ Af en toe
- ☐ Regelmatig
- ☐ Zeer vaak

Toelichting bij uw antwoord (optioneel)

.....

Impact van generatieve AI op lerenden

Q10. Hoe beïnvloedt het gebruik van GenAI de programmeervaardigheden van leerlingen volgens u? 

- ☐ Zeer negatief
- ☐ Enigszins negatief
- ☐ Geen effect
- ☐ Enigszins positief
- ☐ Zeer positief

Toelichting bij uw antwoord (optioneel)

.....

Q11. In welke mate bent u bezorgd over de authenticiteit van leerlingenwerk door GenAI? 

- ☐ Helemaal niet bezorgd

⁹Deze vraag verschijnt enkel bij andere antwoorden dan “(Bijna) niemand” op Q7.

- ☐ Eerder niet bezorgd
- ☐ Neutraal
- ☐ Eerder bezorgd
- ☐ Zeer bezorgd

Q12. Heeft u andere bezorgdheden over het gebruik van GenAI in het programmeeronderwijs?

.....

Kansen en uitdagingen van generatieve AI

Q13. Welke uitdagingen ervaart u bij het gebruik van GenAI in programmeeronderwijs?

.....

Q14. Welke kansen ziet u bij het gebruik van GenAI in programmeeronderwijs?

.....

Q15. Hoe verandert GenAI volgens u de rol van de programmeerleerkracht?

.....

Toelating van generatieve AI

Q16. Laat u het gebruik van GenAI toe tijdens de programmeerlessen? * 

- ☐ Nooit
- ☐ Zelden
- ☐ Af en toe
- ☐ Regelmatig
- ☐ Zeer vaak

Toelichting bij uw antwoord (optioneel)

.....

Q17. Laat u het gebruik van GenAI toe tijdens toetsen of examens? * 

- ☐ Nooit
- ☐ Zelden
- ☐ Af en toe
- ☐ Regelmatig
- ☐ Zeer vaak

Toelichting bij uw antwoord (optioneel)

.....

Q18. Laat u het gebruik van GenAI toe bij programmeeropdrachten buiten de klas? * 4a

- ☐ Niet van toepassing
- ☐ Nooit
- ☐ Zelden
- ☐ Af en toe
- ☐ Regelmatig
- ☐ Zeer vaak

Toelichting bij uw antwoord (optioneel)

.....

Aanpassingen aan het programmeeronderwijs

Q19. Heeft u uw manier van lesgeven al aangepast om rekening te houden met GenAI? *

- ☐ Ja, altijd
- ☐ Ja, soms
- ☐ Nee, maar ik overweeg het
- ☐ Nee, en ik overweeg het niet

Q20. Op welke manier(en) heeft u uw manier van lesgeven al aangepast om rekening te houden met GenAI?¹⁰

.....

Q21. Heeft u uw programmeeropdrachten al aangepast om rekening te houden met GenAI? *

- ☐ Ja, altijd
- ☐ Ja, soms
- ☐ Nee, maar ik overweeg het
- ☐ Nee, en ik overweeg het niet

Q22. Op welke manier(en) heeft u uw programmeeropdrachten al aangepast om rekening te houden met GenAI?¹¹

.....

Evaluatie en toekomstige vaardigheden

Q23. Op welke manier evalueert u leerlingen die GenAI gebruiken bij programmeeropdrachten? *

¹⁰Deze vraag verschijnt enkel bij antwoorden “Ja, altijd” of “Ja, soms” op Q19.

¹¹Deze vraag verschijnt enkel bij antwoorden “Ja, altijd” of “Ja, soms” op Q21.

- ☐ Ik beoordeel hun werk op dezelfde manier als bij andere leerlingen
- ☐ Ik pas mijn beoordeling aan wanneer AI is gebruikt (bijvoorbeeld strengere criteria, extra controle, nadruk op begrip)
- ☐ Ik vraag een mondelinge toelichting of een code walkthrough om hun inzicht te controleren
- ☐ * Andere:

Q24. Welke strategieën acht u meest effectief om opdrachten ‘AI-proof’ te maken? ✂

- ☐ Mondelinge toelichting of code-walkthroughs
- ☐ Meer reflectievragen
- ☐ Meer nadruk op proces (bv. versiegeschiedenis)
- ☐ Documenteren van AI-gebruik
- ☐ Samenwerkend leren (pair programming)
- ☐ Meer open of contextuele opdrachten
- ☐ ↻ Ik zie geen mogelijkheden om opdrachten ‘AI-proof’ te maken
- ☐ ↻ Ik zie geen nood om opdrachten ‘AI-proof’ te maken
- ☐ * Andere:

Q25. Welke vaardigheden moeten leerlingen volgens u méér ontwikkelen nu GenAI beschikbaar is? * ✂

- ☐ Probleemanalyse (programmeervraagstukken begrijpen en ontleden vóór het programmeren)
- ☐ Code begrijpen en beoordelen
- ☐ Debugging
- ☐ Documenteren en redeneren over oplossingen
- ☐ Prompt engineering (effectieve opdrachten en instructies formuleren voor GenAI-tools)
- ☐ ↻ Ik zie geen nood om bepaalde vaardigheden extra te benadrukken
- ☐ * Andere:

Professionalisering

Q26. Voelt u zich voldoende voorbereid om GenAI zinvol te integreren in het programmeeronderwijs? *

- ☐ ↻ Ja
- ☐ Nee, ik heb training nodig
- ☐ Nee, ik heb richtlijnen nodig
- ☐ Nee, ik heb ondersteuning vanuit de school/instelling nodig

Toelichting bij uw antwoord (optioneel)

.....

Slot

Q27. Heeft u nog opmerkingen of ervaringen over GenAI in programmeeronderwijs die u wilt delen?

.....

Q28. Zijn er nog andere opmerkingen of suggesties (over deze vragenlijst of het onderzoek) die u graag wilt meegeven?

.....

Q29. Laat uw emailadres achter indien ik u mag contacteren voor opvolging of om de resultaten te delen.¹²

.....

¹²Validatie: emailadres.

B. Vragenlijst: antwoorden

Deze bijlage bevat een volledig overzicht van de antwoorden op de vragenlijst *Generatieve AI in het Vlaamse programmeeronderwijs*, die van 8 januari 2026 tot en met 8 mei 2026 via het enquêteplatform Qualtrics van de Universiteit Gent beschikbaar was voor programmeerleerkrachten in het Vlaamse onderwijs, zoals gerapporteerd door Qualtrics.

Vragen en antwoordopties aangeduid met $\star$ zijn verplicht in te vullen. Vragen aangeduid met $\mathbb{A}$ maken gebruik van een symmetrische en gebalanceerde Likertschaal. Antwoordopties aangeduid met $\mathbb{Q}$ zijn exclusief en kunnen niet gecombineerd worden met andere antwoordopties. Vragen aangeduid met $\mathbb{R}$ presenteren de antwoordopties in willekeurige volgorde, met uitzondering van verplichte of exclusieve opties, die hun positie behouden.

Uit de 52 inzendingen werden enkel de 37 volledig doorlopen vragenlijsten in deze antwoorden opgenomen ($N = 37$), ongeacht eventuele niet-beantwoorde optionele vragen. Vrije tekstinvoer werd in deze bijlage redactioneel aangepast (inclusief taalkundige correcties, inkorting en anonimisering) ter bevordering van aggregatie, uniformiteit en duidelijkheid, met behoud van de feitelijke inhoud. Cursieve tekst werd rechtstreeks overgenomen uit de oorspronkelijke antwoorden.

Het aantal respondenten dat een vraag heeft beantwoord wordt weergegeven met n . Het rekenkundig gemiddelde wordt aangeduid met $\bar{x}$, de standaarddeviatie met SD en de variantie met Var . Bij meerkeuzevragen kan het aantal geselecteerde antwoordopties groter zijn dan n , terwijl de som van de percentages per antwoordoptie hoger kan liggen dan 100%.

Achtergrond en context

Q1. In welke Vlaamse onderwijscontext geeft u programmeeronderwijs? *

☐ Secundair onderwijs	33	(89%)
☐ Hoger onderwijs	1	(3%)
☐ Volwassenenonderwijs	4	(11%)
☐ * Andere	0	(0%)

$n = 37$

Q2. Hoeveel jaar ervaring heeft u met het geven van programmeeronderwijs? *

☐ Minder dan 1 jaar	2	(5%)
☐ 1–3 jaar	9	(24%)
☐ 4–7 jaar	10	(27%)
☐ 8–12 jaar	2	(5%)
☐ 13–20 jaar	7	(19%)
☐ Meer dan 20 jaar	7	(19%)

$n = 37$

Q3. Met welke programmeertalen werkt u vooral in uw lessen?¹³ $n = 37$

Python: 27; C#: 13; Java, JavaScript: 7; PHP: 4; C++, Scratch: 3; PowerShell, R, VBA: 1.

Gebruik van generatieve AI door lesgevers

Q4. Hoe vaak gebruikt u zelf GenAI als leerkracht in het programmeeronderwijs? * 

☐ Nooit $\mapsto 1$	8	(22%)
☐ Zelden $\mapsto 2$	3	(8%)
☐ Af en toe $\mapsto 3$	10	(27%)
☐ Regelmatig $\mapsto 4$	12	(32%)
☐ Zeer vaak $\mapsto 5$	4	(11%)

$n = 37$; $\bar{x} = 3,03$; $SD = 1,30$; $Var = 1,70$

Q5. Welke GenAI-tools gebruikt u zelf als leerkracht in het programmeeronderwijs? * 

☐ ChatGPT	24	(83%)
☐ Gemini	12	(41%)
☐ Claude	9	(31%)
☐ Copilot	9	(31%)
☐ Replit AI	3	(10%)

¹³Opmaaktalen, querytalen en pseudocode werden buiten beschouwing gelaten en het antwoord ‘React’ werd als JavaScript geïnterpreteerd.

☐ DeepSeek	0	(0%)
☐ * Andere	1	(3%)
Andere: Adobe Firefly.		$n = 29$

Q6. Voor welke doeleinden gebruikt u zelf GenAI in uw programmeeronderwijspraktijk? 

☐ Toets- of opdrachtmateriaal genereren of herwerken	26	(90%)
☐ Inspiratie of voorbeeldcode genereren	25	(86%)
☐ Lesmateriaal of cursusnota's voorbereiden	17	(59%)
☐ Code verbeteren of optimaliseren	14	(48%)
☐ Feedback geven aan leerlingen	4	(14%)
☐ * Andere	2	(7%)
Andere: antwoorden structureren op meer technische vragen; debuggen.		$n = 29$

Gebruik van generatieve AI door lerenden

Q7. Hoeveel leerlingen gebruiken naar schatting GenAI tijdens de programmeerlessen? * 

☐ (Bijna) niemand $\mapsto 1$	12	(32%)
☐ Ongeveer een kwart $\mapsto 2$	5	(14%)
☐ Ongeveer de helft $\mapsto 3$	6	(16%)
☐ Ongeveer driekwart $\mapsto 4$	4	(11%)
☐ (Bijna) iedereen $\mapsto 5$	10	(27%)
$n = 37$; $\bar{x} = 2,86$; $SD = 1,61$; $Var = 2,60$		

Toelichtingen:

- *(Bijna) niemand*: Leerlingen hebben geen mogelijkheid om gebruik te maken van GenAI.
- *(Bijna) niemand*: Gebruik van GenAI is niet toegestaan en enkel toegelaten indien expliciet in de oefening vermeld.
- *(Bijna) niemand*: Internettoegang wordt tijdens de les gedeactiveerd via Google Classroom, al wordt leerlingen occasioneel toegestaan om met behulp van GenAI te werken.
- *(Bijna) niemand*: Leerlingen geven eerlijk aan welk deelprobleem met behulp van GenAI werd opgelost.
- *(Bijna) niemand*: Idealiter maakt geen enkele leerling gebruik van GenAI, maar in de praktijk wordt het soms ingezet als vorm van spieken.
- *(Bijna) niemand*: Er wordt getracht te bewaken dat leerlingen tijdens de lessen geen GenAI gebruiken, maar er is geen controle op het gebruik wanneer zij thuis werken.
- *(Bijna) niemand*: Er wordt lesgegeven met zicht op de computerschermen van de leerlingen om het gebruik van GenAI tijdens de les te bewaken.
- *(Bijna) niemand*: Door het lage programmeerniveau van de leerlingen is het doorgaans onmiddellijk zichtbaar wanneer zij GenAI gebruiken.
- *Ongeveer een kwart*: Het gebruik van GenAI wordt niet actief aangeleerd; leerlingen moeten eerst zelfstandig leren programmeren alvorens gebruik

te maken van door GenAI gegenereerde code, al zoeken sommigen wel zelf ondersteuning via tools zoals ChatGPT.

- *Ongeveer een kwart*: De meeste leerlingen willen zelf leren programmeren, maar een aantal leerlingen probeert oefeningen snel op te lossen met behulp van GenAI zonder zelf na te denken.
- *Ongeveer de helft*: Het gebruik buiten de klas is moeilijk in te schatten, terwijl GenAI binnen de klas uitzonderlijk en op een doordachte manier wordt aangeleerd en gebruikt.
- *Ongeveer driekwart*: Het betreft een inschatting, aangezien niet alle leerlingen openlijk aangeven dat zij GenAI gebruiken. Het gebruik bevindt zich nog deels in een sfeer waarin het als ongeoorloofd wordt beschouwd.
- *(Bijna) iedereen*: Gebruik van GenAI is toegestaan tijdens de lessen en bij het maken van oefeningen, maar bij evaluatie moeten leerlingen een functionele en technische demo geven zodat kan worden nagegaan in welke mate zij de oplossing effectief begrijpen.
- *(Bijna) iedereen*: De inschatting is dat (bijna) alle leerlingen in zekere mate GenAI gebruiken, maar vaak zonder grondig begrip van de gegenereerde code. Hierdoor wordt het zelfstandig nadenken, fouten maken en probleemoplossend leren minder geoefend, terwijl dit essentieel is voor het ontwikkelen van programmeervaardigheden. GenAI kan ondersteunend zijn, maar niet als vervanging van het denkproces.

Q8. Over de leerlingen die GenAI gebruiken tijdens de programmeerlessen: hoe vaak gebruikt een typische leerling GenAI, naar schatting? 

☐ Nooit $\mapsto 1$	0	(0%)
☐ Zelden $\mapsto 2$	3	(12%)
☐ Af en toe $\mapsto 3$	7	(28%)
☐ Regelmatig $\mapsto 4$	10	(40%)
☐ Zeer vaak $\mapsto 5$	5	(20%)

$n = 25$; $\bar{x} = 3,68$; $SD = 0,93$; $Var = 0,86$

Toelichtingen:

- *Zelden*: Tijdens de les is er controle via Classroom.cloud, waardoor het gebruik van GenAI meestal goed beheersbaar blijft.
- *Regelmatig*: Wanneer leerlingen opdrachten niet zelfstandig kunnen oplossen, grijpen zij naar GenAI-hulpmiddelen.
- *Zeer vaak*: Leerlingen vallen snel terug op het gebruik van GenAI wanneer zij ergens over twijfelen.

Q9. Indien u programmeeropdrachten buiten de klas geeft: hoe vaak vermoedt u dat leerlingen GenAI gebruiken voor deze taken? 

☐ Nooit $\mapsto 1$	2	(6%)
☐ Zelden $\mapsto 2$	2	(6%)
☐ Af en toe $\mapsto 3$	4	(11%)
☐ Regelmatig $\mapsto 4$	14	(39%)
☐ Zeer vaak $\mapsto 5$	14	(39%)

$n = 36$; $\bar{x} = 4,00$; $SD = 1,11$; $Var = 1,22$

Toelichtingen:

- *Nooit*: Er worden geen programmeeropdrachten buiten de les gegeven.
- *Nooit*: Programmeeropdrachten worden bewust enkel tijdens de les gemaakt om gebruik van GenAI te vermijden.

- *Zelden*: Door het lage programmeerniveau van de leerlingen is het doorgaans onmiddellijk zichtbaar wanneer zij GenAI gebruiken.
- *Af en toe*: In het 4e en 5e jaar worden nog zelden taken thuis gegeven zodat leerlingen de basis zelfstandig moeten beheersen en er in de klas meer controle is op het gebruik ervan; in het 6e jaar wordt bij grotere projecten af en toe gebruik gemaakt van GenAI.
- *Regelmatig*: Er worden nog weinig programmeeropdrachten buiten de les gegeven.
- *Regelmatig*: Gebruik van GenAI wordt onder meer afgeleid uit kenmerken van de ingediende opdrachten.
- *Regelmatig*: Het gebruik van GenAI wordt mogelijk overschat, aangezien sommige leerlingen oplossingen van medeleerlingen overnemen in plaats van zelf GenAI te gebruiken.
- *Zeer vaak*: Sommige leerlingen gebruiken GenAI nauwelijks of hebben ingezien dat dit niet de juiste aanpak is, terwijl anderen het hardnekkig blijven gebruiken. Dit wordt vaak snel zichtbaar in de code en tijdens presentaties.
- *Zeer vaak*: Leerlingen gebruiken GenAI ook wanneer zij de oplossing zelf kunnen maken, bijvoorbeeld om de kwaliteit van hun code te controleren.
- *Zeer vaak*: Er worden hierdoor geen pure programmeeroefeningen meer buiten de klascontext gegeven.
- *Zeer vaak*: Leerlingen vinden het gebruik van GenAI normaal en maken er gemakkelijk gebruik van.
- *Zeer vaak*: Er wordt bewust meer oefentijd in de klas voorzien en minder huiswerk gegeven.
- *Zeer vaak*: De code is vaak herkenbaar als door GenAI gegenereerd of bevat technieken die nog niet werden behandeld.

Impact van generatieve AI op lerenden

Q10. Hoe beïnvloedt het gebruik van GenAI de programmeervaardigheden van leerlingen volgens u? * 

☐ Zeer negatief $\mapsto 1$	8	(22%)
☐ Enigszins negatief $\mapsto 2$	16	(43%)
☐ Geen effect $\mapsto 3$	5	(14%)
☐ Enigszins positief $\mapsto 4$	8	(22%)
☐ Zeer positief $\mapsto 5$	0	(0%)

$n = 37$; $\bar{x} = 2,35$; $SD = 1,05$; $Var = 1,09$

Toelichtingen:

- *Zeer negatief*: In lessen waar van nul voorkennis wordt uitgegaan, helpt het genereren van code zonder begrip niet bij het leerproces; dit is vergelijkbaar met het gebruik van automatische spellingcorrectie bij kinderen die net leren schrijven.
- *Zeer negatief*: Basisvaardigheden zoals algoritmisch denken worden moeilijker aangeleerd wanneer steeds naar GenAI wordt gegrepen, evenals ondersteunende vaardigheden zoals het analyseren en ontleden van opgaven, kritisch evalueren van oplossingen (testen/debuggen) en taakinitiatie wanneer het antwoord niet onmiddellijk duidelijk is.
- *Zeer negatief*: Door het ontbreken van zelfstandig oefenen worden competenties onvoldoende opgebouwd.

- *Zeer negatief*: Code genereren via GenAI is vergelijkbaar met navigatie in een auto: leerlingen leren er niets van bij. Dit effect is vooral zichtbaar bij leerlingen die snel naar GenAI grijpen uit onzekerheid en de leercurve niet durven aangaan.
- *Zeer negatief*: GenAI belemmert het zelfstandig nadenken over fouten en leidt tot afhankelijkheid.
- *Zeer negatief*: Leerlingen denken niet meer na en begrijpen de code vaak niet.
- *Enigszins negatief*: Gebruik van GenAI in een vroeg stadium zorgt ervoor dat een belangrijke leerfase in het programmeren wordt overgeslagen.
- *Enigszins negatief*: Hoewel het gebruik van GenAI positief kan zijn, gebruiken te veel leerlingen het als een snelle oplossing zonder begrip; het correct leren gebruiken van GenAI wordt daarom opgenomen in de leeropdrachten.
- *Enigszins negatief*: Sommige leerlingen grijpen te snel naar GenAI.
- *Enigszins negatief*: Leerlingen nemen gegenereerde code klakkeloos over zonder te controleren of deze correct is of voldoet aan de opdracht.
- *Enigszins negatief*: Leerlingen doen aan vibecoding en hebben geen inzicht in wat individuele lijnen code effectief doen.
- *Enigszins negatief*: Leerlingen begrijpen de output van GenAI niet altijd.
- *Enigszins negatief*: Leerlingen begrijpen soms niet wat er in de gegenereerde code staat.
- *Enigszins negatief*: Leerlingen moeten minder nadenken en begrijpen hun code soms niet meer.
- *Enigszins negatief*: Leerlingen denken niet meer na en nemen code over.
- *Enigszins negatief*: Leerlingen leren minder nadenken en worden minder creatief; de inspanning neemt af en de retentie van leerstof blijft beperkt.
- *Enigszins negatief*: Leerlingen die alle oefeningen tijdens de les met behulp van GenAI maken, vallen bij testen en proefwerken door de mand doordat zij onvoldoende inzicht hebben in programmacode en niet begrijpen hoe zij een opgave naar code moeten vertalen.
- *Enigszins negatief*: (Bijna) alle leerlingen gebruiken GenAI, maar vaak niet op de juiste manier. Ze laten code genereren en nemen die over zonder echt te begrijpen wat er gebeurt. Daardoor oefenen ze te weinig in zelf nadenken, fouten maken en problemen oplossen, terwijl dat net de basis is om goed te leren programmeren. GenAI kan zeker helpen, maar dan als ondersteuning en niet als vervanging van het denkproces.
- *Enigszins negatief*: Voor sommige leerlingen is GenAI een verrijking wanneer zij het pas gebruiken als ze vastlopen, maar leerlingen die amper eigen inspanningen leveren en het denkwerk uitbesteden leren er weinig van.
- *Geen effect*: De school stimuleert het gebruik van GenAI voorlopig niet, aangezien nog moet worden uitgewerkt wat wel en niet is toegestaan.
- *Geen effect*: Er worden zowel positieve als negatieve effecten waargenomen.
- *Enigszins positief*: Bij het aanleren van basisvaardigheden zoals conceptueel kritisch nadenken en testen/debuggen kan GenAI het leerproces vertragen, maar zodra deze beheerst worden, helpt het om sneller uitgebreidere software te ontwikkelen; vooral het inzicht dat grondige beheersing van de basis hen effectiever maakt, is daarbij belangrijk.
- *Enigszins positief*: Leerlingen gebruiken GenAI vaak om hun oplossingen te controleren.

Q11. In welke mate bent u bezorgd over de authenticiteit van leerlingenwerk door GenAI? * 

○ Helemaal niet bezorgd $\mapsto$ 1 0 (0%)

○ Eerder niet bezorgd $\mapsto 2$	6 (16%)
○ Neutraal $\mapsto 3$	9 (24%)
○ Eerder bezorgd $\mapsto 4$	16 (43%)
○ Zeer bezorgd $\mapsto 5$	6 (16%)
$n = 37; \bar{x} = 3,59; SD = 0,94; Var = 0,89$	

Toelichtingen:

- *Eerder niet bezorgd*: Het gebruik van GenAI wordt sinds kort gezien als een onderdeel van het ontwerpen van programma's, vergelijkbaar met het gebruik van spellingscontrole in een tekstverwerker.
- *Eerder niet bezorgd*: Leerlingen vallen tijdens schriftelijke toetsen door de mand.
- *Neutraal*: Tijdens toetsen wordt getracht achterpoortjes voor misbruik zoveel mogelijk te sluiten.
- *Neutraal*: Via demo's kan zeer snel en grondig worden gecontroleerd wat leerlingen zelf gerealiseerd hebben en of zij de concepten en vaardigheden effectief beheersen.
- *Neutraal*: Cursisten moeten met GenAI kunnen werken, maar moeten er wel voor zorgen dat zij kwaliteitsvolle output verkrijgen.
- *Neutraal*: Leerlingen die alles met GenAI oplossen, kunnen moeilijkheden ondervinden bij examens. Leerlingen beseffen dat GenAI kan helpen, maar niet door oefeningen steeds te laten oplossen.
- *Eerder bezorgd*: Het is niet altijd duidelijk of leerlingen GenAI gebruikt hebben.
- *Eerder bezorgd*: Hoewel het gebruik van GenAI herkenbaar is, blijft de cognitieve meerwaarde van zelfstandige thuisopdrachten bij gebruik van GenAI een open vraag.
- *Eerder bezorgd*: Opdrachten die leerlingen thuis maken, tellen niet mee voor evaluaties.
- *Eerder bezorgd*: De impact van GenAI hangt sterk af van de intrinsieke motivatie van de leerling; ongemotiveerde leerlingen leren niet echt programmeren wanneer zij GenAI gebruiken, terwijl intrinsiek gemotiveerde leerlingen GenAI ter ondersteuning van oefeningen kunnen gebruiken en hier veel uit kunnen leren.
- *Zeer bezorgd*: Leerlingen denken niet meer na en nemen code over.
- *Zeer bezorgd*: Leerlingen begrijpen vaak niet wat hun code doet en dienen deze zonder nadenken in.

Q12. Heeft u andere bezorgdheden over het gebruik van GenAI in het programmeeronderwijs? $n = 11$

- Het gebruik van GenAI kan programmeeronderwijs in de toekomst doen verdwijnen, door de perceptie dat GenAI programmeertaken zou kunnen overnemen en leerlingen het nut van programmeeronderwijs niet inzien.
- Het gebruik van GenAI vormt een grote uitdaging voor het aanleren van basisprogrammeervaardigheden, doordat leerlingen snel naar GenAI grijpen om oplossingen te genereren zonder enige reflectie, en omdat de relevantie van programmeervaardigheden onder druk komt te staan: "*ChatGPT kan dit toch al, waarom moeten we het dan nog leren?*"
- GenAI kan een waardevol hulpmiddel zijn wanneer leerlingen het gebruiken om uitleg te vragen, ideeën te verkennen of ondersteuning te krijgen wanneer zij vastlopen. Het wordt echter problematisch wanneer gegenereerde code zonder nadenken wordt overgenomen, waardoor zelfstandig denken, fouten

oplossen en begrip van code onvoldoende ontwikkeld worden. Net deze vaardigheden vormen de basis van programmeren.

- Leerlingen denken niet meer zelfstandig na en laten alles over aan GenAI, terwijl het belangrijk blijft zelf inzicht te verwerven in gegenereerde code.
- Leerlingen laten het werk en de eindresultaten door GenAI genereren, waardoor blijvende kennis en inzicht achteruitgaan.
- *Het heeft een afstompend effect op alles en iedereen.*
- Leerlingen beschikken niet over voldoende kennis of vaardigheden met betrekking tot GenAI.
- Leerlingen moeten kritisch leren omgaan met code die door GenAI wordt gegenereerd, wat niet vanzelfsprekend is.
- Zoals bij elk GenAI-gebruik blijft verstandig omgaan met GenAI belangrijk.
- Er wordt getracht het gebruik van GenAI zeer goed te begeleiden.
- Leerkrachten worden onvoldoende grondig opgeleid om snel evoluerende technologieën zoals GenAI aan te leren. Daarnaast krijgen zij hier vaak te weinig tijd voor en ontbreekt in heel wat scholen een degelijk beleid rond dergelijke technologieën.

Kansen en uitdagingen van generatieve AI

Q13. Welke uitdagingen ervaart u bij het gebruik van GenAI in programmeeronderwijs? $n = 32$

- GenAI wordt niet gebruikt tijdens het lesgeven.
- Er worden geen uitdagingen ervaren.
- Er worden weinig uitdagingen ervaren doordat bewust gekozen wordt voor educatieve programmeeromgevingen met beperkte GenAI-functionaliteit en internetgebruik tijdens lessen zoveel mogelijk beperkt wordt, tenzij er specifiek met GenAI gewerkt wordt.
- Het leren gebruiken van GenAI zal een nieuwe basiscompetentie worden.
- GenAI kan een opportuniteit zijn om de kwaliteit sneller te verhogen, mits basisvaardigheden op een degelijke manier aangeleerd blijven.
- Het efficiënt inzetten van GenAI als leermiddel moet aangeleerd worden.
- Het vinden van een ideale combinatie tussen GenAI-gebruik en programmeeronderwijs vormt een uitdaging.
- Leerlingen aanleren hoe zij GenAI op een correcte manier kunnen gebruiken, vormt een uitdaging.
- Het is noodzakelijk leerlingen bewust te maken van de rol en gevolgen van GenAI en hen te leren hier kritisch mee om te gaan.
- Correct gebruik van GenAI stimuleren en hersenloos kopiëren vermijden blijft een uitdaging.
- Het blijft een uitdaging leerlingen te motiveren om programmeertechnieken onder de knie te krijgen.
- Leerlingen laten inzien dat zij zelf nog veel moeten kunnen, vormt een uitdaging.
- Leerlingen denken niet meer zelfstandig na en gaan ervan uit dat GenAI problemen zal oplossen. Daardoor zien zij het nut van programmeren niet meer in.
- Het gebruik van GenAI vormt een self-fulfilling effect waarbij leerlingen het nut van programmeervaardigheden minder inzien.
- Motivatie van leerlingen, opvolging van de authenticiteit van het leerproces en een simplistische visie op programmeren en de meerwaarde daarvan blijven

belangrijke uitdagingen.

- Veel leerlingen gebruiken GenAI om code te laten genereren en nemen deze over zonder echt te begrijpen wat er gebeurt. Daardoor worden zelfstandig nadenken, debuggen en het logisch opbouwen van oplossingen minder geoefend, terwijl net deze vaardigheden belangrijk zijn in programmeren. Daarnaast blijft het moeilijk in te schatten of leerlingen hun werk zelf gemaakt hebben. In de klas kan dit deels opgevolgd worden via Classroom.cloud, maar buiten de les blijft hier weinig zicht op.
- Leerlingen begrijpen zelden de oplossingen die GenAI aanreikt. Zij vertrouwen vanaf de eerste stap op GenAI, waardoor het volledige denkproces wordt overgeslagen.
- Leerlingen lijken gegenereerde code eerder klakkeloos over te nemen dan moeite te doen om deze te begrijpen.
- Leerlingen kopiëren en plakken vaak zomaar code uit GenAI zonder deze te begrijpen. De code bevat daarbij soms instructies die leerlingen nog niet hebben gezien.
- Leerlingen doen onvoldoende moeite om zelf code te genereren en kiezen vaak voor de gemakkelijksoplossing door code door GenAI te laten genereren.
- Studenten kunnen zelf geen code meer genereren en zijn daardoor onvoldoende in staat gegenereerde code te begrijpen, beoordelen of verbeteren.
- Leerlingen hebben moeite om hun programma zelf aan te passen.
- Het blijft een uitdaging na te gaan of leerlingen begrijpen wat zij zien.
- Het kritisch nadenken van leerlingen bij de antwoorden die GenAI geeft, vormt een uitdaging.
- Leerlingen hebben een zekere basiskennis nodig om de resultaten van GenAI te beoordelen.
- Het is moeilijk leerlingen duidelijk te maken waarom inzicht in programmacode belangrijk blijft. Het gevaar bestaat dat leerlingen slagen in hun opleiding zonder zelf inzicht te hebben in programmeren.
- Leerlingen verwerven geen diepgaande kennis meer door het gebruik van GenAI.
- Ondanks lessen over GenAI blijven leerlingen GenAI intensief gebruiken. Leerlingen falen soms voor oefeningen omdat duidelijk zichtbaar is dat oplossingen door GenAI werden gegenereerd.
- Afstandsonderwijs combineren met evaluatie vormt een uitdaging.
- Het beoordelen van cursisten en hun kennis vormt een uitdaging.
- Het controleren en bewaken van de metacognitieve waarde van taken en lessen vormt een uitdaging.
- IT wordt steeds complexer en evolueert steeds sneller. GenAI biedt soms oplossingen aan die voor leerlingen niet meer begripbaar zijn, waardoor sommige leerlingen de houvast verliezen en afhaken.

Q14. Welke kansen ziet u bij het gebruik van GenAI in programmeeronderwijs?

$n = 32$

- In het secundair onderwijs lijken weinig tot geen kansen aanwezig.
- In het secundair onderwijs lijken de kansen beperkt; GenAI lijkt eerder relevant in het hoger onderwijs of later op de werkvloer, nadat leerlingen eerst hebben leren programmeren.
- Bij het aanleren van de basis van programmeren zouden leerlingen idealiter geen GenAI gebruiken tot zij een bepaald niveau bereikt hebben, maar dit blijkt praktisch niet realiseerbaar. Eens dat niveau bereikt is, kan GenAI het leerproces wel versterken.
- In geavanceerdere programmeerlessen kan GenAI zeer nuttig zijn, op voor-

- waarde dat leerlingen eerst een sterke en brede programmeerbasis verwerven.
- Bij een goede kennis van programmeren kan GenAI helpen om snel extra informatie te vinden.
 - Ervaren leerlingen kunnen zeker iets opsteken van het gebruik van GenAI.
 - GenAI biedt kansen als ondersteunend leermiddel. Leerlingen kunnen sneller hulp krijgen, extra uitleg vragen, voorbeelden bekijken wanneer zij vastlopen, verschillende oplossingswegen verkennen en hun werk stapsgewijs verbeteren. Mits correct gebruik kan GenAI het leerproces versterken in plaats van vervangen.
 - GenAI kan als leermiddel worden ingezet, zolang resultaten niet klakkeloos worden overgenomen.
 - GenAI kan nuttig zijn als leermiddel, bijvoorbeeld door een concept op meerdere manieren te laten uitleggen.
 - GenAI is een snel beschikbaar hulpmiddel, maar mag geen voortdurend gemaksmiddel worden.
 - GenAI kan bijdragen aan meer zelfstandigheid bij het leren.
 - Bij correct gebruik kan GenAI helpen bij het zelfstandig verwerken van leerstof.
 - GenAI kan gebruikt worden om leerlingen meer leerondersteuning te geven.
 - GenAI kan dienen als extra hulp bij vragen.
 - GenAI kan leerlingen helpen om leerstof te verwerken na afwezigheid.
 - Leerlingen kunnen buiten de les vragen stellen en concepten opnieuw laten uitleggen in eenvoudigere taal, wat vooral nuttig is voor leerlingen met een andere thuistaal.
 - GenAI kan leerlingen sneller laten groeien doordat het als uitgebreid hulpmiddel beschikbaar is.
 - GenAI kan leerlingen helpen om tot een redenering te komen bij moeilijkere oefeningen en kan ervoor zorgen dat leerlingen extra uitdagende elementen in projecten kunnen uitvoeren.
 - GenAI kan leerlingen zaken en programmeerconcepten tonen die zij eerder nog niet zagen.
 - GenAI kan nuttig zijn als instap in programmeren doordat leerlingen snel met een programmeertaal in contact komen en eenvoudige programma's kunnen schrijven, maar vormt een valkuil wanneer zij dit ook zelfstandig moeten kunnen.
 - Mits een goede aanpak kan GenAI bijdragen aan het opbouwen van een uitgebreidere basiskennis bij lerenden en de drempel tot programmeervaardigheden verlagen, waardoor programmeren ook in meer studierichtingen opgenomen zou kunnen worden.
 - GenAI kan het bereiken van resultaten versnellen, waarbij de aandacht meer verschuift van zelf programmeren naar begrijpen wat een programma doet en het vervolgens debuggen.
 - GenAI kan helpen bij het ontwikkelen van kortere en efficiëntere code.
 - GenAI kan bijdragen aan het schrijven van betere code.
 - GenAI kan ervoor zorgen dat grotere projecten sneller en efficiënter gerealiseerd worden.
 - De kwaliteit van programmeerprojecten en eindwerken kan toenemen doordat cursisten binnen een beperkte tijd betere toepassingen zelf kunnen realiseren, wat kan leiden tot meer voldoening over het bereikte eindresultaat.
 - GenAI kan sneller werken mogelijk maken, leiden tot betere en meer diverse codedemo's en helpen om eigen vaardigheden af te stemmen op moderne ontwikkelingen.
 - GenAI biedt kansen om de kwaliteit van antwoorden op ontwerp vragen te evalueren en via prompts verder te optimaliseren richting een goed ontwerp,

gebaseerd op principes die door de leraar werden aangebracht.

- GenAI kan ondersteuning bieden bij moeilijkheden van leerlingen, foutopsporing en evaluatie, mits verantwoord gebruik.
- GenAI kan soms helpen bij het bedenken van opgaven en het genereren van kant-en-klare databanken die via eigen programma's verwerkt of beheerd moeten worden.
- GenAI kan leerkrachten ondersteunen bij het realiseren van grotere projecten door code te genereren. Leerlingen kunnen GenAI gebruiken om foutieve oefeningen te laten opstellen waarbij zij zelf naar oplossingen moeten zoeken, om beknopte uitleg te krijgen in verschillende talen en om data te genereren voor databanktoepassingen.
- GenAI kan de kwaliteit van lesmateriaal verhogen en de productiviteit van de leraar ondersteunen.

Q15. Hoe verandert GenAI volgens u de rol van de programmeerleerkracht?

$n = 30$

- De impact is momenteel moeilijk in te schatten, aangezien GenAI nog een relatief recent fenomeen is en de toekomst zal moeten uitwijzen hoe dit verder evolueert.
- Geen idee. *I'm getting too old for this shit.*
- GenAI verandert de rol niet.
- De rol verandert weinig, aangezien technieken en concepten nog steeds aangeleerd moeten worden.
- De rol verandert weinig; net zoals de rekenmachine de rol van de wiskundeleraar niet dramatisch veranderde.
- GenAI is een aanvulling.
- De leerkracht blijft expert, maar GenAI kan dienen als extra hulpmiddel.
- GenAI verandert de rol van de programmeerleerkracht in sterk negatieve zin.
- De focus van het programmeeronderwijs verschuift mogelijk van louter programmeren en code schrijven naar meer computationeel denken. Hoewel GenAI een groot deel van de code zal genereren, blijft de verantwoordelijkheid over de code bij de leerling, waardoor ook de rol van de leerling en de leerkracht in het programmeerproces zal veranderen.
- Omdat syntax op de werkvloer minder belangrijk wordt, zullen leerdoelen evolueren naar onderwerpen zoals architectuur en security.
- De leerdoelen moeten worden aangepast.
- De nadruk verschuift meer naar ideeënvorming en programmastructuur, en minder naar het uitvoerig typen van codeblokken; prompting wordt een bijkomend onderdeel.
- Leerkrachten gaan op verschillende manieren om met de impact van GenAI. Een mogelijke aanpak bestaat erin de inhoudelijke focus meer te verschuiven naar een wiskundige benadering, met minder nadruk op programmeren zelf en meer op het herkennen en toepassen van grotere structuren en technieken, zoals lussen, keuzestructuren, variabelen, recursie en numerieke methodes.
- De rol van de leerkracht verschuift meer naar coach en begeleider: minder stap voor stap code uitleggen, en meer leerlingen sturen in hun denkproces en kritisch leren omgaan met GenAI. Het gaat er vooral om leerlingen te leren begrijpen wat zij doen en niet enkel code te laten genereren.
- GenAI verandert de rol van de programmeerleerkracht fundamenteel: de rol wordt niet kleiner, maar inhoudelijk rijker en didactisch complexer, met een evolutie van kennisoverdrager naar leercoach.
- De rol van de leerkracht wordt veel meer sturend.
- De programmeerleerkracht zal meer en meer het realisatieproces moeten

begeleiden, sterker moeten zijn in het aanleren van nieuwe en bijkomende technologieën en vaker moeten lesgeven in projectvormen.

- De leerkracht wordt meer een coach die leerlingen helpt bij het oplossen van fouten in code, omdat GenAI niet alles oplost en veel afhangt van de gebruikte prompt.
- De rol verandert sterk: er worden minder vragen gesteld, de leerkracht moet aandachtig opvolgen hoe leerlingen GenAI gebruiken, meer individuele leergesprekken voeren, meer differentiëren en anders evalueren. De leerkracht neemt daarbij steeds meer een mentorrol op.
- De programmeerleerkracht krijgt meer een controlerende rol.
- Er wordt soms meer een rol als bewaker dan als lesgever ervaren en er wordt gezocht naar manieren om leerlingen offline te leren programmeren.
- Leerlingen kritisch leren omgaan met gegenereerde code wordt belangrijker; ook het testen van code krijgt een grotere rol.
- Leerkrachten moeten leerlingen aanleren om niet alles zomaar over te nemen.
- Het klassieke lesgeven en de praktijk veranderen: leerlingen mogen hulpmiddelen gebruiken, maar moeten ermee leren omgaan en een kritische houding aannemen, wat zij vaak vergeten.
- Toetsen en examens verlopen zonder GenAI; leerlingen moeten per probleemstelling eerst een beredeneringsschema opstellen voor zij code noteren.
- GenAI maakt het gemakkelijker om authentieke oefeningen te ontwikkelen, met telkens nieuw en uniek materiaal.
- GenAI levert bij de voorbereiding veel tijdswinst op, maar in de klas blijft de menselijke uitleg bij oefeningen belangrijk.
- GenAI kan gebruikt worden om ondersteunende tools te ontwikkelen.
- GenAI kan een goede aanvulling en hulp zijn bij het oplossen van problemen van leerlingen, bijvoorbeeld doordat leerstof soms op een andere manier wordt uitgelegd.
- GenAI is een goede bron van inspiratie, maar blijkt minder nodig voor technische ondersteuning bij ruime programmeerervaring.

Toelating van generatieve AI

Q16. Laat u het gebruik van GenAI toe tijdens de programmeerlessen? * 

☐ Nooit $\mapsto$ 1	8 (22%)
☐ Zelden $\mapsto$ 2	8 (22%)
☐ Af en toe $\mapsto$ 3	13 (35%)
☐ Regelmatig $\mapsto$ 4	4 (11%)
☐ Zeer vaak $\mapsto$ 5	4 (11%)

$n = 37$; $\bar{x} = 2,68$; $SD = 1,23$; $Var = 1,52$

Toelichtingen:

- *Nooit*: De programmeerlessen richten zich op het aanleren van de basis; met slechts één lesuur per week wordt het minimumniveau waarbij GenAI nodig zou zijn, niet bereikt.
- *Zelden*: Leerlingen moeten zelfstandig leren nadenken over mogelijke oplossingen en mogen niet volledig afhankelijk worden van GenAI.
- *Nooit*: Net zoals leerlingen eerst leren hoofdrekenen en cijferen voor zij een rekenmachine gebruiken, wordt ook GenAI niet van bij het begin ingezet.
- *Zelden*: Leerlingen moeten eerst zelfstandig leren programmeren alvorens hulpmiddelen zoals GenAI te gebruiken.

- *Zelden*: GenAI wordt enkel gebruikt om tijd te winnen bij het begeleiden van individuele leerlingen, zodat lessen vlotter kunnen verlopen, of bij configuraties van tools zoals VS Code, Python, GitHub of libraries die issues kunnen vertonen.
- *Af en toe*: Als leerkracht is het niet te vermijden dat leerlingen GenAI gebruiken.
- *Af en toe*: Leerlingen mogen gebruikmaken van GenAI, enerzijds om inzicht te krijgen in hoe zij hiermee omgaan en anderzijds om hen te ondersteunen in het doordacht en kritisch leren gebruiken van GenAI.
- *Af en toe*: Er worden afspraken gemaakt over het gebruik van GenAI. In het vijfde jaar wordt het minder gebruikt, terwijl het in het zesde jaar vaker ingezet wordt om websites een modernere lay-out te geven met frameworks zoals Bootstrap of Tailwind.
- *Af en toe*: Het gebruik van GenAI hangt af van de opdracht. In het vijfde jaar wordt gewerkt met specifieke gedeelde prompts om leerlingen redeneringen te laten opbouwen zonder kant-en-klare oplossingen te verkrijgen. In het zesde jaar kan GenAI bij projecten wel gebruikt worden, omdat leerlingen daar complexere code moeten schrijven en gegenereerde code nog steeds zelf moeten interpreteren en aanpassen.
- *Af en toe*: Leerlingen mogen GenAI gebruiken om hun oefeningen zelf te laten controleren en verbeterde uitwerkingen te verkrijgen.
- *Regelmatig*: Leerlingen mogen tijdens de les verschillende bronnen raadplegen, zoals de leerkracht, leerstof, fora en GenAI. Daarbij weten zij dat begrip belangrijker is dan het louter bekomen van het juiste antwoord.
- *Regelmatig*: Er is berusting in het gebruik van GenAI: *Ik heb het opgegeven*. Programmeercode staat niet centraal binnen de lessen.
- *Zeer vaak*: GenAI is onderdeel van het leren programmeren zelf.
- *Zeer vaak*: Het gebruik van GenAI wordt toegestaan, op voorwaarde dat leerlingen expliciet aangeven wat zij niet zelfstandig konden oplossen.

Q17. Laat u het gebruik van GenAI toe tijdens toetsen of examens? * 

☐ Nooit $\mapsto 1$	27	(73%)
☐ Zelden $\mapsto 2$	2	(5%)
☐ Af en toe $\mapsto 3$	1	(3%)
☐ Regelmatig $\mapsto 4$	5	(14%)
☐ Zeer vaak $\mapsto 5$	2	(5%)

$n = 37$; $\bar{x} = 1,73$; $SD = 1,31$; $Var = 1,71$

Toelichtingen:

- *Nooit*: Dat heeft geen zin, omdat in deze fase van het leerproces nog gewerkt wordt aan basislogica, algoritmes en syntax.
- *Nooit*: Evaluaties dienen om na te gaan of leerlingen zelf de opgaven kunnen oplossen en niet of GenAI dit kan.
- *Nooit*: Het gebruik van GenAI is niet toegestaan tijdens toetsen en examens, omdat deze evaluatiemomenten bedoeld zijn om na te gaan in welke mate leerlingen de leerstof zelfstandig beheersen en of zij tijdens lessen en opdrachten op een doordachte manier met GenAI zijn omgegaan.
- *Nooit*: Tijdens evaluatiemomenten is het belangrijk dat leerlingen hun eigen capaciteiten tonen. Bovendien kan het gebruik van GenAI de objectiviteit van evaluaties ondermijnen, omdat identieke prompts verschillende antwoorden kunnen opleveren.
- *Zelden*: GenAI wordt soms toegestaan tijdens toetsen, maar er worden bijkomende elementen toegevoegd zodat leerlingen ook verplicht worden

zelfstandig na te denken. Examens worden op papier afgelegd.

- *Zelden*: Bepaalde websites kunnen tijdens evaluaties uitgeschakeld worden, al blijft GenAI moeilijk volledig te weren, bijvoorbeeld via tools zoals Copilot.
- *Regelmatig*: Evaluaties bevatten een offline gedeelte op papier waarbij GenAI niet gebruikt mag worden.
- *Zeer vaak*: Het gebruik van GenAI maakt deel uit van het programmeerproces.
- *Zeer vaak*: Het gebruik van GenAI vormt geen struikelblok, op voorwaarde dat leerlingen mondeling hun code functioneel en technisch verdedigen en kleine wijzigingen op verzoek van de leerkracht kunnen uitvoeren.

Q18. Laat u het gebruik van GenAI toe bij programmeeropdrachten buiten de klas? * 

☐ Niet van toepassing $\mapsto$ niet meegerekend	— (—%)
☐ Nooit $\mapsto$ 1	4 (15%)
☐ Zelden $\mapsto$ 2	3 (11%)
☐ Af en toe $\mapsto$ 3	8 (30%)
☐ Regelmatig $\mapsto$ 4	6 (22%)
☐ Zeer vaak $\mapsto$ 5	6 (22%)

$n = 27$; $\bar{x} = 3,26$; $SD = 1,32$; $Var = 1,75$

Toelichtingen:

- *Niet van toepassing*: Er worden geen oefeningen thuis voorbereid.
- *Niet van toepassing*: Het is niet controleerbaar.
- *Niet van toepassing*: Het gebruik van GenAI buiten de klas valt niet te controleren en er kan weinig tegen ingebracht worden.
- *Niet van toepassing*: Leerlingen worden gevraagd GenAI op een correcte manier te gebruiken, al is daar geen controle op.
- *Nooit*: De programmeerlessen richten zich op het aanleren van de basis; met slechts één lesuur per week wordt het minimumniveau waarbij GenAI nodig zou zijn, niet bereikt.
- *Nooit*: Er is wel besef dat leerlingen GenAI gebruiken.
- *Zelden*: Leerlingen werken slechts beperkt aan programmeeropdrachten buiten de lessen.
- *Af en toe*: Daar heeft de leerkracht geen controle op.
- *Af en toe*: Het gebruik wordt sterk afgeraden, maar de leerkracht heeft daar geen vat op en kan het nooit volledig verbieden. GenAI is er nu eenmaal.
- *Regelmatig*: Het gebruik van GenAI buiten de klas is onmogelijk te vermijden: *If you can't beat them, join them*. Leerlingen worden er wel op gewezen GenAI enkel te gebruiken ter ondersteuning van het leerproces en niet om competenties voor formele evaluaties te veinzen. Heimelijk gebruik van GenAI wordt bestraft.
- *Zeer vaak*: Omdat volledig online gewerkt wordt en leerlingen van thuis uit werken, wordt het gebruik van GenAI geïntegreerd en omarmd. Een verbod is zinloos, aangezien het gebruik toch niet controleerbaar is.
- *Zeer vaak*: Bij opdrachten buiten de klas is er geen zicht op.
- *Zeer vaak*: Buiten de les heeft de leerkracht er geen vat op.

Aanpassingen aan het programmeeronderwijs

Q19. Heeft u uw manier van lesgeven al aangepast om rekening te houden met GenAI? *

- | | | |
|--|----|-------|
| ☐ Ja, altijd | 9 | (24%) |
| ☐ Ja, soms | 13 | (35%) |
| ☐ Nee, maar ik overweeg het | 9 | (24%) |
| ☐ Nee, en ik overweeg het niet | 6 | (16%) |

$n = 37$

Q20. Op welke manier(en) heeft u uw manier van lesgeven al aangepast om rekening te houden met GenAI? $n = 20$

- De volledige cursusinhoud werd onlangs herschreven met het oog op de integratie van GenAI.
- De lessen werden aangepast door GenAI bewust te integreren in oefeningen en projecten. Leerlingen mogen code genereren met GenAI, maar moeten deze nadien uitleggen, analyseren en verbeteren. De focus verschuift van het schrijven van code naar begrip en probleemoplossend denken. Tijdens toetsen en examens is GenAI niet toegestaan, zodat de individuele kennis en vaardigheden van leerlingen geëvalueerd kunnen worden.
- GenAI wordt geïmplementeerd in de lessen, waarbij leerlingen leren hoe ermee om te gaan en antwoorden kritisch te analyseren, terwijl het denkproces levendig gehouden wordt.
- Tijdens oefenmomenten wordt extra begeleiding voorzien voor het correct gebruik van GenAI.
- Leerlingen krijgen specifieke prompts aangereikt die zij in extreme gevallen kunnen gebruiken om extra uitleg bij oefeningen te krijgen zonder de oplossing te tonen.
- Leerlingen worden soms verplicht GenAI te gebruiken om bepaalde code te begrijpen, waarna hierover vragen gesteld worden om na te gaan of zij de uitleg van GenAI effectief begrijpen.
- Leerlingen worden aangemoedigd GenAI te gebruiken om sneller problemen op te lossen.
- Er wordt gewerkt met individuele opvolging, differentiatie en uitdagende programmeeroefeningen.
- In STEM-lessen wordt bijvoorbeeld code gegenereerd en gebruikt via een Raspberry Pi, zonder dat leerlingen die code zelf hoeven te schrijven.
- Vragen worden zo geformuleerd dat duidelijk zichtbaar is of GenAI gebruikt werd, en er wordt expliciet aangegeven wanneer leerlingen GenAI wel of niet mogen gebruiken.
- De schermen van leerlingen worden meer opgevolgd. Tijdens de eerste lessen werd klassikaal code besproken van leerlingen die wel of geen GenAI gebruikten. Daarnaast wordt meer samengewerkt in groepjes zodat leerlingen eerder raad vragen aan elkaar dan aan GenAI.
- Er wordt veel vaker gewerkt met onder meer mondelinge klassikale vragen, het overlopen en aanvullen van code, denkwijzen en analyse.
- Er worden uitgebreidere uitdagingen aangeboden en klassieke toetsen worden vaker vervangen door demo's en mondelinge verdedigingen.
- De lat wordt hoger gelegd en vragen worden uitgebreider, waardoor GenAI vaker het overzicht verliest.

- Veel zaken, zoals toetsen en examens, verlopen op papier zonder gebruik van laptops.
- Evaluaties verlopen op papier en enkel in de klas, terwijl het leerproces gedeeltelijk thuis plaatsvindt.
- Er worden geen voorbereidingen meer buiten de klas gemaakt.
- Thuis oefenen wordt niet langer gequoteerd, waardoor leerlingen daar zelf verantwoordelijkheid voor moeten opnemen. *Oefening baart kunst, dat weten ze ook.* Grote projecten waaraan thuis gewerkt wordt, moeten in de klas gepresenteerd worden. Leerlingen moeten daarbij hun code regel per regel kunnen uitleggen en gerichte vragen over hun aanpak beantwoorden.
- Websites kunnen sneller en moderner opgemaakt worden.
- GenAI wordt gebruikt voor het genereren van slides.

Q21. Heeft u uw programmeeropdrachten al aangepast om rekening te houden met GenAI? *

- | | | |
|--|----|-------|
| ☐ Ja, altijd | 4 | (11%) |
| ☐ Ja, soms | 13 | (35%) |
| ☐ Nee, maar ik overweeg het | 13 | (35%) |
| ☐ Nee, en ik overweeg het niet | 7 | (19%) |

$n = 37$

Q22. Op welke manier(en) heeft u uw programmeeropdrachten al aangepast om rekening te houden met GenAI? $n = 15$

- Er zijn meer programmeeropdrachten waarbij de focus ligt op computationeel denken. Waar vroeger werkende code vaak voldoende was, wordt nu meer nadruk gelegd op geoptimaliseerde code. GenAI levert daarbij de werkende code aan, terwijl de cursist verantwoordelijk blijft voor de optimalisatie en correcte integratie ervan binnen de programmeeropdrachten.
- Programmeeropdrachten werden aangepast zodat leerlingen GenAI mogen gebruiken, maar hun code ook moeten kunnen uitleggen, analyseren en verbeteren. Daarnaast worden meer vragen gesteld die focussen op denkprocessen en begrip in plaats van enkel op het eindresultaat, zodat leerlingen niet zomaar code kopiëren zonder inzicht.
- Bij programmeeropdrachten wordt op voorhand rekening gehouden met het mogelijke gebruik van GenAI door leerlingen.
- Leerlingen moeten GenAI leren gebruiken; sommige programmeeropdrachten focussen expliciet op dat gebruik, bijvoorbeeld via agents.
- Leerlingen werken opdrachten eerst zonder GenAI uit en verbeteren deze nadien met behulp van GenAI.
- Leerlingen moeten zelf interpretaties geven bij hun oplossingen.
- Leerlingen moeten soms eerst hun structuur in pseudocode op papier uitwerken.
- GenAI kan soms helpen om fouten gemakkelijker terug te vinden en deze in begrijpelijke taal uit te leggen.
- Bij het aanleren wordt gewerkt met live demo's, kleine oefeningen en grotere projecten die over meerdere weken lopen en uitdagender zijn dan vroeger.
- Programmeeropdrachten worden uitdagender en meer afgestemd op de leefwereld van leerlingen.
- Er wordt gewerkt met grote projecten waarbij ook analyse vereist is.
- In het zesde jaar werken leerlingen aan een groter project waarbij GenAI bijvoorbeeld gebruikt kan worden om voorbeelddata voor databanken te ge-

nereren. In dergelijke projecten kunnen ook complexere vereisten opgenomen worden.

- Programmeeropdrachten worden vaker in groep uitgevoerd, met grote projecten waarin verschillende deelfuncties voorkomen en waarbij bijvoorbeeld structuur en namen van variabelen op voorhand afgesproken worden.
- Leerlingen werken samen tijdens oefeningen.
- Programmeeropdrachten worden enkel in de klas uitgevoerd.

Evaluatie en toekomstige vaardigheden

Q23. Op welke manier evalueert u leerlingen die GenAI gebruiken bij programmeeropdrachten? *

- ☐ Ik beoordeel hun werk op dezelfde manier als bij andere leerlingen 6 (16%)
- ☐ Ik pas mijn beoordeling aan wanneer AI is gebruikt (bijvoorbeeld strengere criteria, extra controle, nadruk op begrip) 11 (30%)
- ☐ Ik vraag een mondelinge toelichting of een code walkthrough om hun inzicht te controleren 20 (54%)
- ☐ * Andere 10 (27%)

Andere¹⁴: GenAI is niet toegelaten tijdens evaluaties (3); enkel werk zonder GenAI wordt geëvalueerd (2); mogelijk mondelinge toelichting; afhankelijk van opdracht. $n = 37$

Q24. Welke strategieën acht u meest effectief om opdrachten ‘AI-proof’ te maken? ✂

- ☐ Mondelinge toelichting of code-walkthroughs 30 (81%)
- ☐ Meer reflectievragen 24 (65%)
- ☐ Meer nadruk op proces (bv. versiegeschiedenis) 16 (43%)
- ☐ Documenteren van AI-gebruik 12 (32%)
- ☐ Samenwerkend leren (pair programming) 10 (27%)
- ☐ Meer open of contextuele opdrachten 10 (27%)
- ☐ ☞ Ik zie geen mogelijkheden om opdrachten ‘AI-proof’ te maken 1 (3%)
- ☐ ☞ Ik zie geen nood om opdrachten ‘AI-proof’ te maken ... 3 (8%)
- ☐ * Andere 3 (8%)

Andere: opdrachten op papier; educatieve omgevingen gebruiken en internet afsluiten; gepersonaliseerde data gebruiken en opdrachtspecifieke coderichtlijnen geven. $n = 37$

¹⁴Antwoorden zoals ‘/’ werden buiten beschouwing gelaten.

Q25. Welke vaardigheden moeten leerlingen volgens u méér ontwikkelen nu GenAI beschikbaar is? * 23

- | | | |
|--|----|-------|
| ☐ Probleemanalyse (programmeervraagstukken begrijpen en ontleden vóór het programmeren) | 33 | (89%) |
| ☐ Code begrijpen en beoordelen | 33 | (89%) |
| ☐ Debugging | 27 | (73%) |
| ☐ Documenteren en redeneren over oplossingen | 23 | (62%) |
| ☐ Prompt engineering (effectieve opdrachten en instructies formuleren voor GenAI-tools) | 22 | (59%) |
| ☐ ☞ Ik zie geen nood om bepaalde vaardigheden extra te benadrukken | 0 | (0%) |
| ☐ * Andere | 2 | (5%) |

Andere: code beheren in plaats van enkel code schrijven; *alles toch?*

n = 37

Professionalisering

Q26. Voelt u zich voldoende voorbereid om GenAI zinvol te integreren in het programmeeronderwijs? *

- | | | |
|--|----|-------|
| ☐ ☞ Ja | 16 | (43%) |
| ☐ Nee, ik heb training nodig | 13 | (35%) |
| ☐ Nee, ik heb richtlijnen nodig | 10 | (27%) |
| ☐ Nee, ik heb ondersteuning vanuit de school/instelling nodig | 4 | (11%) |

n = 37

Toelichtingen:

- *Ja*: We kunnen niet anders, we zijn wederom pioniers.
- *Ja*: Ik ben erg zelfredzaam.
- *Ja*: Ik volg een postgraduaat databeheer.
- *Ja*: Er is hieromtrent een omkadering voorzien op school.
- *Ja*: Ik weet waar ik naartoe wil, maar moet het nog uitvoeren.
- *Ja*: Ik sta natuurlijk open voor verbetering.
- *Ja*: Extra zinvolle informatie en/of training kunnen zeker een meerwaarde zijn.
- *Nee, ik heb training nodig*: Wat ik ken, is gebaseerd op eigen initiatief; er zou structureel tijd voorzien moeten worden om dit aan te leren en expertise te delen met collega's.
- *Nee, ik heb training nodig*: Ik heb wel grotendeels een idee, maar de technologie verandert zo snel en leerlingen vinden ook heel snel de achterpoortjes.

Slot

Q27. Heeft u nog opmerkingen of ervaringen over GenAI in programmeeronderwijs die u wilt delen?¹⁵ $n = 11$

- GenAI biedt veel kansen binnen programmeeronderwijs, maar vereist ook een duidelijke didactische aanpak. Leerlingen zijn snel geneigd code te kopiëren zonder inzicht, waardoor begeleiding in kritisch gebruik essentieel is.
- Er moet over gewaakt worden dat leerlingen GenAI niet als drijvende kracht gebruiken, maar als ondersteuning.
- Net zoals bij wiskunde een som voldoende begrepen moet worden vóór er op een rekenmachine getokkeld wordt, is dat bij programmeren niet anders. Als leerlingen een zinvol algoritmisch bewustzijn moeten ontwikkelen, blijft die basis onvermijdelijk. Het motiveren van die aanpak vormt momenteel een grote uitdaging, vooral bij leerlingen, waarvan velen liever een deel van het denkwerk aan GenAI uitbesteden.

Q28. Zijn er nog andere opmerkingen of suggesties (over deze vragenlijst of het onderzoek) die u graag wilt meegeven?¹⁶ $n = 10$

- De vragenlijst is duidelijk en relevant voor de huidige evoluties binnen het onderwijs.
- Aanpakstrategieën voor programmeerlessen zouden aangeboden kunnen worden via opleidingen en nascholingen.

Q29. Laat uw emailadres achter indien ik u mag contacteren voor opvolging of om de resultaten te delen. $n = 23$

¹⁵Antwoorden zoals ‘nee’ of ‘/’ werden buiten beschouwing gelaten.

¹⁶Idem.

C. Verklaring omtrent het gebruik van generatieve AI

Bij het opstellen van deze masterproef werd generatieve AI op verschillende manieren als ondersteunend hulpmiddel gebruikt. Overeenkomstig het richtkader van de Universiteit Gent voor verantwoord AI-gebruik wordt hieronder transparant toegelicht waarvoor en op welke wijze deze technologie tijdens het onderzoeks- en schrijfproces werd ingezet.

ChatGPT werd gebruikt als sparringpartner bij het uitwerken en verfijnen van ideeën, het beoordelen van tekststructuren en het voorstellen van alternatieve verwoordingen. Ik schreef steeds zelf een initiële versie van de tekst, die vervolgens met behulp van ChatGPT werd nagelezen en iteratief verfijnd. ChatGPT werd daarbij gebruikt om mogelijke onduidelijkheden, stijlproblemen en structurele verbeterpunten te identificeren. Ook werden alternatieve formuleringen overwogen wanneer deze de helderheid, academische toon of samenhang van de tekst konden verbeteren. Alle inhoudelijke keuzes, argumenten, interpretaties en nuanceringen werden door mij bepaald, gecontroleerd en goedgekeurd.

Voor het literatuuronderzoek werd ChatGPT sporadisch ingezet om recente en relevante wetenschappelijke bronnen en bijhorende referenties op te sporen. Deze bronnen werden vervolgens zelfstandig opgezocht, geverifieerd en geraadpleegd via academische databanken alvorens ze eventueel in deze masterproef werden opgenomen.

Tot slot werd ChatGPT gebruikt bij het ontwikkelen en verfijnen van de vragenlijst, onder meer voor het formuleren, herstructureren en kritisch evalueren van vragen. Alle methodologische keuzes, waaronder de inhoud, opbouw en uiteindelijke selectie van de vragen, werden zelfstandig beoordeeld en gevalideerd.