VX COMPANY
CONSULTANCY SERVICES

VX AI LAB

WHITEPAPER

DE IMPACT VAN AI
OP SCRUM, AGILE EN KANBAN

Hoe AI-agents de rolverdeling, flow en besluitvorming in teams fundamenteel veranderen

---

versie 1.0 | april 2026 | M. Hoppen
vxcompany.com

INLEIDING

De opkomst van AI-agents verandert softwareontwikkeling in een fundamenteel ander speelveld. Waar AI tot voor kort vooral een persoonlijke productiviteitstool was (denk aan code-suggesties via GitHub Copilot of het genereren van documentatie met ChatGPT) verschuift de technologie nu richting autonome systemen die zelfstandig taken uitvoeren, beslissingen nemen en met elkaar samenwerken.

Die verschuiving raakt niet alleen de techniek. Ze raakt de manier waarop teams werken, hoe rollen zijn verdeeld en waar de bottlenecks zitten in het ontwikkelproces. En daarmee raakt ze de kern van Scrum, Kanban en Agile.

Dit whitepaper onderzoekt wat AI-agents betekenen voor de dagelijkse praktijk van Agile teams. Niet als theoretisch gedachte-experiment, maar als een concrete analyse van verschuivingen die nu al zichtbaar worden. We bouwen voort op de richtlijnen uit ons eerdere whitepaper "Verantwoord gebruik van AI-tools bij de klant" en verleggen de focus: van individueel toolgebruik naar de systemische impact op teams en processen.

Voor wie is dit whitepaper?

Dit document is geschreven voor Scrum Masters, Product Owners, Agile coaches, ontwikkelteams en managers die willen begrijpen hoe AI-agents hun werkwijze gaan beïnvloeden en hoe ze daar nu al op kunnen anticiperen.

VAN COPILOT NAAR ORKESTRATOR: DE VERSCHUIVENDE ROL VAN DE DEVELOPER

De evolutie van de developer-rol volgt een duidelijk patroon dat zich in steeds kortere cycli ontvouwt:

FASE 1	De developer schrijft zelf code: het traditionele ambacht van regel-voor-regel programmeren.
FASE 2	De developer gebruikt AI als assistent: tools als Copilot suggereren code, de developer beslist.
FASE 3	De developer stuurt een team van agents aan: één schrijft code, één genereert tests, één reviewt, één documenteert.

In fase 3 wordt de developer een regisseur en systeemdenker. De operationele uitvoering wordt in toenemende mate gedelegeerd aan agents. Dit levert enorme productiviteitswinst op: agents werken parallel, worden niet moe en leveren consistent output.

Maar er is een keerzijde. Met delegatie aan agents vermindert de directe grip op details. De developer moet vertrouwen op output die niet altijd transparant tot stand is gekomen. En er ontstaat een nieuwe afhankelijkheid: niet van collega's, maar van tooling.

Spanning in de praktijk

Enerzijds biedt agent-based development een enorme productiviteitsboost met parallel werk en minder handmatig werk. Anderzijds ontstaat minder grip op details, afhankelijkheid van tooling en wordt het moeilijker te begrijpen "wat er echt gebeurt."

VAN LINEAIR PROCES NAAR PARALLEL SYSTEEM

Traditionele softwareontwikkeling volgt, ondanks Agile principes, in de praktijk vaak een redelijk voorspelbaar pad: analyse, design, build, test, deploy. Stappen worden weliswaar iteratief doorlopen, maar overwegend sequentieel.

AI-agents doorbreken dat patroon. Ze kunnen tegelijkertijd experimenteren met meerdere oplossingen, alternatieven genereren en zichzelf corrigeren. Het ontwikkelproces wordt daarmee eerder een complex adaptief systeem dan een geordende pipeline.

En juist dát sluit aan op de kern van Agile en Lean denken. Complexe adaptieve systemen vragen om empirische procescontrole — inspectie, adaptatie en transparantie — precies de pijlers onder Scrum. Het flow-denken uit Kanban wordt relevanter dan ooit, juist omdat de hoeveelheid gelijktijdig werk explosief toeneemt.

DE NIEUWE BOTTLENECK: VAN BOUWCAPACITEIT NAAR BESLISCAPACITEIT

Jarenlang was development-capaciteit de voornaamste bottleneck in softwareprojecten. Te weinig developers, te veel werk, te lange sprints. In volwassen Agile teams zagen we de bottleneck al verschuiven richting gebruikers die de opgeleverde functionaliteiten niet meer konden verwerken. Maar AI-agents verschuiven die bottleneck fundamenteel.

Wanneer bouwen goedkoop en snel wordt, wordt de schaarse resource niet langer "wie schrijft de code?" maar "wie neemt de besluiten?" Hiermee wordt een tipping point bereikt, waardoor de bottleneck voorgoed verschuift van het team naar de Product Owner, de stakeholders en de gebruikers.

Drie nieuwe bottlenecks

- **Besluitvorming:** Agents kunnen razendsnel vijf UI-varianten, tien algoritmes of drie architecturen genereren. Maar iemand moet kiezen wat waardevol is. Dat vergt focus, domeinkennis en de moed om opties te laten vallen.
- **Validatie:** De hoeveelheid output stijgt, maar de capaciteit om die output te beoordelen blijft menselijk begrensd. De cognitieve load op Product Owners en reviewers neemt fors toe.
- **Acceptatie:** Meer features, meer varianten, meer data… maar de organisatie en eindgebruiker kunnen slechts beperkt absorberen. Het risico op overproductie van opties is reëel.

Lean-perspectief

In Lean-termen ontstaat een nieuw type verspilling: overproductie van opties. Vergelijkbaar met te veel WIP of te veel features, maar nu te veel mogelijke oplossingen. Zonder expliciete flow-sturing krijg je meer output, maar slechtere flow.

ARCHITECTUUR WORDT GENADELOOS

Wanneer meerdere agents tegelijkertijd aan een codebase werken, wordt de kwaliteit van de architectuur genadeloos zichtbaar. Slechte architectuur, onduidelijke boundaries, ontbrekende API-contracts, gedupliceerde logica, leidt tot inconsistenties die agents niet zelfstandig kunnen oplossen.

Dit maakt Domain-Driven Design (DDD) relevanter dan ooit. Scherpe bounded contexts, heldere domeinmodellen en expliciete contracts zijn niet langer “nice to have” maar een harde voorwaarde voor effectief werken met agents. Zonder die kaders schalen we weliswaar de output van agents, maar schalen we ook de problemen alsook het gebrek in kwaliteit.

- **Boundaries:** Definieer heldere grenzen tussen domeinen en services zodat agents binnen een afgebakend werkgebied opereren.
- **Contracts:** Zorg voor expliciete API-specificaties zodat de output van de ene agent past op de input van de andere.
- **Domeinmodellen:** Investeer in een gedeeld begrip van het domein. Agents die op basis van verschillende aannames werken, versterken elkaars fouten.

KWALITEIT VERSCHUIFT: VAN OUTPUT NAAR SYSTEEM

In een wereld waarin agents code produceren, verandert de aard van kwaliteitsborging. Het beoordelen van individuele code is nog steeds belangrijk, maar niet langer voldoende. De vraag wordt: hoe werken agents samen? Hoe komen beslissingen tot stand? Hoe functioneren de feedback loops?

QA verschuift daarmee van het testen van features naar het valideren van systeemgedrag. Niet “doet deze functie wat hij moet doen?” maar “gedraagt het samenspel van agents, code en data zich zoals verwacht?”

De valkuil van schijnbare autonomie

Een belangrijk risico verdient extra aandacht: agents lijken zelfstandig, maar ze missen echte context. Ze optimaliseren lokaal, niet systeembreed. En ze kunnen elkaars slechte aannames bevestigen: een feedbackloop die zonder menselijk toezicht snel escaleert.

Zonder goede kaders krijg je snelle output, maar ook snelle rommel. De parallel met “hallucineren” bij taalmodellen is treffend: het systeem produceert met grote zekerheid output die inhoudelijk onjuist kan zijn.

CONCRETE IMPACT OP SCRUM EN KANBAN

Impact op Scrum-rollen

De Product Owner

De Product Owner wordt de cruciale schakel in een agent-gedreven wereld. Als bouwcapaciteit vrijwel onbeperkt schaalbaar wordt, verschuift de verantwoordelijkheid naar het scherp stellen van de juiste vragen: wat bouwen we, waarom en is dit goed genoeg? De PO moet een sterke productvisie vertalen naar duidelijke kaders waarbinnen agents en teams effectief kunnen werken. Zonder die scherpte dreigt keuzestress of oppervlakkige besluitvorming.

De Scrum Master

De Scrum Master krijgt een uitgebreide taak. Naast het faciliteren van het team en het bewaken van het Scrum-proces, moet de Scrum Master nu ook sturen op de interactie tussen menselijke teamleden en AI-agents. De kernvraag verschuift van “hoe werkt het team samen?” naar “hoe functioneert het systeem van mensen én agents?” Flow-management, WIP-limieten en het signaleren van nieuwe bottlenecks worden nog belangrijker.

Het Development Team

Teams worden mogelijk kleiner in menselijke bezetting, aangevuld met "virtuele teams" van agents. De vaardigheden die dominant worden zijn systeemdenken, probleemdefinitie, architectuur en kritisch beoordelen. Pure syntaxkennis en boilerplate coding worden minder onderscheidend.

Impact op Scrum Events

SPRINT PLANNING	Meer nadruk op het formuleren van duidelijke doelen en acceptatiecriteria, minder op het inschatten van bouwtijd. De planning wordt een oefening in precisie van opdrachten aan agents.
DAILY SCRUM	Focus verschuift van "wat heb ik gedaan?" naar "wat hebben de agents opgeleverd en welke beslissingen zijn nodig?" Het team moet sneller impediments rond agent-output signaleren.
SPRINT REVIEW	De hoeveelheid opgeleverd werk neemt potentieel sterk toe. De Review moet scherper filteren op waarde in plaats van volume. Stakeholders moeten leren kiezen uit overvloed.
RETROSPECTIVE	Nieuwe reflectievragen: werken de agents effectief? Zijn de prompts scherp genoeg? Verliezen we als team het overzicht? De Retro wordt het moment om de mens-agent samenwerking te verbeteren.

Impact op Kanban en flow

Vanuit Kanban-perspectief sluit de agent-revolutie naadloos aan op flow-denken, maar brengt het ook nieuwe uitdagingen. Agents zijn in essentie extra capaciteit. Maar flow wordt nog steeds bepaald door WIP, afhankelijkheden en bottlenecks.

- **WIP-limieten:** Moeten niet alleen gelden voor menselijk werk, maar ook voor het aantal gelijktijdige agent-opdrachten. Zonder deze begrenzing ontstaat overproductie.
- **Expliciete keuzeprocessen:** Wanneer agents meerdere varianten genereren, is een expliciet selectiemoment nodig. Voeg een "triage"- of "select"-kolom toe aan het board.
- **Feedback loops verkorten:** De snelheid van agents vereist snellere validatie. Wacht niet op een Sprint Review om agent-output te beoordelen.

VERSCHUIVING NAAR KORTERE FEEDBACKLOOPS

De toenemende snelheid van agent-based development dwingt teams om hun ritme van afstemming te herzien. Waar Scrum traditioneel werkt met vaste ceremonies op sprint-niveau, ontstaat er druk om feedbackloops op alle niveaus te verkorten.

Op het niveau van het development team is de verwachting dat teamleden meerdere afstemmomentjes per dag nodig hebben met hun team van agents, een soort Hourly Scrum. Niet als formele ceremonie, maar als lichtgewicht check: wat hebben de agents opgeleverd, welke beslissingen zijn nodig en waar liggen blokkades?

Op Scrum Master-niveau verandert de Daily Scrum in essentie in een Scrum of Scrums. Wanneer meerdere agent-teams parallel werken, ontstaat een Team of Teams-dynamiek die vraagt om expliciete coördinatie tussen de verschillende stromen van werk.

Op het niveau van de Product Owner en stakeholders zal ook de Review frequentie moeten toenemen. Wanneer agents dagelijks substantiële output leveren, is een tweewekelijkse Sprint Review onvoldoende om tijdig bij te sturen. Dagelijkse of ad-hoc reviewmomenten worden de norm.

Van structureel naar adaptief

Deze verschuiving naar kortere feedbackloops heeft een belangrijk gevolg: ceremonies die ooit structureel en planbaar waren, worden steeds vaker ad-hoc en contextgedreven. Op het moment dat dat het geval is, biedt Kanban mogelijk een betere invulling dan Scrum. Kanban kent geen vaste cadans van ceremonies, maar stuurt op continue flow en expliciete keuzemomenten: precies wat een omgeving met hoge agent-output vraagt.

DE PARADOX: SNELLER ÉN COMPLEXER

Agent-based development brengt een paradox met zich mee die herkenbaar is voor iedereen die Agile transformaties van dichtbij heeft meegemaakt. Agents maken development sneller en goedkoper, maar tegelijkertijd complexer en minder voorspelbaar.

De interessante verschuiving zit in de aard van de complexiteit. Het is niet langer de technische complexiteit van het bouwen die dominant is, maar de complexiteit van het coördineren, kiezen en valideren. En dat is precies het domein waar Scrum en Kanban hun kracht bewijzen: omgaan met onzekerheid via korte feedback loops, empirisch werken en continue aanpassing.

Twee scenario's

Scenario 1: AI Overload	Scenario 2: Sterke regie
Alles wordt gegenereerd, niemand kan bijbenen. Het resultaat: chaos en middelmatige keuzes. Meer output, maar slechtere uitkomsten.	Een scherpe productvisie en duidelijke kaders zorgen dat AI gericht wordt ingezet. Het verschil zit niet in technologie, maar in discipline en focus.

Menselijke aandacht als schaarse resource

Als we alle verschuivingen samenvatten, komen we op een kernconlusie: menselijke aandacht en oordeelsvermogen worden de schaarse resource in softwareontwikkeling. Tools worden slimmer, maar denken wordt waardevoller.

De gebruiker (of dat nu de eindgebruiker, de Product Owner of de organisatie is) verschuift van consumer naar curator. Niet langer "degene die iets vraagt," maar "degene die richting geeft en filtert."

Dit heeft directe consequenties voor hoe we teams begeleiden. De centrale vraag voor Agile coaches, Scrum Masters en teamleiders wordt: hoe help je teams en organisaties om niet te verdrinken in mogelijkheden?

AANBEVELINGEN VOOR TEAMS EN ORGANISATIES

Voor Scrum Masters en Agile Coaches

- **Verbreed de definitie van "team":** Beschouw agents als onderdeel van het systeem dat je faciliteert. Hun output, interacties en afhankelijkheden vallen onder het werkproces.
- **Stuur op cognitieve load:** Monitor niet alleen velocity maar ook de besliscapaciteit van het team. Te veel opties en te veel output kan verlammend werken.
- **Maak agent-gebruik zichtbaar:** Voeg agent-output toe aan het board, maak zichtbaar waar menselijke beslissingen nodig zijn en waar agents autonoom werken.
- **Faciliteer expliciete triage:** Richt het proces zo in dat er bewuste keuzemomenten zijn wanneer agents meerdere opties genereren.

Voor Product Owners

- **Investeer in scherpe productvisie:** Hoe goedkoper bouwen wordt, hoe waardevoller het vermogen om te kiezen wat je níet bouwt.
- **Definieer duidelijke acceptatiecriteria:** Agents leveren wat je vraagt. De kwaliteit van de vraag bepaalt de kwaliteit van het antwoord.
- **Beperk het aantal parallelle experimenten:** Meer opties is niet altijd beter. Pas WIP-limieten toe op het aantal gelijktijdig te evalueren varianten.

Voor ontwikkelteams

- **Investeer in architectuur:** DDD, heldere bounded contexts en API-contracts zijn de fundering voor effectief werken met agents.
- **Onderhoud je ambacht:** Zorg dat je problemen kunt analyseren en oplossen zonder agents. Volledige afhankelijkheid is een risico.
- **Ontwikkel systeemdenken:** De waarde verschuift van code schrijven naar het ontwerpen en beheersen van systemen waarin agents opereren.

Voor management en organisaties

- **Faciliteer leren:** Het AI-landschap verandert wekelijks. Investeer in continu leren, interne kennisdeling en experimenteerruimte.
- **Pas governance aan:** Bestaande kaders voor kwaliteit, security en compliance moeten worden uitgebreid naar agent-based werkprocessen.
- **Stuur op uitkomsten, niet op output:** Meer code is niet meer waarde. Meet resultaten bij de eindgebruiker, niet het volume van wat agents produceren.

CONCLUSIE

Agent-based development duwt softwareontwikkeling in de richting van een fundamentele verschuiving: van vakmanschap naar systeemregie. Dat is geen pure verbetering of verslechtering. Het hangt volledig af van hoe bewust je het inzet, hoe goed je je systeem ontwerpt en hoe sterk je feedback loops zijn.

Agile, Scrum en Kanban zijn bij uitstek geschikt om deze transitie te begeleiden. Ze bieden de empirische procescontrole, de focus op flow en de feedbackmechanismen die nodig zijn om de belofte van AI-agents waar te maken zonder in de valkuilen te trappen.

De uitdaging voor de komende jaren is helder: niet méér bouwen, *maar* beter kiezen. Niet toenemende output, maar betere regie. Dat begint bij teams die zich durven af te vragen of hun huidige werkwijze nog wel voldoet.

Dit document is tot stand gekomen in samenwerking met AI. De oorspronkelijke aanwijzingen, bronmateriaal en inhoudelijke kaders zijn door VX Company Consultancy Services zelf opgesteld. Er is een Large Language Model ingezet om een uitgebreidere conceptversie te genereren, waarna de tekst door de auteur inhoudelijk is herzien, herschreven en afgestemd op de VX Company-context.