TERUG NAAR HOME

MAKER MANUALS 04

AI-AUTONOMIE

Onafhankelijk blijven in een wereld vol modellen, platforms en AI-tools.

Formaat

Webeditie in neo brutalism-stijl

Licentie

Status

Alpha-editie 0.4

`Maker Manual 04`

`Onafhankelijk blijven tussen modellen, platforms en AI-tools`

`Alpha-editie 0.4 - vroege versie in ontwikkeling`

Dit boek is een tussenstand. Het onderwerp beweegt nog, dus de tekst beweegt mee.

Gebaseerd op het praktijkwerk en de nieuwsbrieven van Erwin Blom, met hulp van AI uitgewerkt tot een versie die verder kan groeien.

INHOUD

Hoofdstuk 1. Waarom AI-autonomie nu telt

De meeste mensen komen AI binnen via gemak. Een chatvenster dat meteen werkt. Een model dat helpt schrijven. Een tool die in een paar minuten iets oplevert waar je eerder een middag of een week aan kwijt was.

Dat voelt goed en vaak is het dat ook.

Het probleem komt later. Meestal niet op dag één, maar ergens veel verderop. Dan merk je dat je aantekeningen in chats zijn gaan

wonen. Dat vaste instructies nergens echt staan. Dat projectkennis half in je bestanden zit en half in een interface die je niet beheert.

Dan gaat dit onderwerp ineens niet meer over gemak.

Dan gaat het over afhankelijkheid.

AI-autonomie gaat voor mij niet over de droom om alles zelf te hosten en nooit meer iets van een leverancier te gebruiken. Dat is voor de meeste mensen niet haalbaar en meestal ook niet nodig. Het gaat over iets veel nuchterders: zorgen dat je werk niet stilvalt zodra een model verandert, een prijs omhoog gaat, een tool verdwijnt of een interface slechter wordt.

Die vraag dringt zich nu op omdat AI sneller beweegt dan gewone software. Bij een tekstverwerker of CMS kon je jaren vooruit zonder je hele manier van werken te herzien. Bij AI kan het speelveld in een paar maanden al anders zijn. Modellen schuiven, interfaces veranderen en nieuwe spelers duiken overal op.

Wie alles in één leverancier stopt, bouwt dus op lossere grond dan hij misschien denkt.

Daar komt nog iets bij. AI raakt niet alleen je software, maar ook je denkwerk. Bij een tekstverwerker kun je redelijk makkelijk weg. Bij een AI-systeem waarin je toon, context, werkwijze en gewoontes zijn gaan wonen, neem je niet alleen bestanden mee als je overstapt. Je moet ook een manier van werken terugwinnen.

Dat is de reden voor dit boek.

Gebruik de kracht van AI voluit, maar zorg dat wat jij opbouwt niet volledig van die krachtbron afhankelijk wordt.

Hoofdstuk 2. De nieuwe afhankelijkheid

Afhankelijkheid van technologie is oud nieuws. We leven al jaren tussen clouds, abonnementen en gesloten platforms. Toch voelt AI anders. Het kruipt dieper het werk in.

Bij oudere software wist je meestal nog goed wat van jou was. Je had documenten, spreadsheets, mappen, een inbox. Bij AI vervaagt die grens. Een gesprek levert een idee op, een analyse, een eerste versie, een besluit. Een week later wil je erop terugkomen en merk je dat het nergens echt geland is. Het zit half in een chat, half in je hoofd en half in een tool die alweer bezig is met het volgende.

Daar begint de nieuwe afhankelijkheid.

De eerste vorm zit in gedrag. Je went aan snelheid. Daardoor leg je minder netjes vast wat ertoe doet. Je vertrouwt erop dat de chatgeschiedenis het later wel teruggeeft. Soms lukt dat. Vaak niet.

De tweede vorm zit in structuur. Veel mensen bouwen kleine systemen van prompts, custom instructions, agent-workflows en projectruimtes binnen één tool. Daar is op zichzelf niets mis mee. Het wordt riskant zodra die hele werklaag in huurgrond blijft hangen.

De vraag is daarom niet of een tool goed is. De vraag is hoeveel schade het doet als je weg wilt.

Zodra je denkt: ik zou best willen wisselen, maar dan raak ik te veel kwijt, zit je al verder vast dan je lief is.

AI-tools helpen daar zelf ook aan mee. Ze willen dat je meer context aanlevert, meer taken in dezelfde omgeving doet en meer gewoontegedrag ontwikkelt. Dat is hun productlogica. Het maakt ze bruikbaarder, en plakkeriger.

Ik vind dat niet schokkend. Ik vind het wel iets om helder naar te kijken.

Niet elke afhankelijkheid is verkeerd. Sommige koop je bewust in omdat de winst groter is dan het risico. Het probleem begint pas wanneer je niet meer goed ziet wat van jou is gebleven en wat alleen nog in een platform leeft.

Wie dat onderscheid kwijt is, raakt sneller in de war bij elke AI-verschuiving.

Hoofdstuk 3. Bouw je eigen laag boven de modellen

Als je maar één gedachte uit dit boek meeneemt, laat het dan deze zijn: vertrouw niet alleen op de motor, bouw ook je eigen laag erboven.

De meeste AI-gebruikers beginnen andersom. Ze leren eerst een model kennen, dan een agent, dan een handige interface. Dat is logisch. Alleen blijft het daar te vaak bij. Daardoor werken ze rechtstreeks op de machine van iemand anders.

Een eigen laag hoeft niet ingewikkeld te zijn. Voor de meeste mensen is ze verrassend simpel:

- een logische mapstructuur
- bronbestanden die leesbaar blijven
- vaste templates
- een paar goede prompts in gewone documenten
- checklists
- projectoverzichten
- publicatielagen

Dat klinkt bijna te eenvoudig. Juist daar zit de kracht.

Wie zijn werk in gewone bestanden en heldere routines organiseert, hoeft minder te vertrouwen op toevallig chatgeheugen. Je kunt context opnieuw aan een model voeren. Je kunt een

andere tool op hetzelfde materiaal zetten. Je kunt later nog begrijpen hoe iets tot stand kwam.

Dan wordt een model een motor, geen woonplaats.

Markdown is hier een goed voorbeeld van. Niet omdat markdown heilig is, maar omdat het weinig pretenties heeft. Je kunt er boeken in schrijven, notities in bewaren, prompts in beheren, checklists in bouwen en publicatieroutes aan koppelen. Het dwingt je niet tot een zwaar systeem. Het laat je wel doorwerken als de tool van vandaag over een jaar niet meer de tool van morgen is.

Hetzelfde geldt voor prompts en instructies. Veel mensen hebben inmiddels sterke prompts, maar bewaren die rommelig. Een paar in een notitieapp, een paar in custom settings, een paar in oude chats, een paar in het hoofd. Dan heb je geen eigen laag. Dan heb je verspreide afhankelijkheid.

Zodra je die dingen als eigen documenten behandelt, verandert dat.

Hoofdstuk 4. Waarom chats geen geheugen mogen zijn

Chats zijn verleidelijk. Ze voelen snel, direct en slim. Je denkt, probeert, herschrijft en gaat weer verder. Daardoor lijkt een chatvenster al snel op een werkruimte, een notitieblok en een geheugen tegelijk.

Dat laatste is het probleem.

Chats zijn uitstekend als tussenlaag. Ze zijn zwak als eindbestemming.

De eerste reden is simpel: terugvinden gaat slecht. Zelfs in goede interfaces blijft een chatgeschiedenis een lang spoor van half

gelukte probeersels, losse ingevingen en af en toe iets dat echt goed was. Juist daarom verdwijnt het makkelijk.

De tweede reden is context. Een antwoord uit een chat is vaak alleen volledig te begrijpen binnen het gesprek waarin het ontstond. Haal je het eruit, dan ontbreken bron, opdracht of eerdere keuzes.

De derde reden is dat een chat zelden jouw eigen informatiestructuur volgt. Projecten lopen door elkaar. Thema's raken versnipperd. Belangrijke stukken zijn niet vanzelf gekoppeld aan je bestanden, je publicatielaag of je routines.

Dat zie je overal terug. Iemand heeft een scherp inzicht, een agent levert een bruikbare analyse, een planning klopt ineens, en een week later is het half verdwenen in de diepte van een tijdlijn.

Daarom hanteer ik liever een harde regel:

alles wat later nog waarde heeft, moet uit de chat naar je eigen laag.

Naar een markdownbestand. Naar een projectmap. Naar een bronnotitie. Naar een manuscript. De precieze vorm doet er minder toe dan het principe.

Een chat is een werkbank. Geen archief.

Hoofdstuk 5. Bestanden, formats en markdown

Bestanden klinken saai. Dat is hun voordeel.

In een wereld die voortdurend nieuwe interfaces, agents en slimme lagen voortbrengt, zijn gewone bestanden een vorm van rust. Ze vragen niet om bewondering. Ze vragen om beheer.

Wat in een eenvoudig of open format leeft, blijft verplaatsbaar.

Een tekstbestand kun je openen in allerlei programma's. Een markdownbestand kun je lezen zonder de tool die het maakte. HTML kun je publiceren, bewerken en omzetten. Een EPUB is breed bruikbaar. Een PDF is niet ideaal als bron, maar wel nuttig als distributielaag. Een nette mapstructuur maakt projecten begrijpelijk zonder dat je eerst een platform moet openen.

Daarom doen formats ertoe. Niet als technisch detail, maar als dagelijkse vrijheid.

Een gesloten tool kan geweldig zijn. Alleen betaal je daar later voor als jouw werk er niet zonder kleerscheuren uit kan. Misschien niet vandaag. Wel zodra je wilt migreren, hergebruiken, samenvoegen, publiceren of automatiseren.

Daarom hou ik mijn bronlaag liever eenvoudiger dan mijn zichtlaag.

Je website mag mooier zijn dan je bronbestanden. Je publicatie-ervaring mag rijker zijn dan je manuscriptlaag. Alleen wil je onder dat alles iets hebben dat leesbaar blijft en niet instort zodra een leverancier iets verandert.

Ook bestandsnamen en mapstructuur horen daarbij. Een goede bestandsnaam is geen detail. Een heldere map is geen decoratie. Het zijn kleine keuzes die later het verschil maken tussen rustig verder kunnen en onnodig zoekwerk.

Hoofdstuk 6. Werken met meerdere modellen

Niet elk model is overal het beste in. Dat is inmiddels wel duidelijk. Het ene denkt scherper mee, het andere schrijft prettiger, een derde codeert sterker, een vierde is sneller of goedkoper.

Die verscheidenheid is nuttig. Ze wordt pas onhandig als je zonder systeem gaat hoppen.

Veel mensen slaan door naar een van twee kanten. Ze doen alles hardnekkig in één tool, ook wanneer die niet meer de beste keuze is. Of ze springen van model naar model en verliezen ritme, consistentie en overzicht.

Ik geloof meer in een derde route.

Niet één model voor alles. Ook niet tien modellen door elkaar. Wel een kleine, bewuste set van uitwisselbare motoren met een eigen werklaag erboven.

Je kunt best een voorkeursmodel hebben voor redactie, een ander voor productbouw en een derde voor snelle research. Zolang de context, bronbestanden en kwaliteitslat niet in die tools opgesloten raken, is daar weinig mis mee.

Dan voelt een beter model als een upgrade, niet als een verhuizing.

Daarvoor moet je een paar dingen vermijden. Bewaar geen cruciale instructies alleen in custom settings van één platform. Laat projectkennis niet alleen in één chatomgeving ontstaan. Zorg dat goede prompts, stijlregels en werkafspraken ook buiten de tool bestaan.

Dat is de praktische kant.

De strategische kant is rust. Nieuwe spelers worden dan kansen in plaats van bedreigingen. Prijswijzigingen voelen minder dwingend. Hype verliest een deel van zijn macht. Je hoeft niet steeds meteen over. Je hoeft ook niet halsstarrig te blijven zitten.

Hoofdstuk 7. Wat houd je zelf, wat huur je in

Er bestaat een romantisch idee over autonomie: dat je alles zelf zou moeten bezitten. Je eigen infrastructuur, je eigen hosting, je eigen modellen, je eigen opslag, je eigen tooling.

Voor sommige specialisten is dat interessant. Voor de meeste mensen is het vooral ballast.

Volledige controle is duur. Niet alleen in geld, ook in aandacht. Alles wat je zelf optuigt, moet je onderhouden, beveiligen, begrijpen en blijven verbeteren.

Daarom is autonomie niet hetzelfde als alles zelf doen.

De relevante vraag is: wat moet van mij blijven, en wat mag best gehuurd zijn zolang ik de uitgang kan vinden?

Wat je meestal zelf wilt houden:

- bronbestanden
- projectstructuur
- eigen kennislagen
- prompts en instructies die veel waarde dragen
- publicatieteksten
- beslisregels en kwaliteitslatten

Wat je vaak prima kunt huren:

- rekenkracht van modellen
- specifieke interfaces
- specialistische apps
- snelle experimenteertools
- distributielagen zolang export mogelijk blijft

Daar tussenin zit het grijze gebied. Een CMS. Een notitieomgeving. Een automatiseringslaag. Een agentplatform. Dan gaat het minder om de tool zelf en meer om de exit.

Kun je exporteren? Kun je bestanden meenemen? Kun je elders verder? Zijn je belangrijkste routines nog te reconstrueren?

Daar zou ik steeds weer naar kijken.

Hoofdstuk 8. Je playbooks en kwaliteitslat

Veel mensen denken bij autonomie vooral aan tools en bestanden. Dat is terecht, maar niet genoeg.

Een groot deel van je werklaag zit in gewoontes. In hoe je iets aanpakt. In hoe je kwaliteit beoordeelt. In welke stappen je altijd zet voordat iets goed genoeg is.

Zolang die kennis alleen in je hoofd zit, blijf jij zelf de bottleneck.

Daarom zijn playbooks nuttig. Niet als bureaucratie, maar als geheugen voor goed werk.

Een playbook kan klein zijn. Een checklist voor publicatie. Een vaste structuur voor research. Een redactieroute. Een werkwijze voor het omzetten van bronmateriaal naar manuscript. Een standaard voor hoe je agent-output beoordeelt.

Zodra je zulke dingen opschrijft, gebeurt er iets belangrijks. Je maakt je werk overdraagbaar. Aan een ander mens, aan een toekomstig model en ook aan jezelf over drie maanden.

Je hoeft dan niet steeds opnieuw te bedenken wat goed eigenlijk betekent.

Dat maakt je minder speelbal van de tool. Wat er goed uitziet, voelt dan niet automatisch al goed genoeg. Je hebt een eigen norm.

Hoofdstuk 9. Bouw een klein systeem dat tegen een stootje kan

Een model-onafhankelijk systeem hoeft niet groot te zijn. Liever niet.

De fout die veel mensen maken, is dat ze autonomie verwarren met compleetheid. Dan willen ze ineens een perfect dashboard, een eigen geheugenlaag, een promptbibliotheek, een publicatiesysteem, een agentnetwerk en een mapstructuur voor alles.

Meestal krijg je dan meer ambitie dan werking.

Begin kleiner.

Voor veel mensen is een eerste sterke versie al dit:

- een duidelijke inbox voor losse input
- een bronlaag voor wat bewaard moet blijven
- een projectlaag per lopend onderwerp
- een manuscript- of outputlaag voor wat gepubliceerd wordt
- een paar vaste formats
- een document met werkinstructies

Meer heb je vaak niet nodig om al rustiger te werken.

Als je die basis hebt, kun je daar modellen bovenop zetten. Een model helpt dan bij ordenen, samenvatten, structureren, redigeren, analyseren of bouwen. Maar de ruggengraat van het werk blijft van jou.

Dat is de praktische vertaling van autonomie. Geen spectaculair systeem. Wel een systeem dat jou niet gijzelt.

De kunst is niet om een totaalplatform te dromen. De kunst is om een kleine werklaag te bouwen die tegen verandering kan.

Hoofdstuk 10. Onafhankelijk genoeg

Er zijn twee slechte eindbeelden rond AI.

Het eerste is totale overgave: alles in één platform, alles in chats, alles in de cloud van een leverancier, zonder plan B en zonder

eigen structuur.

Het tweede is zuiverheidsdrift: alles zelf willen, alles lokaal, alles open source, alles onder volledige controle, ongeacht de prijs in tijd, aandacht en complexiteit.

Beide zijn onhandig.

Het eerste maakt je kwetsbaar. Het tweede maakt je traag.

Daarom is de beste uitkomst meestal niet volledige autonomie, maar voldoende autonomie. Genoeg om te kunnen bewegen. Genoeg om te kunnen wisselen. Genoeg om niet opnieuw te hoeven beginnen als de markt verschuift.

Dat is ambitieus genoeg.

Wie onafhankelijk genoeg is, kan de beste motor van dit moment gebruiken zonder zichzelf eraan te verkopen. Je profiteert van snelheid en kwaliteit, maar bewaart je eigen basis.

Misschien is dat wel de volwassen vorm van AI-gebruik.

Niet de vraag welke tool wint.

Wel de vraag hoe jij je werk zo inricht dat goede tools je helpen, zonder dat ze je opsluiten.

LICENTIE

- delen mag
- hergebruiken mag
- bewerken mag
- ook commercieel gebruik is toegestaan

Zolang je Erwin Blom als bron noemt en afgeleide versies onder dezelfde licentie deelt.

Meer informatie:

Creative Commons BY-SA 4.0