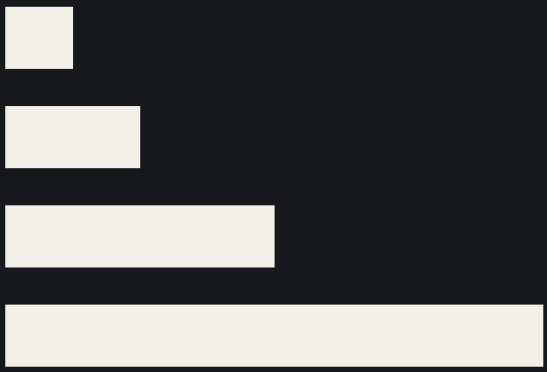
AI-AGENTEN DIE HET WERK DOEN

Hoe je ze maakt en aan het werk zet



FLORIS KNOL

AI-agenten die het werk doen

Hoe je ze maakt en aan het werk zet

Floris Knol

Inhoud

Voorwoord

Inleiding

Hoe de opdrachten in dit boek werken

De voorbeeldmap

DEEL I — ÉÉN DING GOEDKOOP MAKEN

1 Eerst het gereedschap, dan het ding

2 Wat wegviel, en wat niet

3 “Ja, maar jij bent programmeur”

4 Feiten, geen meningen

5 Bereikbaarheid boven waarde

6 Multipliers

DEEL II — EN DAN HONDERD TEGELIJK

7 En als je het duizend keer deed?

8 Waarom doe je dat nog zelf?

9 Zichzelf controleren

10 Snelheid is geen comfort

11 Niet op schijf, niet gebeurd

DEEL III — ALS HET MISGAAT

12 Fratsen

13 Fouten zijn je beste leraar

14 Mijn gereedschap loog tegen me

15 De tweede keer

16 Zoek de sleutel, niet de deuren

17 Nul afhankelijkheden

DEEL IV — HET WORDT EEN ORGANISATIE

18 Het vliegwiel

19 Wat een agent mist

20 Grenzen, en wie tekent

21 Verbeteren zonder release

DEEL V — IN DE ECHTE WERELD

22 Toegang is schaarser

23 Vastzitten is een ontwerpkeuze

24 Meedoen in de formele wereld

25 Als iedereen het kan

26 Waar staan we nu

TOT SLOT

27 Waar gaat dit heen

ACHTERIN

Termenlijst

Voorwoord

Er is iets veranderd in het werk, en bijna niemand kan je precies vertellen wát.

Je hoort dat AI alles verandert. Je ziet voorbeelden die indrukwekkend zijn en waar je vervolgens niets mee kunt. Je leest berichten van mensen die het weten, cursussen die het beloven, en artikelen die na drie alinea's zeggen dat je vooral moet beginnen. En dan sta je maandagochtend voor je eigen werk en is er niets veranderd.

Dit boek is geschreven door iemand die het wél heeft gedaan, en het gaat over wat hij daarbij heeft geleerd.

Wat hier gebeurde

Waar dit boek over gaat is een omgeving waarin agenten zelfstandig werk doen. Niet één assistent die je helpt, maar meerdere die tegelijk draaien, met een opdracht, een grens en een plek waar ze opschrijven wat ze deden. En dat is op zijn beurt geen eindpunt: het is de stap naar een organisatie die op AI draait in plaats van er gebruik van maakt.

Ik heb dat gebouwd, vanaf niets, met wat er nu gewoon te krijgen is. Wat het verschil maakte was de manier van kiezen, en die vond ik onderweg.

Het ongewone eraan is niet de snelheid. Het is dat het elke dag makkelijker werd in plaats van moeilijker.

Dat is bij bijna alles andersom. Wat je bouwt wordt zwaarder: meer onderdelen, meer uitzonderingen, meer dingen die kapot kunnen, en dus minder tempo. Hier gebeurde het omgekeerde, en dat kwam niet door geluk of door talent. Het kwam door een klein aantal zetten die er op de dag zelf niets uitzagen.

Die zetten zijn wat dit boek je geeft.

Een handboek van patronen die zich telkens herhaalden, met de voorbeelden erbij waaruit ze kwamen.

Alles is opgeschreven terwijl het gebeurde: wat er gedaan werd, waarom, wat het kostte, en wat er misging. Daar komen een stuk of tien patronen uit. **Die patronen zijn het boek; waar ze toevallig uit voortkwamen, is het voorbeeld** — en ze staan hier in de volgorde waarin je ze nodig hebt, niet in de volgorde waarin ze ontstonden.

Dat onderscheid is belangrijk, want anders leest de helft van de lezers mee als toeschouwer. Ik bouwde iets technisch, maar geen van de patronen gaat over techniek. Ze gaan over de volgorde waarin je dingen doet, over waar je tijd heen gaat, over wat je aan een machine kunt overlaten en wat niet, en over hoe je merkt dat je de verkeerde kant op werkt. Een boekhouder, een fotograaf, een advocaat en een aannemer komen ze alle vier tegen. Daarom staat bij elk patroon ook een voorbeeld dat niets met software te maken heeft.

En je kunt het zelf nadoen. Elk getal in dit boek komt uit werk dat echt is gedaan, en bij elk getal staat hoe het gemeten is. De opdrachten draai je op je eigen scherm, en dan zie je of je in de buurt komt. Dat is

zeldzaam in dit onderwerp en het is met opzet: wat je zelf hebt zien werken, hoef je van mij niet aan te nemen.

Inclusief wat er misging. Er staat in dit boek een dag waarop ik drie keer achter elkaar de verkeerde conclusie trok, een moment waarop mijn eigen gereedschap tegen me loog, en een meting die bewees dat mijn manier van meten stuk was en niet het ding dat ik maat. Die staan er niet uit bescheidenheid. Ze staan er omdat daar het meeste te leren viel.

Waar dit heen loopt

Wat begint bij één persoon met een leeg apparaat, eindigt bij iets dat zelf werk aanneemt in de echte wereld.

Zo gezegd klinkt het als grootspraak, en dat snap ik. Maar het is letterlijk de opbouw van dit boek, en de stappen ertussen zijn geen van alle spectaculair. Het helpt om ze vooraf te zien, want dan begrijp je waarom het begin zo klein is.

Het begint bij één ding goedkoop maken. Eén taak sneller, één controle die zichzelf doet, één fout die je niet twee keer maakt. Dat is de eerste helft van het boek en het is gewoon handwerk.

Daarna gaan er honderd tegelijk. Niet omdat het indrukwekkend is, maar omdat de vraag verandert zodra er meer dan één is: wat je bij één met een blik ziet, moet bij honderd opgeschreven staan, en wat je bij één zelf doet, moet bij honderd zonder jou kunnen.

Dan gaat er iets mis, en herstelt het zichzelf. Iets dat merkt dat het vastloopt en stopt. Iets dat ziet dat dezelfde fout terugkomt en de instructie aanpast in plaats van de fout. Iets dat een grens heeft en er niet overheen gaat, ook niet als het uitkomt.

En dan wordt het een organisatie. Dat woord is niet groot bedoeld. Een organisatie is iets dat een doel heeft dat niemand elke ochtend opnieuw hoeft te geven, een budget dat op kan, rollen die van elkaar weten, en een geheugen. Zodra die vier er zijn, is het geen gereedschap meer dat op jou wacht.

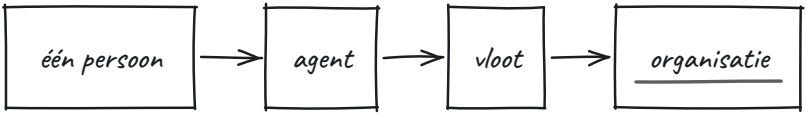
En dan komt het deel dat bijna niemand noemt. Zo'n organisatie doet dingen in de echte wereld. Ze koopt in. Ze meldt zich aan bij diensten en gaat daarmee verplichtingen aan. Ze huurt mensen in — want het idee dat er dan geen mensen meer aan te pas komen, is een fantasie: er is werk dat een mens moet doen en werk waarvoor een mens moet tekenen. Ze stelt doelen bij, en soms schaalt ze zichzelf op.

En dus moet ze **meedoen in de formele wereld.** Er komt een accountant. Er komt een audit. Er is een norm waaraan een proces moet voldoen, en iemand die vraagt: laat maar zien wie dit heeft goedgekeurd, wanneer, en waarop hij zich baseerde.

Dat is geen lastige bijkomstigheid maar de voorwaarde. Een organisatie die niet kan laten zien wat ze gedaan heeft, is niet autonoom — die is onbeheerd. Het verschil tussen die twee woorden is precies het verschil tussen iets wat je in een bedrijf mag inzetten en iets wat je alleen thuis kunt uitproberen.

En het aardige is dat het dezelfde dingen zijn. Het vastleggen dat je nodig hebt om ervan te leren, is het vastleggen dat een auditor wil zien. De grens die voorkomt dat het geld op raakt, is de grens waar een controller naar vraagt. Wie het ene goed doet, heeft het andere er gratis bij — en wie het overslaat, moet het later alsnog bouwen op het moment dat het het slechtst uitkomt.

Ik ben nog niet op het punt van een autonome organisatie. Ik zit ergens in het midden van die opbouw, en het boek zegt precies waar. Maar de richting is duidelijk, en elke stap erheen is er een die je vandaag al kunt zetten.



elke stap maakt de volgende mogelijk

De opbouw van dit boek: wat begint bij één persoon, eindigt bij iets dat zelf werk aanneemt.

Hoe je het leest

De vijf stappen hierboven zijn de vijf delen van het boek, in precies die volgorde: één ding goedkoop maken, dan honderd tegelijk, dan wat er gebeurt als het misgaat, dan wordt het een organisatie, en dan de echte wereld. Daarna volgt nog één hoofdstuk dat verder kijkt dan waar ik zelf sta — het enige dat nog geen praktijk is.

Elk hoofdstuk daalt minstens één keer af naar iets dat je vanavond kunt uitproberen. Het staat gewoon in de tekst, met erbij waarom het werkt en waar het misgaat. Je kunt eroverheen lezen; je kunt ze ook allemaal doen, en dan heb je aan het eind iets gebouwd terwijl je las.

En elk hoofdstuk sluit met één regel: **wat dit goedkoper maakte.** Die regels stapelen. Lees ze een keer achter elkaar terug en je ziet waar het boek eigenlijk over gaat.



De vijf delen staan op elkaar: elk deel maakt het volgende mogelijk.

Waarom nu

Omdat je nog niet te laat bent.

Dat is geen verkooppraatje maar een waarneming. De dingen die dit mogelijk maken zijn er nu allemaal, ze zijn goedkoop, en bijna niemand gebruikt ze samen. Dat is precies het moment waarop iemand die het wél doet een voorsprong opbouwt die niet in te halen is met geld — want de voorsprong zit in wat je op elkaar hebt gestapeld, en stapelen kost tijd die niemand kan overslaan.

Over twee jaar is dit gewoon. Nu nog niet.

Inleiding

Ik ben programmeur. Dat is de makkelijkste manier om me weg te zetten, dus laat ik er meteen iets naast leggen.

Ik heb ook jarenlang mensen begeleid naar werk. Mensen zonder betaald werk, mensen met autisme, mensen die lang buiten het arbeidsproces stonden, mensen voor wie de gewone route niet werkte. Dat is ander werk dan software bouwen — andere woorden, andere uitkomsten, en niemand verwart de twee.

In dat werk leerde ik vier dingen af en aan, en geen ervan ging over computers. **Vooroordelen helpen niet**, want wie met een dossier in zijn hoofd binnenkomt ziet bevestiging van wat hij al dacht. **Opbouwen wél**, stap voor stap, omdat een stap die mislukt alleen iets vertelt als het er één was. **Herhalen en stapelen**, want wat één keer lukt kan toeval zijn en wat drie keer lukt is een bodem om op te staan. En **systematisch en voorspelbaar maken** — dezelfde volgorde, dezelfde woorden — omdat je daarmee alle ruis weghaalt die niets met de zaak te maken heeft.

Er was er nog één, en die bleek later de belangrijkste: **geef feiten, geen aannames en geen meningen.** *"Hij is niet gemotiveerd"* is een eindpunt. *"Hij kwam de afgelopen drie keer op dinsdag niet"* is een beginpunt. Het eerste sluit af, het tweede opent.

Toen ik met AI ging werken, hoefde ik niets nieuws te leren. Ik deed wat ik al deed, en het werkte meteen.

Dat verbaasde me meer dan het had moeten doen. Een model heeft geen voorgeschiedenis met je, kan niet raden wat je bedoelt, en leest niet tussen de regels door. Alles wat je niet als feit levert, vult het zelf in. Precies dezelfde beperking als in een kamer met iemand die je voor het eerst ziet — en dus precies dezelfde werkwijze.

Zo is dit boek ontstaan. Niet vanuit een idee over AI, maar vanuit de vaststelling dat drie werelden die niets met elkaar te maken hebben, dezelfde vragen stellen. De patronen hierin heb ik niet bedacht; ik heb ze herkend. Dat maakt ze niet nieuw. Het maakt ze bruikbaar op plekken waar je ze niet zou zoeken.

Wat er wél nieuw aan is, is niet het idee. **Het is dat er nu iets bestaat dat ze uitvoert terwijl jij er niet bij bent.**

Dit boek gaat over wat dat verandert, en wat je moet doen om het te laten kloppen.

En daarom staat er geen enkele truc in. Wat een paar maanden werkt en daarna onjuist is, leert je niets — je weet nog steeds niet wáárom het werkte toen het werkte.

Wat er wel in staat zijn concepten en patronen. Die gaan over de volgorde waarin je dingen doet, over waar je aandacht heen gaat, over wat je kunt overlaten en waaraan je dat ziet. Zulke dingen zijn niet gebonden aan een programma, een leverancier of een versie. **Ze golden voordat dit gereedschap bestond, ze gelden nu, en ze gelden nog steeds als alles wat je vandaag gebruikt vervangen is door iets beters.**

Dat is de belofte van dit boek, en het is de enige die ik durf te doen: je hebt er over een paar jaar nog steeds iets aan, in welke omgeving je ook werkt.

En toch daalt elk hoofdstuk één keer af naar iets dat je vanavond kunt doen. Dat lijkt in tegenspraak met de alinea hierboven en dat is het niet — maar het verdient een eigen uitleg, en die staat hierna. Wie liever meteen begint slaat hem over en gaat naar hoofdstuk 1.

Hoe de opdrachten in dit boek werken

Dit hoofdstuk staat vooraan omdat het over de vorm gaat en niet over de inhoud. Je kunt het overslaan en meteen aan hoofdstuk 1 beginnen; je komt dan alleen een paar keer iets tegen waarvan je je afvraagt waarom het er zo uitziet.

In het kort: elk hoofdstuk dat zich ervoor leent daalt één keer af naar iets dat je vanavond kunt doen. Wat hieronder staat is waarom dat er is, hoe je het gebruikt, waar de voorbeelden vandaan komen, en wat er aan veranderd is voordat het in dit boek kwam.

Wat een uitgewerkt voorbeeld doet

Een uitgewerkt voorbeeld is de kortste weg van een concept naar iets dat op jouw scherm gebeurt, en juist daardoor blijft het concept hangen. Je hebt het één keer zien werken; de tweede keer verzin je hem zelf.

Dus daalt elk hoofdstuk dat zich ervoor leent één keer af naar iets dat je vanavond kunt doen: wat je doet, wat je letterlijk typt, wat je terugkrijgt, en waar het misgaat. Reken op vijf tot twintig minuten. Je kunt ze overslaan en het boek klopt nog; je kunt ze allemaal doen en dan heb je aan het eind iets gebouwd terwijl je las.

De voorbeelden zijn geschreven voor één programma, en dat is Claude Code. Niet omdat het het beste is en niet omdat het moet, maar omdat er iets moet staan dat je kunt overtypen. Een voorbeeld dat zegt *"vraag je AI om zoiets"* is geen voorbeeld maar een uitvlucht: het schuift het enige moeilijke stuk — precies opschrijven wat je wilt — terug naar de lezer.

Bij elke andere aanbieder werkt het hetzelfde, met dezelfde woorden. Wat deze voorbeelden van je vragen is nooit een knop, een menu of een instelling; het is een opdracht in gewone taal, en gewone taal is nu juist wat al die programma's gemeen hebben. Wáár je het intypt verschilt en verandert bovendien elk halfjaar. Wát je intypt niet.

En dit is geen inleiding in het gebruik van zo'n programma. Dat je een opdracht kunt typen en antwoord krijgt, wordt hier verondersteld; daar zijn handleidingen voor en die zijn beter dan wat ik erover zou schrijven. Wat hier begint is de stap dáárna, en dat is een andere stap dan de meeste mensen denken.

Die stap is: van iets dat antwoordt naar iets dat wérkt terwijl jij er niet bij bent. Niet een gesprek waarin jij elke beurt de volgende zet doet, maar een opdracht die wordt uitgevoerd, gecontroleerd en afgemaakt zonder dat jij ertussen zit. Het verschil tussen die twee is niet gradueel. Zodra jij niet meer elke beurt meekijkt, moet alles wat jij stilzwijgend deed ergens anders vandaan komen: wat er precies moet gebeuren, waar het ophoudt, waaraan je ziet dat het klopt, en wat er gebeurt als het niet klopt.

Dat is waar dit boek over gaat, en waar de voorbeelden je heen brengen. Het eerste is klein en doe je nog met je hand op de knop. Elk

volgende haalt een stukje van jou uit de lus — tot er iets staat dat een opdracht aanneemt, uitvoert, zichzelf nakijkt en pas terugkomt als het klaar is of als het ergens niet uit komt.

Alle voorbeelden gaan over hetzelfde bouwbedrijf. Eén wereld die meegroeit: opdrachten, opdrachtgevers, personeel met certificaten, materieel met keuringen, facturen, een agenda, aanbestedingen — en twaalf regels waar het zich aan hoort te houden. Klein genoeg om erin te kijken, echt genoeg om ergens op mis te gaan. Het eerste voorbeeld laat je die map beschrijven; daarna werkt elk voorbeeld erin. Aan het eind heb je één ding opgebouwd in plaats van twintig losse proefjes.

Die map maak je zelf, en dat kost één opdracht. Hoe dat gaat en waar je op let, staat in het volgende hoofdstuk — daar begint het werk.

En alles wat je in die voorbeelden ziet terugkomen, is er echt uitgekomen. Elk voorbeeld in dit boek is uitgevoerd in die map, met precies de opdracht die erbij staat. Wat er onder *wat er terugkwam* staat, is wat er terugkwam — niet een nette versie ervan, niet iets dat ik heb overgetypt omdat het beter stond.

Twee dingen heb ik er wél aan gedaan, en het is eerlijker om ze te noemen. Ik heb de stukjes uitvoer **ingekort** zodat ze op een boekbladzijde passen: nooit meer dan acht regels en nooit breder dan de kolom. En ik heb er de paden en de datums van mijn eigen machine uit gehaald, omdat die bij jou toch anders zijn. **Geformuleerd is er niets.** Waar een antwoord onhandig was, staat het onhandig; waar het fout was, staat de fout er nog — en in een paar gevallen is die fout het punt van het hoofdstuk geworden.

Eén ding heb ik niet echt gedaan, en dat hoor je te weten. In de laatste hoofdstukken reiken de voorbeelden naar buiten: een e-mail aan een opdrachtgever, een melding aan de gemeente, een boeking in een boekhouding. Die opdrachten zijn echt uitgevoerd, maar in een oefenopstelling — er is geen mail verstuurd en er is met geen enkel systeem van iemand anders gepraat. Wat er terugkomt is wat zo'n ontvanger zou antwoorden, inclusief de keren dat hij níet antwoordt.

Bij jou gaat het wél echt de deur uit, en dat is precies waarom die hoofdstukken over grenzen en handtekeningen gaan.

Dat is met opzet. Een voorbeeld dat is bijgewerkt tot het mooi stond, zou precies de ervaring wegpoetsen die je zelf gaat hebben.

Kopieer ze niet. Doe ze.

Hier wil ik één ding met nadruk zeggen, omdat het het verschil maakt tussen een boek dat je hebt gelezen en een boek waar je iets aan hebt gehad.

De opdrachten in dit boek zijn geen recepten. Ze staan er zodat je ziet hoe zo'n opdracht eruitziet en wat hij oplevert — niet zodat je ze overtikt en dan verder leest. Wie ze alleen overtikt, krijgt iets soortgelijks als wat er in het boek staat en leert daar bijna niets van, want het klopte al.

En let op: **je krijgt sowieso niet letterlijk wat er in dit boek staat.** Vraag twee keer hetzelfde en je krijgt twee keer iets soortgelijks, in andere woorden en soms in een andere volgorde. Dat is geen storing maar hoe dit werkt — en het is de tweede reden om niet te vergelijken maar te proberen. Wat moet kloppen is de uitkomst, niet de bewoording.

Verander ze. Zet er je eigen regel in, je eigen map, je eigen getal. Vraag om iets anders dan wat er staat. Laat iets weg en kijk wat er dan gebeurt. **Dat is waar het leren zit**, en het is ook precies wat er gebeurt zodra je het op je eigen werk toepast: daar staat nooit hetzelfde als hier.

Daarom eindigt bijna elk voorbeeld met een tweede ronde. Eerst de opdracht, dan wat er terugkwam, dan wat er niet goed aan was, en dan een bijgestelde opdracht met een nieuw antwoord. Die tweede ronde is er niet voor de volledigheid. Hij laat zien dat de eerste poging zelden goed is, en dat bijsturen geen teken is dat je het niet snapt maar de manier waarop dit werkt.

Doe er één. Verpruts hem. Stel hem bij. Dan heb je het te pakken.

En ze veranderen van soort onderweg

Wat je doet verandert door het boek heen, en dat is de hele beweging die dit boek beschrijft.

In het begin typ je een opdracht en lees je het antwoord. Jij zet elke keer de volgende stap. Dat is waar bijna iedereen blijft hangen, en het is ook prima — er valt veel te winnen met alleen dat.

Even verderop leg je vast wat je net deed, zodat je het niet elke keer opnieuw hoeft uit te leggen. Je bent dan niet meer aan het uitvoeren maar aan het inrichten, en je merkt dat je sommige dingen nooit meer typt.

Daarna zet je dingen klaar die zichzelf aanroepen. Een controle die draait als er iets is bijgekomen. Een grens die eerst tegenhoudt en dan

pas vraagt. Jij bent er op dat moment niet bij, en dat is het punt: de volgende stap wordt niet meer door jou gezet maar door de situatie.

En aan het eind zet je geen opdrachten meer klaar. Dan loopt er iets dat je ooit hebt opgezet. Je checkt in, je kijkt wat er is gebeurd terwijl je er niet was, en af en toe stuur je iets bij. Meer niet. **Het loopt door.**

Daarom ziet een voorbeeld in het laatste deel er anders uit dan een voorbeeld in het eerste. Daar staat geen *typ dit* meer, maar *kijk wat er is gebeurd*. Dat is niet omdat er niets meer te doen is — het is omdat het doen ergens anders is komen te liggen.

Dat zeg ik hier één keer, en daarna niet meer bij elk voorbeeld. Een voorbehoud dat honderd keer terugkomt, wordt honderd keer overgeslagen.

De voorbeeldmap

Alle opdrachten in dit boek spelen in dezelfde map. Voordat je aan hoofdstuk 1 begint, moet die map er zijn — en dat kost één opdracht.

Waarom een bouwbedrijf

Het gaat om werk met afspraken erin, en dat heeft elke organisatie — een school, een cateringbedrijf, een administratiekantoor. Een bouwbedrijf is in dit boek een concreet voorbeeld.

Daarin zit een web dat administratie stukmaakt. Opdrachten verwijzen naar opdrachtgevers, opdrachtgevers naar facturen, facturen naar uren, uren naar mensen, mensen naar certificaten met een geldigheidsdatum. Dat is precies wat een agent kan nalopen en jij niet.

En je hebt er geen verstand van nodig. Iedereen snapt dat een certificaat geldig moet zijn voordat je de steiger op gaat. Vervang bouwbedrijf door wat jij doet — de rest van het boek leest hetzelfde.

De opdracht

Maak een lege map, open je agent erin, en typ dit:

> *Maak in deze map de administratie van een bouwbedrijf met vijf man vast personeel. Gebruik mappen voor opdrachten, klanten, personeel met certificaten, materieel met keuringen, facturen, urenstaten, offertes, inkoop, onderaannemers, agenda en aanbestedingen — een stuk of dertig in totaal, met een paar documenten elk. Schrijf alles als markdown met velden bovenaan. Zet er een* `regels.md` *bij met twaalf regels waar de administratie zich aan hoort te houden: nummering van opdrachten, geldige certificaten bij het werk dat iemand doet, keuringen van materieel, een grens voor meerwerk, en zo verder. Laat elke regel precies één keer overtreden worden, verspreid over de map, zonder dat erbij staat waar.*

Reken op een paar minuten. Je krijgt geen voortgangsbalk; je ziet mappen verschijnen en daarna een samenvatting van wat er is gemaakt.

Waar je op let als hij klaar is

Vier dingen, en ze zijn allemaal te tellen:

- **ongeveer dertig mappen** op het eerste niveau, met de elf uit de opdracht erbij
- **rond de honderd documenten**, verdeeld over die mappen
- `regels.md` **met twaalf regels**, genummerd
- **velden bovenaan elk document** — een opdracht heeft een nummer, een klant, een bedrag en een datum, geen lopende tekst waar je die uit moet vissen

Klopt dat, dan kun je verder. De twaalf overtredingen ga je niet zoeken: dat is het werk van hoofdstuk 1 en verder, en de agent heeft ze met opzet niet opgeschreven.

Wat je terugkrijgt is niet wat ik heb

Jouw map ziet er anders uit dan de mijne. Andere namen, andere bedragen, andere volgorde, en waarschijnlijk een paar mappen die ik niet heb — of andersom.

Dat is goed, en het is de reden dat dit werkt. De patronen in dit boek hangen niet aan een bestandsnaam. Ze hangen aan de vorm: er staan afspraken vast, er is iets dat zich er niet aan houdt, en er is een manier om dat te vinden zonder alles met de hand na te lopen.

Wat wel afwijkt zijn de **getallen in de uitkomsten.** Waar in dit boek staat dat een controle er negen vond, vindt die van jou er misschien zeven of elf. Vergelijk dus niet het getal maar de vorm: kreeg je een lijst met vindplaatsen, of kreeg je "ziet er goed uit"? Dat verschil is waar het in hoofdstuk 4 over gaat.

Als het misgaat

Er zijn drie manieren waarop dit niet meteen lukt, en geen ervan vraagt om opnieuw beginnen.

Te weinig mappen. Sommige agenten maken er tien en stoppen. Vraag om aanvulling: *"maak er nog twintig bij, dezelfde opzet"*. Aanvullen is goedkoper dan overdoen, en het is meteen de eerste keer dat je merkt hoe zo'n gesprek werkt.

Geen overtredingen. Een agent die nette administratie maakt, heeft het laatste deel van de opdracht genegeerd — dat gebeurt, want het is een rare vraag. Zeg het dan expliciet: *"voeg nu per regel uit regels.md precies één overtreding toe, verspreid, en schrijf nergens op waar"*.

Lopende tekst in plaats van velden. Krijg je documenten die lezen als een brief, vraag dan om de vorm: *"zet bovenaan elk document de velden op een eigen regel, zoals* `bedrag: 1850`*"*. Zonder die vorm kan een controle er later niet op rekenen.

En als je het toch liever overdoet: gooi de map leeg en typ de opdracht opnieuw. Er gaat niets verloren — dit is precies het soort werk dat goedkoop hoort te zijn, en dat is waar hoofdstuk 10 over gaat.

Dit is de enige stap in het boek die je maar één keer doet. Alles daarna bouwt erop voort: elk voorbeeld werkt in deze map, en aan het eind staat er iets dat zichzelf nakijkt.

1. Eerst het gereedschap, dan het ding

Het is twee minuten over acht 's avonds. Er is nog niets.

Vier minuten later staan er vier dingen op schijf. Geen van de vier is het ding dat gebouwd moest worden.

Er staat **gereedschap** om het mee te maken. Er staat een **manier om te controleren** of wat er gemaakt wordt ook klopt. Er staat een **lijst van wat er nog moet**, met de volgorde erin. En er staan **voorbeelden**: een paar kleine dingen die al werken, om van af te kijken.

Het ding zelf staat er niet. Dat komt morgen.

Wat daar gebeurde

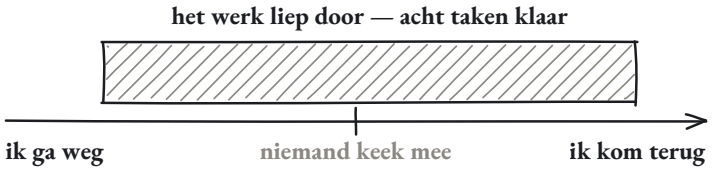
Iemand die iets wil maken, maakt het. Dat is niet gek: je hebt een idee, je gaat aan de slag, en het gereedschap komt wel als je het nodig hebt. Zo doet bijna iedereen het, en zo had ik het zelf ook altijd gedaan.

Wat er die avond gebeurde was iets anders. De eerste zet was niet *beginnen* maar **zorgen dat beginnen goedkoop wordt.**

Dat kostte een avond en leverde die avond niets op. Er was aan het eind van de dag niets te laten zien. Er was alleen een plek waar iets maken minder tijd zou gaan kosten dan ergens anders.

Dat is het eerste patroon van dit boek, en het is het patroon waar alle andere op rusten:

> **Voordat je het ding maakt, maak je de dingen die het ding goedkoop maken om te maken — en goedkoop om te controleren.**



Wat er gebeurde tussen het moment van weglopen en het moment van terugkomen.

Waarom die avond er überhaupt was

Er hoort een reden bij, want anders is dit een avond met een goede gewoonte en niet het begin van iets.

De aanleiding was de omvang. Niet van het project — van het gereedschap. Een moderne ontwikkelstapel is zo groot geworden dat er geen korte lus meer in past. Er zit zoveel tussen het moment dat je iets verandert en het moment dat je weet of het klopt: bouwen, laden, wachten, starten, afhankelijkheden die iets anders willen. Voor een mens is dat vervelend. Voor een machine die het honderd keer achter elkaar doet, is het fataal.

En dat is het hele probleem in één zin: bij zo'n stapel kán er geen korte terugkoppeling voor een machine ontstaan. Alles wat er in dit boek staat over proberen, meten, opnieuw proberen en zichzelf la-

ten nakijken, rust op een lus die kort genoeg is om honderd keer te draaien. Is hij dat niet, dan is het allemaal theorie.

Die avond ging dus over het inkorten van die lus, en wat eruit kwam was wat er nodig was om hem kort te krijgen.

Verwachtte ik dat het zou lukken? Ja, en dat is geen bravoure. Ik had het de maanden ervoor al zien gebeuren in het klein — genoeg om te weten dat het zou moeten kunnen. Niet *dat het zou lukken* maar *dat het moest kunnen*, en dat is precies het verschil tussen een gok en een verwachting die op iets rust. Dit is dezelfde beweging als het turven op dat kladblaadje uit hoofdstuk 3: eerst meten, dan zeker weten.

Dit gaat niet over software

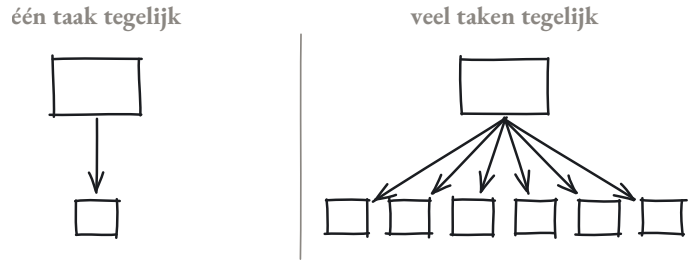
Het klinkt misschien alsof dit iets is voor mensen die programma's schrijven. Het is precies andersom: het valt daar alleen sneller op, omdat je er alles kunt meten.

Een fotograaf die tweeduizend opnamen per opdracht maakt, kan die allemaal één voor één bekijken. Of hij kan eerst een halve dag besteden aan het vastleggen van zijn belichting en zijn bewerking, en daarna begint elke opdracht op tachtig procent. Die halve dag levert die dag niets op.

Een advocaat kan elk contract opnieuw schrijven. Of hij kan een verzameling clausules aanleggen die hij hergebruikt, en dan is het schrijven van een contract iets anders geworden dan het daarvoor was.

Een docent kan elk jaar dezelfde uitleg geven en elk jaar opnieuw merken waar leerlingen struikelen. Of hij schrijft één keer op wáár ze struikelen, en verandert de uitleg daar.

In alle drie de gevallen is de zet die telt de zet die op die dag niets oplevert. Dat is geen toeval; dat is de definitie. Een zet die meteen iets oplevert, levert meestal alleen dát op.



Eén persoon, één taak tegelijk — en dezelfde persoon met veel taken tegelijk.

Zelf doen · Vraag wat er eigenlijk in staat

Wat je doet. Open de voorbeeldmap en vraag wat erin zit. Niets bouwen, niets veranderen — alleen kijken. Dit is de kortste versie van wat dit hoofdstuk beschrijft: je typt één zin en je krijgt iets terug waar je anders een ochtend voor had gezeten. Reken op twee minuten.

TYP DIT

> Dit is de administratie van een bouwbedrijf. Beschrijf in tien regels welke bedrijfsprocessen je hier terugziet, en noem daarna drie dingen die je opvallen.

Wat je ziet. Een opsomming van processen die je zelf nooit zo op een rij had gezet — van acquisitie en calculatie tot kwaliteitsborging en nazorg — en drie opmerkingen over wat eruit springt. **Let op de derde opmerking.** De eerste twee zijn meestal wat je al wist; de derde is regelmatig iets waar je nooit naar had gekeken.

WAT ER TERUGKWAM

```
koppelingen 17  opdrachten 15  uitvoering 14
urenstaten 12   agenda 11      ontwikkeling 11
kam 11          facturen 9     calculaties 8
magazijn 8      personeel 8
werkvoorbereiding 8
... 49 mappen, 254 bestanden
opvallend: 1 gat in de opdrachtnummering
opvallend: 1 opdracht zonder materiaal
```

Waar het misgaat. Vraag om een aantal regels. Zonder "in tien regels" krijg je een verslag van twee kantjes, en dan lees je alsnog alles. **Vraag er iets bij dat een oordeel vraagt.** "Beschrijf wat erin staat" levert een inhoudsopgave op; "noem drie dingen die je opvallen" dwingt tot kiezen, en in dat kiezen zit de waarde. **En geloof de opsomming niet zomaar.** Wat hier opgesomd wordt is wat er in de bestandsnamen staat, niet wat er werkelijk gebeurt — dat verschil is precies waar hoofdstuk 14 over gaat.

EN DAN ZO

> Noem bij elk van die drie opvallende punten het bestand waar je het uit haalt, zodat ik het zelf kan nakijken.

EN DAN DIT

```
gat in de nummering
  -> opdrachten/, 15 bestanden, NLOP-2026-005 ontbreekt
opdracht zonder materiaal
  -> opdrachten/NLOP-2026-003.md
opdrachtgever onbekend
  -> opdrachten/NLOP-2026-006.md
```

Het ongewone deel

Wat er later stond had ik in mijn eentje nooit kunnen bouwen als ik gewoon was begonnen. Maar dat is niet het interessante getal.

Het interessante is dit: **de laatste dag leverde meer op dan de eerste.**

Bij bijna alles is dat andersom. Wat je bouwt wordt zwaarder naarmate het groter wordt — meer onderdelen, meer uitzonderingen, meer dingen die kapot kunnen, en dus minder tempo. Iedereen die ooit aan iets groots heeft gewerkt kent het gevoel dat het steeds stroperiger wordt.

Hier gebeurde het omgekeerde, en dat is geen geluk. Elke keer dat ik koos voor de zet die niets opleverde maar de volgende zetten goedkoper maakte, werd de dag erna een stukje lichter. Die zetten stapelen, en gestapeld gedragen ze zich anders dan opgeteld.

Dat is waar dit boek over gaat. Niet over wat ik gebouwd heb — dat is het voorbeeld. Over de keuzes die maken dat werk zichzelf lichter maakt in plaats van zwaarder, en hoe je ze herkent op het moment dat je ze moet nemen.

Want dat is het lastige eraan. Achteraf is zo'n zet vanzelfsprekend. Op de avond zelf ziet hij eruit als uitstel.

Wat dit goedkoper maakte: alles wat erna kwam. De avond waarop er niets gemaakt werd, is de reden dat er daarna zoveel gemaakt kón worden.

2. Wat wegviel, en wat niet

Na een week viel me op dat ik dingen niet meer deed. Niet omdat ik ze had afgeschaft, maar omdat ze niet meer langskwamen.

Het aardige is dat je dat niet merkt terwijl het gebeurt. Er is geen moment waarop iets ophoudt. Je kijkt op enig moment terug en denkt: wanneer heb ik dat voor het laatst gedaan.

Waarom je het pas achteraf ziet

Hier zit een asymmetrie die het waard is om te kennen, want hij geldt ver buiten dit onderwerp.

Wat erbij komt, merk je meteen. Een nieuw hulpmiddel, een nieuwe stap, een nieuwe vergadering: die vragen aandacht, dus je ziet ze. **Wat wegvalt, vraagt per definitie geen aandacht meer.** Er is geen melding, geen moment, geen laatste keer die zich als laatste keer aankondigt.

Het gevolg is dat je de winst structureel onderschat. Je ziet de dingen die er bij kwamen — de extra controles, de nieuwe manier van opschrijven — en die voelen als werk. De vier dingen die verdwenen, tel je niet mee, want ze zijn er niet.

De enige manier om het te zien is twee weken naast elkaar leggen. Niet uit het hoofd, want je geheugen doet precies het omgekeerde:

het bewaart wat aandacht vroeg. Vandaar de oefening verderop in dit hoofdstuk — het is geen invuloefening maar het enige instrument dat hier werkt.

Dit is dezelfde beweging als op het callcenter uit het volgende hoofdstuk: niet vinden maar tellen. Wat je niet telt, kun je alleen maar voelen, en gevoel kijkt de verkeerde kant op.

De ene taak, en beide keren in tijd

Voordat ik het opsom, één concreet geval, want een opsomming zonder getal is een mening.

Vroeger implementeerde ik. Nu stuur ik aan. Dat klinkt als een woordkeuze en het is een andere eenheid: **vijf minuten aansturen levert werk op waar ik vroeger weken handmatig programmeren voor nodig had.**

Dat is de helft van het antwoord. De andere helft is belangrijker en wordt bijna nooit genoemd: **er komen ook dingen uit die ik zelf niet zomaar zou kunnen.** Niet "sneller hetzelfde" — ook dingen die buiten mijn eigen bereik lagen.

Let op wat die twee samen betekenen, want ze wijzen niet dezelfde kant op. Een factor is een verhouding: hetzelfde werk, minder tijd. Het tweede is geen verhouding maar een grens die verschuift. **En alleen het eerste voelt als winst.** Het tweede voelt in het begin ongemakkelijk, want het is precies het deel waar je zelf niet meer kunt nakijken of het zo hoort — daar gaat de rest van dit boek over.

Wat wegviel

Het uittypen. Niet het denken — het typen. De honderd regels die volgen uit één beslissing, het vierde geval van hetzelfde patroon, de vertaling van iets dat je al helemaal voor je zag. Dat was nooit het interessante deel en het was wel het grootste deel.

Het onthouden. Waar iets stond, hoe het ook alweer heette, welke van de drie manieren we hier gebruikten. Dat is geen geheugenwerk dat ergens toe leidt; het is wrijving die je zelf draagt.

Het wachten. Niet op de machine, want die was snel. Op mezelf. Op een avond waarop ik er nog zin in had, op een blok van twee uur, op het moment dat het hele plaatje weer in mijn hoofd zat.

Het vervelende deel van zorgvuldig zijn. Overal hetzelfde aanpassen, nalopen of het ergens anders ook stond, de laatste twee gevallen die je bijna vergeten was. Dat is precies het werk dat je doet als je moe bent, en precies het werk dat dan misgaat.

En het opnieuw beginnen. Vroeger kostte een verkeerde afslag een dag, dus dacht je lang na voordat je hem nam. Als een verkeerde afslag twintig minuten kost, neem je hem gewoon en kijk je wat eruit komt.

wat wegviel	wat ervoor in de plaats kwam
het uittypen	de vorm volgde zelf
het onthouden	het stond ergens
het wachten	het liep door
het nalopen	een toets deed het

Wat wegviel zat bijna allemaal tussen het besluit en het resultaat.

Wat niet wegviel

En nu het deel dat belangrijker is, want dit is waar de meeste verhalen over dit onderwerp ophouden.

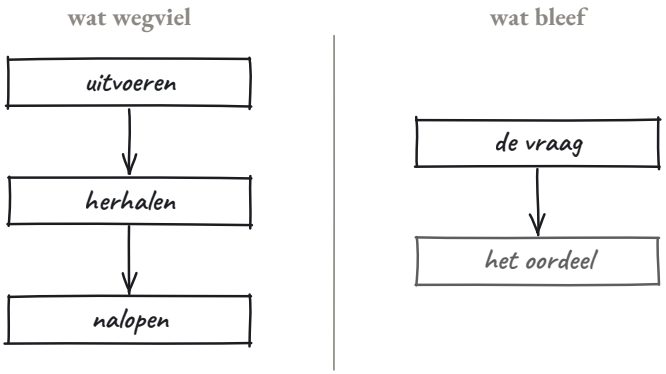
Beslissen wat er moet komen. Geen enkele keer is er iets nuttigs ontstaan doordat ik vroeg om iets nuttigs. De vraag stellen, en hem scherp genoeg stellen, is nog steeds het hele begin — en het is moeilijker geworden, niet makkelijker, omdat er nu veel meer mogelijk is dan er nodig is.

Oordelen of het goed is. Niet of het werkt; of het goed is. Of dit de vorm is die je over een half jaar nog wilt, of de uitleg klopt voor iemand die er niet bij was, of het klein genoeg is. Dat blijft mensenwerk en het wordt zwaarder naarmate er meer langskomt.

Weten waar het over gaat. Je kunt niets beoordelen in een vak dat je niet kent. Wat je wél kunt, is sneller leren — maar de dag dat je het niet weet, kun je ook niet zien dat het antwoord mis is.

De verantwoordelijkheid. Als er iets de deur uitgaat, is het van jou. Dat verandert niet doordat iemand anders het typte, en het verandert al helemaal niet doordat iets anders het typte.

En de smaak. Wat je mooi vindt, wat je te veel vindt, waar je niet aan wilt beginnen. Daar is geen instructie voor, en dat is de reden dat twee mensen met hetzelfde gereedschap twee heel verschillende dingen maken.



Wat wegviel zat tussen besluit en resultaat. Wat bleef staat ervoor en erna.

De vorm die eronder zit

Leg de twee lijstjes naast elkaar en er staat iets eenvoudigs.

Wat wegviel zat allemaal tussen het besluit en het resultaat. Het was de afstand, niet de richting. Wat bleef, staat aan de uiteinden: wat wil ik, en is dit het.

Dat is niet nieuw en het is ook niet van software. Elke keer dat de afstand tussen besluit en resultaat korter werd — de rekenmachine, de spellingcontrole, de zoekmachine — verdween de uitvoering en bleef het oordeel. Wat nieuw is, is hoeveel er in één keer verdwijnt.

En daar zit meteen het ongemak. Als de uitvoering het grootste deel van je werk was, voelt dit als verlies. Als je vak eigenlijk in de vraag en het oordeel zat, voelt het als lucht. Beide zijn waar, en welke van de twee het voor jou is, hangt niet af van je vak maar van waar jij je tijd aan besteedde.

Weggevallen is niet hetzelfde als weg

En nu de kanttekening die dit hoofdstuk nodig heeft, anders leest het te mooi.

Kijk nog eens naar het vierde item: *het vervelende deel van zorgvuldig zijn*. Overal hetzelfde aanpassen, nalopen of het ergens anders ook stond, de laatste twee gevallen die je bijna vergat.

Dat werk was vervelend, maar het was niet nutteloos. **Terwijl je het deed, kwam je langs alles.** Dat is de reden dat je er wel eens iets vond dat je niet zocht. Wie dat werk kwijtraakt, raakt ook die toevallige blik kwijt — en dat is geen kleinigheid.

Dus hoort er bij elk ding dat wegviel één vraag:

> **Is het verdwenen, of is het onzichtbaar geworden?**

Verdwenen is goed: het hoefde niet meer, want er is iets anders dat de uitkomst garandeert. Onzichtbaar is gevaarlijk: het gebeurt niet meer én er is niets voor in de plaats gekomen, alleen merk je dat pas als er iets misgaat dat je vroeger toevallig zag.

Het verschil is te meten, en dat is het goede nieuws. Voor elk van de vijf dingen hierboven: noem het dat het overneemt. Het uittypen is overgenomen door een generator, het onthouden door wat er op schijf staat, het nalopen door een controle die bij elke verandering draait. Kun je bij een regel geen opvolger noemen, dan is dat geen winst maar een gat — en het heeft een naam, want het is precies een frats: er is niets te zien, en het lukt voortaan net iets vaker net niet.

Die vraag is ook de reden dat de rest van dit boek over controles gaat. **Een taak overdragen zonder de controle erbij over te dragen, is geen overdracht maar een weglating.**

Zelf doen · Laat het narekenen wat jij anders zelf naliep

Wat je doet. Vier van de twaalf regels in `regels.md` gaan over of een opdracht compleet is. Die loop je met je ogen na zolang het er vijftien zijn, en niet meer zodra het er vijftienhonderd zijn. Laat ze narekenen. Reken op vijf minuten.

TYP DIT

> In regels.md staan twaalf regels. Neem regel 1 tot en met 4 en controleer de map opdrachten/ en meerwerk/ daarop. Geef per regel alleen de overtredingen, met het nummer erbij, en verder niets.

Wat je ziet. Vier korte lijstjes, samen zes regels lang. Drie van de vier regels worden precies één keer overtreden. **Het gat in de nummering is de interessantste**: er is geen bestand dat ontbreekt, er is een nummer dat nooit gebruikt is — en dat vind je alleen door te tellen wat er níet is.

WAT ER TERUGKWAM

```
regel 1  nummering
  ontbreekt: NLOP-2026-005
regel 2  opdrachtgever
  NLOP-2026-006: supermarkt-van-loon-filiaal-oost
regel 3  uren en materiaal
  NLOP-2026-003
regel 4  meerwerk boven 1000
  NLMW-2026-06: 3950 euro
```

Waar het misgaat. De eerste versie sorteerde de nummers als tekst. Dan komt 10 vóór 2 en meldt hij gaten die er niet zijn — dat gebeurde hier ook echt, en het kostte één ronde. Vraag er daarom bij dat de num-

mers als getal behandeld worden. **"En verder niets" hoort erbij**, anders krijg je bij elke regel ook een opsomming van alles wat wél klopt. **En zet er daarna expres een fout bij**: verwijder een `materiaal:`-regel uit een opdracht en draai opnieuw. Vindt hij er twee, dan kijkt hij echt.

EN DAN ZO

> Haal nu de regel "materiaal:" weg uit NLOP-2026-004 en draai regel 3 opnieuw. Ik wil zien of hij hem vindt.

EN DAN DIT

```
regel 3  uren en materiaal
  NLOP-2026-003
  NLOP-2026-004
2 van de 15 opdrachten
```

Dat onderscheid is niet academisch. Het bepaalt wat je morgen nog zelf doet.

Zelf doen · Splits wat je overlaat van wat je houdt

Wat je doet. Je controle vond negen dingen. **De vraag van dit hoofdstuk is niet hoeveel hij er vond maar welke jij nog moet beantwoorden.** Dat is een andere vraag dan of hij goed werkt, en het is de enige die zegt wat er echt van je bord af ging. Reken op vijf minuten.

TYP DIT

> Neem de bevindingen van de vorige ronde en verdeel ze in twee stapels: wat uit de gegevens zelf volgt, en wat een oordeel vraagt dat niet in deze map staat. Zeg bij die tweede stapel per geval welk oordeel dat is.

Wat je ziet. Zes van de negen zijn na te rekenen: een nummer ontbreekt, een certificaat is verlopen, een keuring te oud. **Drie niet.** Of je door-

werkt met een onderaannemer die niet in orde is, wie er schuift bij een dubbele boeking, en of een marktsignaal zonder bron weg moet of nagevraagd — daar staat het antwoord nergens. **En let op wat er niet gebeurde: hij vond ze alle negen.** Het vinden ging weg, het beslissen niet.

WAT ER TERUGKWAM

```
na te rekenen : 6 van de 9
vraagt oordeel: 3 van de 9
  regel 7  doorwerken of stilleggen?
  regel 8  wie schuift, en wie bel je?
  regel 12 weggooien of navragen?
```

Waar het misgaat. Vraag om het oordeel te BENOEMEN en niet te geven. Vraag je "wat moet ik doen", dan krijg je een advies dat plausibel klinkt en nergens op rust. Vraag je "welk oordeel is hier nodig", dan krijg je de vraag terug — en dat is wat je nodig hebt. **Verwacht geen scherpe scheiding.** Regel 8 is half en half: dát er geschoven moet worden volgt uit de gegevens, wie er schuift niet. Zet zulke gevallen bij het oordeel, niet bij het rekenwerk. **En doe dit niet één keer.** Over een half jaar staat er meer in de map, en dan verschuift de grens — dat is precies wat dit hoofdstuk zegt dat er gebeurt.

EN DAN ZO

Zeg er per geval uit de tweede stapel bij wat je zou moeten vastleggen om hem de volgende keer wél na te kunnen rekenen.

EN DAN DIT

```
regel 7  vastleggen: bij welke tekortkoming
         we stilleggen en bij welke niet
regel 8  vastleggen: wie voorrang heeft als
         twee opdrachten botsen
regel 12 vastleggen: hoe lang een signaal
         zonder bron mag blijven staan
drie regels erbij, en de stapel oordelen wordt
de volgende keer kleiner
```

Wat er níet voor in de plaats kwam

Eén ding heeft me verrast, en het is de moeite waard omdat bijna iedereen het omgekeerd verwacht.

Er kwam geen vrije tijd voor in de plaats.

De vrijgekomen uren gingen naar werk dat er vóór die tijd niet was: nadenken over wat er zou moeten zijn, nakijken of het klopt, opschrijven waarom iets zo besloten is. Dat is ander werk, het is zwaarder per uur, en het is precies het werk waar de rest van dit boek over gaat.

Wie hoopt dat dit tijd oplevert, moet het nu zeggen. Het levert geen tijd op. Het levert bereik op: je komt verder met dezelfde dag. Dat is wat anders, en het is meer waard.

Probeer: schrijf een gewone werkweek in twee kolommen — links alles wat je deed tussen *"ik weet wat er moet gebeuren"* en *"het staat er"*, rechts wat daarvóór en daarná kwam. Laat Claude Code er per regel bij zetten of het uitvoering was of oordeel. De linkerkolom is waar de winst zit en hij is groter dan je denkt. Schrijf op wat je werkelijk deed, niet wat er in je functieomschrijving staat.

Wat dit goedkoper maakte: *de afstand tussen een besluit en het resultaat werd zo kort dat een verkeerd besluit nemen goedkoper werd dan er lang over nadenken.*

3. “Ja, maar jij bent programmeur”

Dit is het bezwaar dat altijd komt, en het is een goed bezwaar. Laat ik het eerst eerlijk maken voordat ik het beantwoord.

Het klopt. Ik ben programmeur. En het klopt ook dat een deel van wat hier gebeurde niet lukt als je dat niet bent — niet omdat je te dom zou zijn, maar omdat er dingen zijn die je alleen doet als je weet dat ze bestaan.

Wat ik betwist is niet het bezwaar. Het is wélk deel er precies aan mijn vak hangt.

Voordat ik programmeur was

Het eerlijkste antwoord op dit bezwaar komt uit een tijd waarin het nog niet op mij van toepassing was.

Tijdens mijn studie werkte ik op het callcenter van een bank. Bellen, een script volgen, verkopen. Ik was er niet bijzonder goed in. En op een gegeven moment stelde ik mezelf een vraag die niet in de training zat: *hoe kan ik hier meer succes hebben?*

Wat ik toen deed was niet slim en niet technisch. Ik legde een kladblaadje naast de telefoon en begon te **turven**. Welke reactie kwam er op welke zin. Niet hoe het voelde — wat er gebeurde. Na een paar honderd ge-

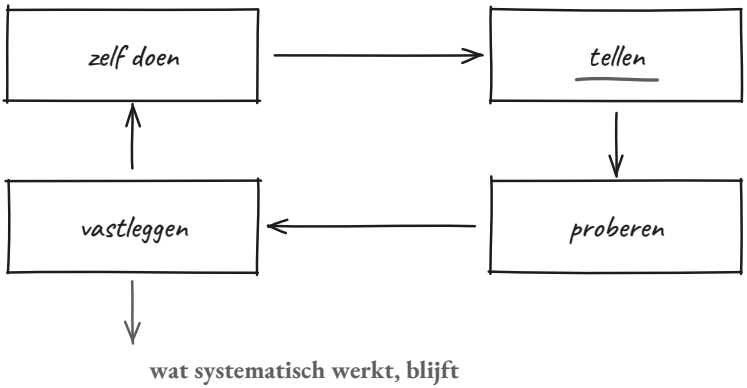
sprekken zag ik patronen die niemand me had verteld, en die ook niemand wist, omdat niemand ze had geteld.

Toen ging ik het script aanpassen. Niet aan de feiten — die veranderden niet — maar aan de volgorde en de formulering. Weer turven. Weer aanpassen. Wat het beter deed, hield ik; de rest ging eruit.

Het werkte. Ik ging van platte verkoop naar premium retentie en het terughalen van klanten die weggingen — het moeilijkste werk dat er op zo'n vloer te doen is. En het interessante is: dat was niet omdat ik een betere beller was geworden. Ik hield er alleen een systematische werkwijze op na. Uiteindelijk maakte ik er een stappenplan van en automatiseerde ik de saaie stukken — voor een belvloer een omslachtige manier van doen, en precies daarom viel het op.

Die lus is daarna nooit meer weggegaan:

1. **Zelf doen, handen vuil maken.** Niet in theorie begrijpen maar echt: voelen wat er gebeurt. Alles hierna is waardeloos als deze stap wordt overgeslagen.
2. **Meten, bijhouden, registreren.** Turven. Feiten, geen indrukken
3. **Experimenteren en aanpassen.** Eén ding tegelijk, zodat je weet wat het deed
4. **Opnieuw meten.** Anders weet je niet of het toeval was
5. **Behouden wat systematisch werkt.** En alleen dat
6. **Formaliseren**: er een werkwijze van maken, een systeem, een stuk gereedschap



Zelf doen, tellen, proberen, vastleggen — en dan opnieuw. Wat systematisch werkt, blijft.

Vijfentwintig jaar later sta ik op precies datzelfde punt, met ander gereedschap en een veel grotere hefboom. Dit boek is die lus, toegepast op agenten.

En daarom is "ja, maar jij bent programmeur" geen antwoord op wat hier staat. Toen ik dit voor het eerst deed, was ik het niet.

Wat het vak er daarna aan toevoegde

Toen ik wél programmeur was, kwam er iets bij. Drie dingen, en ze zijn concreter dan "hij kan programmeren".

Ik wist waar ik moest kijken. Toen het antwoord dat ik terugkreeg er goed uitzag maar niet klopte, kon ik dat zien. Dat is geen slimheid maar bekendheid: ik had die fout eerder gemaakt. Wie een vak niet kent, mist dat.

Controleren was voor mij bijna gratis. Ik kon iets laten draaien en binnen een seconde weten of het klopte. Dat is een enorm voordeel, en

het is het onderwerp van een later hoofdstuk — want het verklaart bijna alles wat hier goed ging.

Ik kon zelf een stukje bouwen. Niet groot, maar genoeg om een gat te dichten. Dat is de vaardigheid die het verschil maakt tussen "dit kan niet" en "dit kan nog niet".

Alle drie zijn echt. Ze maken dit boek voor een programmeur makkelijker dan voor iemand anders, en ik ga niet doen alsof dat niet zo is.

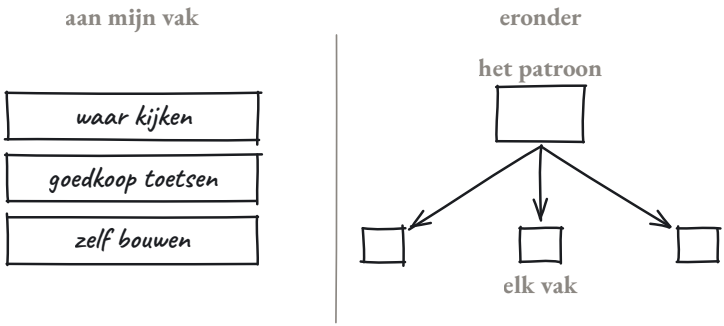
Maar let op wat ze zijn: **een versnelling op een werkwijze die er al was.** Ze maakten de lus hierboven sneller. Ze hebben hem niet bedacht.

En het vak daarna

Er kwam nog een vak bij, en dat is het vak waar de meeste patronen in dit boek vandaan komen.

Ik heb jarenlang mensen begeleid naar werk — mensen zonder betaald werk, mensen met autisme, mensen die lang buiten het arbeidsproces stonden. In dat werk leerde ik: geen aannames vooraf, stap voor stap opbouwen, herhalen tot iets geen toeval meer is, en voorspelbaar maken zodat de ruis eruit gaat. En vooral: feiten geven in plaats van oordelen.

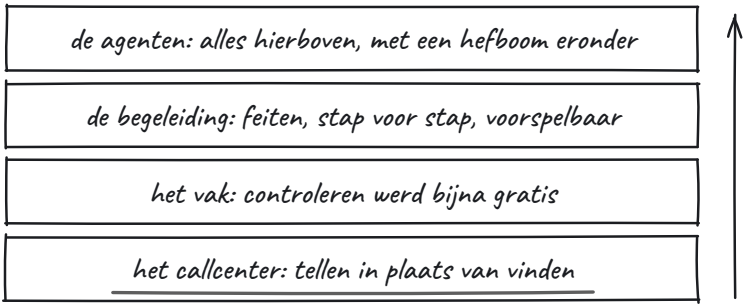
Toen ik met AI ging werken, hoefde ik daar niets aan te veranderen. Niet als metafoor — letterlijk hetzelfde, met ander gereedschap. Dat is geen grappig toeval; het is de reden dat dit boek bestaat. Een patroon dat je in één vak ziet, is een mening. Hetzelfde patroon in vier vakken die niets met elkaar te maken hebben — een callcenter, begeleiding, software, agenten — is een patroon.



Wat aan het vak hangt is klein. Wat eronder ligt, geldt overal.

Vier lagen, en ze bleven alle vier staan

Dat is het punt van deze volgorde, en het is belangrijker dan elk van de vier apart: **het stapelt.**



Vier lagen, in de volgorde waarin ze kwamen. Elke laag bleef staan.

Op het callcenter leerde ik **tellen in plaats van vinden**. In de software werd **controleren bijna gratis**, en dat is wat de lus van maanden naar minuten bracht. In de begeleiding leerde ik **feiten geven in plaats van oordelen**, stap voor stap opbouwen, en voorspelbaar maken. En bij de

agenten kwam er niets nieuws bij — daar kwam er een hefboom onder alles wat er al stond.

Alle vier de lagen bleven staan, en juist daarom houdt het. Een werkwijze die je in één vak leert en in het volgende weer inruilt, is een gewoonte. Een werkwijze die vier keer achter elkaar bruikbaar blijkt, is iets anders.

En dat is meteen het eerlijke antwoord op het bezwaar. Van die vier lagen is er precies één die aan mijn vak hangt. Hij helpt enorm. Hij is niet de bodem.

Wat er die eerste avond in zat, en wat niet

Het is de moeite waard om precies op te schrijven wat er aan middelen was, want bij zo'n verhaal vult een lezer het vanzelf aan met dingen die er niet waren.

Wat erin zat: een lege computer met internet, en één AI-assistent. Dat was het.

En wat er wél was, is het eerlijke antwoord op dit hele hoofdstuk: vijfentwintig jaar werkervaring. Dat is geen bescheiden lijstje. Het is precies de vier lagen hierboven, en het is het enige dat niet te kopen was.

Let op wat dat betekent voor de lezer, want het snijdt twee kanten op. **De gunstige kant:** alles wat er aan middelen in zat, heeft iedereen. Een computer, een verbinding, een abonnement. Er is geen drempel die met geld te nemen was en die jij niet kunt nemen. **De onaangename kant:** wat het verschil maakte, was het enige dat tijd kost. Er is geen versie van dit verhaal waarin dat deel wordt overgeslagen.

Daarom staat er in dit boek nergens dat het snel gaat zonder dat je iets kent. Het gaat snel *boven op* wat je kent — en wat je kent hoeft niet software te zijn. Dat is het punt van de vier lagen: de onderste is een kladblaadje op een belvloer.

De vraag die je wél moet beantwoorden

Het bezwaar is dus niet "jij bent programmeur". Het echte bezwaar zit in het tweede punt hierboven, en het is scherper:

Hoe controleer jij of het klopt?

Bij mij was dat bijna gratis. Bij een jurist, een boekhouder, een docent of een monteur is dat de vraag waar alles op staat of valt. En hij is te beantwoorden, want elk vak heeft ergens iets dat als toets kan dienen — het staat alleen zelden klaar.

Wat je zoekt is niet "een expert die het nakijkt". Dat is de dure vorm. Je zoekt iets dat **je zelf kunt aflopen**, dat **snel** is, en dat **hetzelfde antwoord geeft als iemand anders hem draait**. Drie eisen, en de eerste twee zijn de lastigste.

- **Een boekhouder** heeft cijfers die moeten sluiten, en dat is een toets die zichzelf afvinkt.
- **Een jurist** heeft de bron: staat het er ook echt, in dat artikel, in dat contract. De controle is niet "klinkt dit goed" maar "sla het op".
- **Een docent** heeft het werk van de leerling: begrijpen ze het na afloop.
- **Een monteur** heeft de installatie zelf: doet hij het, en doet hij het nog steeds na een uur.

In alle vier is de vorm dezelfde en is de kunst hetzelfde: **bedenk de toets vóór je begint**, en niet als er iets ligt waar je een gevoel bij hebt. Zonder toets vooraf beoordeel je achteraf op overtuigingskracht, en dat is precies waar dit gereedschap goed in is en jij slecht.

Zelf doen · Laat een patroon zoeken in tekst die niemand heeft geordend

Wat je doet. De dagrapporten in `uitvoering/dagrapporten/` zijn losse zinnen die een uitvoerder aan het eind van de dag opschreef. Er staat geen veld in, niets is geteld, en toch zit er een patroon in. Laat dat zoeken. Reken op drie minuten.

TYP DIT

> Lees alle dagrapporten in uitvoering/dagrapporten/ en vertel me welk soort verstoring het vaakst terugkomt. Groepeer naar oorzaak, niet naar bewoording, en onderbouw het met de regels waar je het uit haalt.

Wat je ziet. Tien dagen, zeven met een bijzonderheid, en ze zijn alle zeven anders opgeschreven. Maar gegroepeerd naar oorzaak vallen er drie samen: **de verstoring komt van iemand buiten de eigen ploeg** — een kraanchauffeur, een onderaannemer, een bewoner die niet thuis was. **Dat is geen conclusie die in de gegevens stond**; hij ontstaat pas als je naar oorzaak groepeert in plaats van naar woorden.

WAT ER TERUGKWAM

```
van buiten de eigen ploeg (3 van de 7):
  kraan stond stil, chauffeur te laat
  oorzaak zit bij de dakdekker, doorgezet
  drie woningen niet toegankelijk
overig: weer (1), maatvoering (1),
        meerwerk (1), zaterdagtoeslag (1)
```

Waar het misgaat. Zeg erbij dat je naar oorzaak wilt groeperen. Doe je dat niet, dan krijg je zeven keer één, want alle zinnen zijn anders geformuleerd — en dan lijkt er geen patroon te zijn. **Vraag om de onderbouwing.** Zonder de regels erbij kun je niet zien of de groepering klopt of dat er iets is bijgedacht, en dat is precies waar je bij dit soort vragen op moet letten. **En reken op één ronde bijsturen.** De eerste indeling was hier "weer, mensen, techniek", en dat klopte niet: de kraanchauffeur en de dakdekker zijn geen mensenprobleem maar een afhankelijkheid.

EN DAN ZO

> Groepeer opnieuw, maar nu op één vraag: hadden wij deze verstoring kunnen voorkomen, ja of nee.

EN DAN DIT

```
konden wij niet voorkomen (4 van de 7):
  kraan stond een uur stil
  halve dag stilgelegd
  oorzaak zit bij de dakdekker
  drie woningen niet toegankelijk
konden wij wel voorkomen (3 van de 7):
  maatvoering wijkt 4 cm af
  afwijking t.o.v. bestaande situatie
```

Dat onderscheid is niet academisch. Het bepaalt wat je morgen nog zelf doet.

Zelf doen · Laat het gat beschrijven in plaats van dichten

Wat je doet. Vraag nu eens iets dat **niet** kan. Niet om te zien of het misgaat, maar omdat het beschrijven van een gat de helft van het werk is — en dat deel kun je wel. Reken op vijf minuten.

TYP DIT

> Ik wil een controle op de vraag of de planning haalbaar is. Bouw hem niet. Zeg eerst welke gegevens daarvoor nodig zijn en welke daarvan niet in deze map staan. Zeg daarna wat er met wat er wél is te zeggen valt.

Wat je ziet. Vier dingen ontbreken, en ze zijn geen van alle technisch: hoeveel een soort werk realistisch kost, wie wat kan, hoe lang een levering echt duurt, en wat er al aan afspraken vastligt. **Maar het antwoord is niet "kan niet".** Uit wat er wél staat volgt een getal: begroot en werkelijk lopen tot zestien procentpunten uiteen. **Dat is geen antwoord op de vraag, maar het is er wel de helft van** — en het is een getal waar je morgen mee verder kunt.

WAT ER TERUGKWAM

```
ontbreekt:
  uren per soort werk, vaardigheid per persoon,
  echte levertijden, vastgelegde afspraken
wel aanwezig:
  urenstaten, nacalculaties, agenda, verlof
wat daar al uit volgt:
  begroot tegen werkelijk loopt tot 16 punten
  uiteen -- dat is de helft van het antwoord
```

Waar het misgaat. Zeg erbij dat hij het niet moet bouwen. Laat je dat weg, dan krijg je een controle die werkt op de gegevens die er zijn en zwijgt over wat ontbreekt — en dat is precies de fout waar hoofdstuk 14 over gaat. **Vraag naar de GEGEVENS en niet naar de oplossing.** "Wat heb ik nodig" geeft een boodschappenlijst; "welke gegevens ontbreken hier" geeft een lijst die je kunt nalopen. **En vraag altijd wat er wél te zeggen valt.** Anders eindigt het bij "onvoldoende gegevens", en dan heb je vijf minuten besteed aan een nee.

EN DAN ZO

> Zet die vier ontbrekende gegevens op volgorde van wat het minste werk kost om vast te leggen.

EN DAN DIT

```
1  echte levertijden -- staat in de orderbe-
   vestigingen, alleen niet verzameld
2  uren per soort werk -- af te leiden uit
   urenstaten plus opdrachten
3  vastgelegde afspraken -- moeten uit hoofden
4  vaardigheid per persoon -- vraagt een oordeel
   per mens, en dat kost het meest
```

En als je niets kunt bouwen

Dan valt het derde punt weg, en dat kost je iets. Je kunt een gat niet zelf dichten, en je bent afhankelijk van wat er bestaat.

Wat ervoor in de plaats komt is niet niets. **Je kunt het gat beschrijven**, en dat is de helft van het werk: benoemen wat er precies ontbreekt, wat je zou willen dat er gebeurde, en waaraan je zou zien dat het gelukt was. Dat is precies de vorm van een opdracht die nog werkt als iemand anders — of iets anders — hem uitvoert.

En het gaat de goede kant op. Het deel van dit werk waarvoor je moet kunnen programmeren, wordt elk jaar kleiner. Het deel waarvoor je moet kunnen bedenken wat er moet gebeuren en of het klopt, wordt elk jaar groter.

Dat is geen troostprijs. Dat is waar het vak heen gaat.

Probeer: neem één taak uit je eigen vak die je zou willen overlaten, en vraag om drie manieren om te controleren of het resultaat klopt zonder

het over te doen — van goedkoop naar duur, en bij elke erbij wat hij *niet* vangt. De goedkoopste bruikbare toets bepaalt wat je kunt overlaten, want die draai je ook echt. En een controle waarvan je de blinde vlek niet kent, geeft vertrouwen op de verkeerde plek.

Heb je die toets eenmaal, dan kantelt het bezwaar. De vraag is dan niet meer of je het kunt, maar waar je begint.

Probeer ook: vraag welke vijf dingen in jouw vak het vaakst herhaald worden en door iemand anders gedaan zouden kunnen worden als ze maar goed genoeg beschreven waren — met erbij waaraan je zou zien dat het goed gedaan is. *"Jij bent programmeur"* betekent meestal: ik weet niet waar ik zou beginnen. Begin bij het vaakst herhaalde, niet bij het moeilijkste; het moeilijkste is aantrekkelijk en het herhaalde is waar je tijd zit.

> **Wat dit goedkoper maakte:** *het bezwaar werd een vraag met een antwoord, en die vraag — hoe controleer ik of het klopt — is de vraag waar de rest van dit boek op gebouwd is.*

4. Feiten, geen meningen

Er is één gewoonte die ik uit ander werk heb meegenomen en die hier alles verklaart. Hij komt niet uit de software. Hij komt uit een kamer met een mens erin.

Ik heb jarenlang mensen begeleid naar werk — mensen zonder betaald werk, mensen met autisme, mensen voor wie de gewone route niet werkte. En het eerste wat je daar afleert, is oordelen.

Niet uit vriendelijkheid. Omdat het niet werkt.

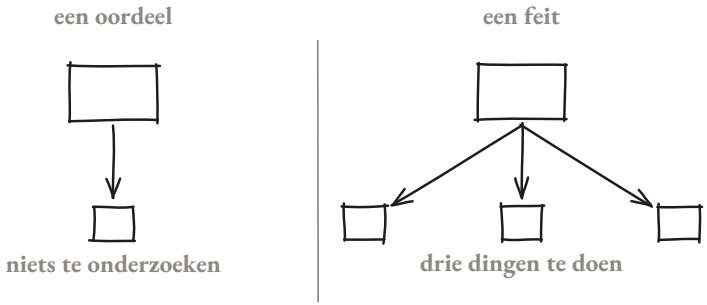
Wat een oordeel doet met de volgende stap

Zeg je *"hij is niet gemotiveerd"*, dan heb je iets gezegd waar niemand iets mee kan. Het is niet na te gaan, het is niet te weerleggen, en het geeft geen aanknopingspunt. Het enige wat het doet is de deur dichtdoen: als dat waar is, hoeft er niets meer onderzocht te worden.

Zeg je *"hij is de afgelopen drie keer op dinsdag niet gekomen, en op de andere dagen wel"*, dan staat er iets heel anders. Nu is er iets te doen. Wat is er op dinsdag anders? Het is te controleren, het kan onjuist blijken, en het wijst een kant op.

Dat is het verschil, en het is groter dan het lijkt. Een oordeel is een eindpunt. Een feit is een beginpunt.

En let op wat het met de ander doet. Een oordeel vraagt om instemming of verweer — je bent het ermee eens of je verdedigt jezelf. Een feit vraagt om een verklaring, en daar kun je samen aan werken.



Een oordeel sluit af. Een feit opent de volgende stap.

Vier gewoonten uit dat werk

Ze klonken destijds als geduld. Achteraf waren het werkwijzen.

Geen aannames vooraf. Een dossier vertelt je wat er eerder misging en vrijwel nooit waarom. Wie met dat dossier in zijn hoofd binnenkomt, ziet bevestiging van wat hij al dacht. Beginnen met niet weten is niet naïef; het is de enige manier om iets te zien dat er niet in stond.

Stap voor stap opbouwen. Niet omdat mensen weinig aankunnen, maar omdat een stap die mislukt, iets moet vertellen. Bij drie stappen tegelijk weet je bij een mislukking niets: je hebt drie kandidaten en geen antwoord.

Herhalen en stapelen. Wat één keer lukt, kan toeval zijn. Wat drie keer lukt, is een basis waar de volgende stap op mag staan. Zonder die herhaling bouw je op iets dat er misschien niet is.

Voorspelbaar maken. Dezelfde volgorde, dezelfde woorden, dezelfde tijd. Dat lijkt saai en het doet iets precies: het haalt alle ruis weg die niets met de taak te maken heeft. Wat er dan nog misgaat, gaat mis om een echte reden.

En dat werkt hier precies zo

Toen ik met AI ging werken, hoefde ik niets nieuws te leren. Ik deed wat ik al deed, en het werkte meteen — en dat verbaasde me meer dan het had moeten doen.

Een model heeft geen voorgeschiedenis met jou. Het kan niet raden wat je bedoelt. Het heeft geen gevoel voor wat er vorige week gebeurde, geen intuïtie over wat je waarschijnlijk wilt, en geen vermogen om tussen de regels door te lezen. **Alles wat je niet als feit levert, vult het zelf in** — en dan krijg je iets terug dat klopt met wat het zich heeft voorgesteld en niet met jouw situatie.

Leg de vier naast elkaar en het is dezelfde lijst:

in de begeleiding	**in dit werk**
geen aannames over wat iemand kan	geen aannames over wat het al weet — zet het erbij
één stap tegelijk, zodat een fout iets zegt	één wijziging tegelijk, anders weet je bij rood niets
herhalen tot het geen toeval meer is	drie keer laten draaien voordat je het een resultaat noemt
dezelfde volgorde, dezelfde woorden	dezelfde instructie, dezelfde vorm, dezelfde controle

"Het werkt niet goed" is hier precies zo nutteloos als "hij is niet gemotiveerd". Het is niet na te gaan en het wijst nergens heen. *"Bij*

deze invoer geeft hij dit, en ik verwachtte dat" is een feit, en daar begint het werk.

Waarom dit hoofdstuk vooraan staat

Omdat alles wat hierna komt erop leunt, en omdat het de goedkoopste verandering in dit boek is — hij kost niets, hij vraagt geen gereedschap, en je kunt er vanmiddag mee beginnen.

Een controle die zichzelf uitvoert, heeft een feit nodig om op te toetsen; een **subjectief** oordeel is niet te automatiseren. Een logboek van fouten is alleen bruikbaar als er staat wát er stond en niet wat iemand ervan vond. Een lus die zonder jou draait, kan alleen doorgaan als er onderweg iets controleerbaars ontstaat.

Een feit reist. Een mening niet. Dat is de hele reden dat dit werkt met iets dat geen geheugen aan jou heeft, geen context deelt, en niets aanvoelt.



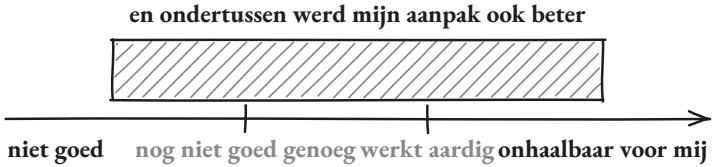
een mening komt hier niet voorbij

Een feit reist mee naar de volgende stap, een mening blijft achter.

Een oordeel over gereedschap heeft een houdbaarheidsdatum

Er is één soort oordeel dat bijna iedereen maakt en bijna niemand herziet, en het kost meer dan alle andere bij elkaar.

Maanden geleden probeerde ik iets waarvan mensen zeiden: *dit gaat niet.* En ze hadden gelijk. Ik probeerde het, het werkte niet, en dat was geen mening maar een waarneming. Toen ik het later nog eens probeerde, was het *nog niet goed genoeg*. Weken daarna *werkte het aardig*. En weer wat later, met de juiste patronen erbij: **hier kan ik zelf niet meer tegenop.**



Vier oordelen over hetzelfde, in een paar maanden tijd. En ondertussen werd mijn eigen aanpak ook beter.

Het patroon, en iedereen kent het op dit moment

Dit patroon zit op dit moment overal, en je hoort het wekelijks:

> *"Dat werkt niet. Ik heb het geprobeerd."*

Vraag er één ding bij en de zin valt uit elkaar: **wanneer?** Meestal blijkt het één keer te zijn geweest, een tijd geleden, zonder iets van wat er inmiddels bij hoort. Het was waar. Het is alleen niet meer waar, en er is niemand die dat komt melden.

Dat is de vorm, en hij is zo gewoon dat hij onzichtbaar is:

1. iemand probeert iets, één keer
2. het werkt niet — terecht
3. de uitkomst wordt een **oordeel** in plaats van een waarneming

4. het oordeel blijft, want er is geen reden om het te herzien
5. de wereld schuift eronder weg, en niemand merkt het

Stap drie is waar het misgaat, en het is precies de stap uit dit hoofdstuk: een waarneming werd een mening op het moment dat de datum eraf viel.

En het is niet dom. Het is de goedkoopste manier om met een wereld om te gaan die traag verandert. Alleen verandert deze dat niet.

Let op wat er in mijn eigen geval precies gebeurde, want er zijn twee dingen aan het bewegen en dat wordt vaak voor één ding aangezien. **Het gereedschap werd beter. En mijn manier van werken werd beter.** Die twee vermenigvuldigen elkaar, en daarom voelde de laatste stap als een sprong terwijl er niets sprong.

Dat gereedschap beter wordt, is niet nieuw. De eenheid wel

Hier hoort meteen een nuancering bij, want *wacht maar, het wordt beter* is geen inzicht — een timmerman, een drukker en een arts wisten dat allemaal al.

Het verschil zit in de **eenheid**. Vroeger was die eenheid een decennium, soms een generatie. Dan is "ik heb het geprobeerd en het werkte niet" een uitspraak die je de rest van je loopbaan kunt blijven gebruiken; hem herzien zou tijdverspilling zijn geweest.

Nu is die eenheid een paar maanden. **En hij is niet lineair**: elke stap is groter dan de vorige, omdat het gereedschap beter wordt én je eigen greep erop beter wordt. Daarmee is er iets stils veranderd:

> **De houdbaarheid van een oordeel is korter geworden dan de herinnering eraan.**

Je onthoudt "dit werkt niet" veel langer dan die zin waar blijft. Dat is geen karakterfout. Het is een gewoonte die honderd jaar lang verstandig was en het sinds kort niet meer is.

Wat je in plaats daarvan doet

Hier komt de regel van dit hoofdstuk terug, en nu met een tijdstempel erbij.

Leg vast dat het op moment X status Y had. Niet *"dit kan niet"*, maar *"dit is geprobeerd, zo, en dit kwam eruit."* Dat is een waarneming en die blijft waar — ook als het antwoord morgen verandert. Een oordeel zonder datum doet alsof het altijd geldt, en dat is precies de fout.

En dan stop je niet. Je zet het op de lijst en je meet opnieuw, op een vaste tijd, zonder erover na te denken of het zin heeft. Want de kans dat je uit jezelf aan het herzien slaat is klein: je weet immers al dat het niet werkt.

Laat het verschil systematisch opvallen, niet toevallig. Als je dezelfde proef elke keer hetzelfde draait, kun je de uitkomsten naast elkaar leggen en zien waar de lijn omslaat. Dat is dezelfde beweging als hierboven: tellen in plaats van vinden, alleen nu over de tijd. Wat je zoekt is het moment waarop iets dat niet kon, opeens wel kan — en dat wil je uit de metingen horen en niet van een gevoel.

En het gaat niet alleen over de machine. De helft van die verbetering zat bij mij. Wie alleen het gereedschap hertest en zijn eigen aanpak niet, meet de kleinste van de twee.

Probeer: schrijf vijf dingen op die je hebt geprobeerd en waarvan je hebt besloten dat het niet gaat, met erbij *wanneer* je het probeerde en *hoe*. Laat het daarna opnieuw doen, op precies dezelfde manier, en vraag waar de uitkomst afwijkt van wat er toen stond. Je gaat niet uit jezelf hertesten wat je hebt afgeschreven — je weet immers al dat het niet werkt. Let op dat het dezelfde proef is en niet een betere: verander je onderweg je aanpak, dan meet je twee dingen tegelijk.

Zelf doen · Laat een melding terugbrengen tot waarnemingen

Wat je doet. In `klachten/` staan vier meldingen zoals ze binnenkwamen. Drie ervan bevatten een oordeel — *te laat*, *niet verholpen*, *wijkt af* — en op een oordeel kun je niets bouwen. Laat ze terugbrengen tot wat er feitelijk staat, en leg die manier meteen vast zodat je hem niet elke keer opnieuw hoeft uit te leggen. Reken op tien minuten.

TYP DIT

```
In klachten/ staan meldingen van opdrachtgevers. Splits per melding
wat een waarneming is van wat een oordeel is, en zoek bij elke waarne-
ming de datum en het bestand waar hij uit komt. Schrijf die manier
daarna op als een herbruikbare werkwijze, met bovenaan één regel over
wanneer hij van toepassing is.
```

Wat je ziet. Vier meldingen uit elkaar getrokken, en daarna een bestand dat je niet zelf hebt geschreven. **De volgende keer dat er een klacht binnenkomt, biedt die werkwijze zich uit zichzelf aan** — dat is het verschil met een afspraak die je op een goede dag nog weet.

WAT ER TERUGKWAM

```
2026-04-03-van-loon
  melding : Werk begon twee dagen later dan
            afgesproken.
  feit    : NLOP-2026-006 op 2026-03-28
  oordeel : "later dan"
werkwijze aangemaakt: melding-omzetten
```

Waar het misgaat. Laat het oordeel staan in plaats van het weg te gooien. *"Te laat"* is geen feit maar wel informatie: het zegt wat de klant verwachtte. Gooi je het weg, dan is de klacht onbegrijpelijk geworden. **Laat onbevestigde dingen ook staan, als onbevestigd.** Een waarneming die je niet in een bestand terugvindt is niet onwaar — hij is alleen niet nagekeken, en dat is iets anders. **En beschrijf in die ene regel wanneer de werkwijze van toepassing is, niet wat hij is.** *"Over klachten"* wordt nooit gevonden; *"als er een klacht of een melding binnenkomt"* wel.

EN DAN ZO

Zet in de werkwijze ook wat er met een onbevestigde waarneming moet gebeuren, en laat me de beschrijvingsregel zien die je bovenaan hebt gezet.

EN DAN DIT

```
melding-omzetten
  "Als er een klacht of een melding binnenkomt --
   brengt hem terug tot waarnemingen met een datum
   en een bron, voordat er iemand over oordeelt."
  stap 4: wat niet te herleiden is blijft staan
          als onbevestigd, niet weglaten
```

En het geldt ook de andere kant op

Dit is geen eenrichtingsregel, en dat deel wordt vergeten.

Wat je terugkrijgt, is ook vaak een mening in de vorm van een feit. *"Dit is de beste aanpak." "Dit werkt niet." "Ziet er goed uit."* Dat zijn oordelen, en ze zijn precies zo weinig waard als die van jou — met het extra ongemak dat ze zelfverzekerd klinken.

De vraag die je terugstelt is dezelfde die je in die kamer zou stellen: **waaraan zie je dat?** Niet als wantrouwen, maar omdat het antwoord bruikbaar is en het oordeel niet.

Probeer: laat je probleembeschrijving eerst ontleden voordat je hem voorlegt — welke zinnen zijn een feit en welke een interpretatie, en wat is er nodig om die interpretaties te vervangen. Je merkt zelf niet meer welke van je zinnen een oordeel zijn; ze voelen als beschrijvingen. Doe het vóór het werk: achteraf verdedig je je beschrijving in plaats van hem te verbeteren.

Dat was de ene richting. De andere wordt vaker overgeslagen en is even belangrijk: wat je terugkrijgt is net zo goed van oordelen te ontdoen.

En andersom: krijg je een oordeel terug — *"dit is de beste aanpak"* — vraag dan waaraan je dat ziet, en vraag uitdrukkelijk om het antwoord *"die waarneming is er niet"*. Zonder dat verzint hij er een bij, en dan sta je slechter dan met het oordeel.

> **Wat dit goedkoper maakte:** *elk feit dat ik opschreef in plaats van een indruk, kon door iemand anders gebruikt worden — en later door iets anders dan iemand.*

5. Bereikbaarheid boven waarde

Er is een zet die bijna altijd de verkeerde blijkt, en het is de zet met de hoogste opbrengst.

Niet omdat de som fout is. De som klopt meestal prima. Het probleem zit een laag eronder: bijna elke zet met een hoge opbrengst per eenheid rust op een voorwaarde die niemand opschrijft, en als die voorwaarde er niet is, is de opbrengst geen tegenvaller maar **nul**.

De vraag die je eerst stelt

Kijk niet naar het bedrag maar naar de voorwaarde.

De zet met de hoogste waarde per eenheid is vaak niet bereikbaar, want er ontbreekt iets onderaan. En de bereikbare zet is dan niet zomaar de kleinere zet — het is de zet die die voorwaarde máákt.

Dat is wat dit hoofdstuk beschrijft, en het heeft een naam nodig omdat het vaak tegen het gevoel in gaat: **bereikbaarheid boven waarde**. Niet de zet kiezen die het meeste oplevert, maar de zet die je werkelijk kunt doen — en van die zetten degene die een ontbrekende voorwaarde invult.

Dat klinkt als een redenering die je achteraf overal op kunt plakken. Dus hier is hij, uitgerekend, aan iets dat iedereen kan narekenen.

Vier keer niets is nog steeds niets

Neem het uitgeven van dit boek.

Wie zonder uitgever publiceert, kan twee kanten op, en de platforms rekenen het keurig voor. Een papieren boek van € 14,99, verkocht via een boekhandel, levert de schrijver **€ 0,87** op. Datzelfde boek rechtstreeks verkocht levert **€ 3,47** op. Vier keer zoveel.

Vier keer zoveel is een groot verschil, en het is de verkeerde grootheid.

Want € 3,47 is alleen meer dan € 0,87 als er iemand komt kopen. Bij rechtstreeks verkopen hoort een voorwaarde die in dat rekensommetje niet voorkomt: **een publiek.** Mensen die weten dat het boek bestaat en die weten waar ze moeten zijn. Een nieuwsbrief, volgers, een plek waar je al gehoord wordt.

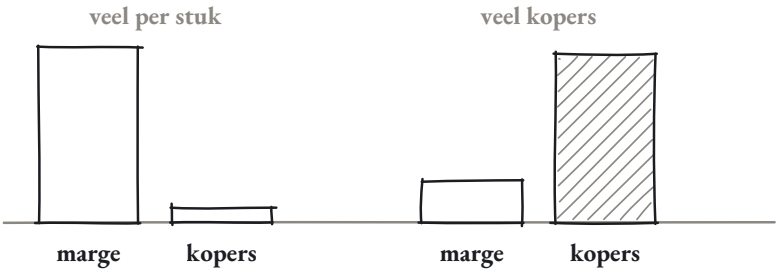
Voor een eerste boek van iemand zonder podium is dat aantal nul. En de hogere marge geldt dan voor een bezoekersaantal van nul.

> **Vier keer niets is nog steeds niets.**

Zet er de andere kant naast en de som draait om. Vijfduizend keer twee euro is tienduizend euro; tien keer tien euro is honderd. Er zit een factor honderd tussen, en dat verschil zit niet in de marge maar in het **aantal**.

Wat een tussenpartij levert is dan ook geen marge maar bereik. Eén handeling zet het boek in de grote online winkels en bij boekhandels in zes landen. In Nederland loopt ongeveer zeventig procent van de online boekverkoop via één winkel; daarin staan is de voorwaarde, en die voorwaarde kost één upload en € 12,75 voor een ISBN.

En er is een tweede reden, die misschien zwaarder weegt dan de eerste. Elk kanaal dat je zelf aansluit is een account, een belastingformulier, een bestandsformaat en een plek waar iets kan breken terwijl je aan iets anders bezig bent. Dat is geen eenmalige prijs maar een abonnement op aandacht — en aandacht is precies wat er niet over is als je alleen werkt.



Vier keer de marge op nul lezers is nog steeds nul.

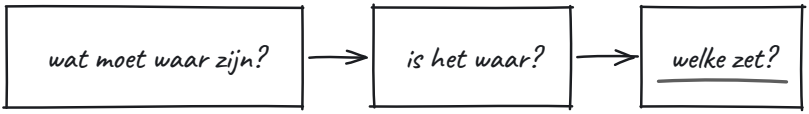
Het mechanisme eronder

Er bestond een zet met een hogere opbrengst per eenheid. Die zet was niet bereikbaar, omdat er een voorwaarde onder lag die niet vervuld was: een publiek dat er nog niet was. En de bereikbare zet was niet zomaar de kleinere zet, maar **de zet die die voorwaarde maakt**.

Dat is het hele mechanisme, en het is te herkennen aan drie vragen die je in die volgorde stelt:

1. Wat moet er waar zijn voordat deze zet iets oplevert?
2. Is dat vandaag waar?
3. Zo niet: welke zet die ik vandaag wél kan doen, maakt het waar?

De eerste vraag is degene die overgeslagen wordt. Bijna niemand rekent de opbrengst van een plan verkeerd uit; bijna iedereen vergeet te noteren welke voorwaarde er onder de som ligt. Een zet die een ontbrekende voorwaarde veronderstelt, levert niet minder op dan gehoopt — hij levert nul op. Dat is een ander soort fout dan een tegenvaller, en hij is niet te herstellen door harder te werken.



de eerste vraag is degene die overgeslagen wordt

De drie vragen, in de volgorde waarin ze gesteld moeten worden.

Zelf doen · Laat uitrekenen waar je aan mee kúnt doen

Wat je doet. In `aanbestedingen/` staan vier uitvragen en in `certificaten/bedrijfscertificaten.md` staat wat het bedrijf heeft. De grootste uitvraag is niet vanzelf de beste: hij is pas iets waard als je eraan mag meedoen. Laat dat naast elkaar zetten. Reken op vijf minuten.

TYP DIT

> Zet per aanbesteding naast elkaar: de omvang, wat er vereist is, en of wij dat volgens certificaten/bedrijfscertificaten.md hebben. Zeg per stuk of we kunnen inschrijven, en zo niet, wat er precies ontbreekt.

Wat je ziet. Vier regels. Twee kunnen, twee niet — en de twee die niet kunnen zijn samen goed voor bijna een miljoen. **Ze vallen niet af op prijs of op drukte maar op één ontbrekend certificaat.** Dat is geen tegenvaller maar een voorwaarde: hij kost maanden en hij is te halen, alleen niet vóór de sluitingsdatum van deze twee.

WAT ER TERUGKWAM

```
NLAB-2026-01  40.000/jr, 3 jaar   kan inschrijven
NLAB-2026-02  circa 320.000       kan inschrijven
NLAB-2026-03  85.000 per jaar     MIST: ISO 45001
NLAB-2026-04  circa 900.000       MIST: ISO 45001
```

Waar het misgaat. Laat hem matchen op de hele naam en niet op het eerste woord. Hier ging dat mis: bij *ISO 9001* keek hij naar "ISO", zag dat *ISO 45001* ontbrak, en meldde dat we ISO 9001 niet hadden — terwijl dat er gewoon is. Twee van de vier regels waren daardoor fout, en het zag er volstrekt geloofwaardig uit. **Vraag om wat er precies ontbreekt**, niet alleen om ja of nee: zonder die naam weet je niet of het een maand of een jaar kost. **En kijk zelf één regel na.** Dit is een lijst waarop je een besluit neemt, en de goedkoopste controle is er één met de hand.

EN DAN ZO

> Vergelijk de certificaatnamen exact in plaats van op het eerste woord, en laat me per aanbesteding zien welke namen je met elkaar hebt vergeleken.

EN DAN DIT

```
NLAB-2026-01  vereist VCA**, ISO 9001 -> beide aanwezig
NLAB-2026-02  vereist VCA**, garantieregeling -> aanwezig
NLAB-2026-03  vereist VCA**, ISO 45001 -> ISO 45001 niet
NLAB-2026-04  vereist VCA**, ISO 9001, ISO 45001
           -> alleen ISO 45001 ontbreekt
```

Waarom dit niet "kies de makkelijke weg" is

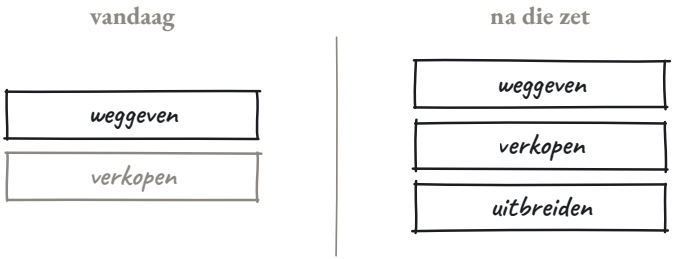
Hier gaat het mis als je het te snel overneemt, dus het hoort erbij.

Bereikbaar is niet hetzelfde als makkelijk. De bereikbare zet telt alleen als hij de ontbrekende voorwaarde invult. Iets weggeven is bereikbaar,

maar ergens anders gaan zitten koffiedrinken is dat ook. Het verschil is of er na afloop iets waar is dat daarvoor niet waar was.

De toets is een zin. Kun je in één zin zeggen welke voorwaarde er door deze zet ontstaat, én waaraan je zou zien dat hij er is, dan is het de zet. Kun je dat niet, dan heb je de makkelijke zet gekozen en er een verhaal bij bedacht.

En let op wat dit zegt over volgorde. Bereikbaarheid is geen eigenschap van een zet maar van een moment. De rechtstreekse verkoop met vier keer de marge is niet afgeschreven — hij staat in de wacht. Als het boek in de winkels ligt, komen er lezers; als er lezers zijn, is er een publiek; en op de dag dat dat publiek er is, is dezelfde zet die vandaag niets oplevert ineens de beste die er is. **Dat is wat "de volgende zet goedkoper maken" concreet betekent: niet dat je minder betaalt, maar dat er zetten bijkomen die er eerst niet waren.**



Een zet die een voorwaarde invult, maakt zetten mogelijk die er eerst niet waren.

En het geldt ook waar er helemaal geen geld in het spel is. In het eerste hoofdstuk stond er aan het eind van de avond gereedschap op schijf en geen product. Dat gereedschap leverde die avond niets op; het vulde een

voorwaarde in — dat controleren gratis wordt — en alles wat daarna kwam, hing daarvan af.

Probeer: schrijf je volgende drie zetten op, en zet er niet bij wat ze opleveren maar wat er waar moet zijn vóórdat ze iets opleveren. Streep door wat een voorwaarde mist die je vandaag niet hebt; wat overblijft is jouw zet. Duurt het langer dan tien minuten, dan zijn de zetten te groot opgeschreven.

Dit is niet *kies de makkelijkste zet*. De bereikbare zet telt alleen als hij de ontbrekende voorwaarde invult. Kun je niet in één zin zeggen welke voorwaarde erdoor ontstaat en waaraan je zou zien dat hij er is, dan heb je gewoon het makkelijke gekozen.

> **Wat dit goedkoper maakte:** *elke volgende lezer en elke volgende gebruiker kwam binnen zonder dat er iemand overtuigd hoefde te worden.*

6. Multipliers

Er zijn twee soorten verbeteringen en ze worden bijna altijd door elkaar gehaald.

De eerste bespaart tijd. Je doet iets slimmer en het kost voortaan twintig minuten minder. Doe je het vijftig keer, dan heb je vijftig keer twintig minuten gewonnen. Dat is echt en het is niet weinig.

De tweede doet iets anders. Die maakt niet één ding sneller maar **alles wat erna komt**. Hij telt niet op bij wat je al had; hij vermenigvuldigt het.

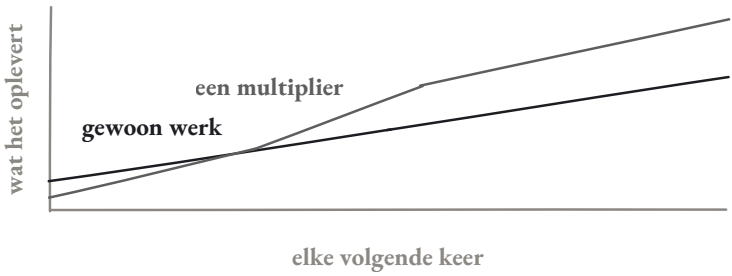
Een besparing is een korting. Een multiplier is een wisselkoers.

Waarom het verschil ertoe doet

Omdat ze op de dag zelf niet uit elkaar te houden zijn, en omdat de tweede bijna altijd de minst aantrekkelijke van de twee lijkt.

Een besparing levert meteen iets op. Je ziet hem, je kunt hem uitleggen, en aan het eind van de dag is er iets veranderd. Een multiplier levert die dag niets op. Hij is moeilijk te verantwoorden, want je hebt niets te laten zien behalve een belofte over morgen.

En dus kiest bijna iedereen, bijna altijd, de besparing. Vijftig keer twintig minuten is zeventien uur, en dat is een prima dag werk. Alleen sta je aan het eind van die vijftig keer nog steeds op precies dezelfde plek.



Gewoon werk levert een keer op. Een multiplier verlaagt de vloer voor alles wat erna komt.

De eerste die ik er een zag zijn

Lang voordat ik dit woord kende, heb ik er een gebouwd, en het had niets met computers te maken.

Op het callcenter uit hoofdstuk 3 kon ik twee dingen doen met de tijd tussen twee gesprekken. Ik kon een gesprek extra voeren — dat is de besparing, en hij telt meteen op. Of ik kon opschrijven wat er in het vorige gesprek gebeurde en het script aanpassen.

Het tweede leverde die dag **niets** op. Minder gesprekken, minder omzet, en niets om te laten zien. Maar een zin die beter werkte, werkte de volgende dag ook beter, en de dag daarna, en bij iedereen die het script gebruikte. Eén middag nadenken reisde mee in elk gesprek dat ik daarna voerde.

Dat is het hele verschil, en het is ouder dan de techniek. Het extra gesprek is een korting. De aangepaste zin is een wisselkoers.

Het instrument was een kladblaadje.

Hoe je er een herkent

De vraag is niet "wat levert dit op" maar "wat wordt hierna goedkoper".

Een multiplier heeft drie eigenschappen, en alle drie zijn ze nodig:

1. **Hij werkt op alles wat volgt, niet op één geval.** Niet dit contract sneller, maar elk contract vanaf nu.
2. **Hij vraagt geen aandacht meer nadat hij er staat.** Iets dat je elke keer opnieuw moet aanzetten, bespaart misschien tijd maar vermenigvuldigt niets — want jouw aandacht is de schaarse grootheid en die groeit niet mee.
3. **Hij is op de dag zelf verlies.** Dat klinkt als een grap en het is de betrouwbaarste toets die er is. Als het vandaag al iets oplevert, is het meestal een besparing.

De negen die het verschil maakten

Toen ik achteraf ging kijken waar het tempo vandaan kwam, bleven er negen zetten over. Geen ervan was slim. Ze waren alle negen saai, en ze kostten alle negen een halve dag die niets opleverde.

de zet	wat hij vermenigvuldigde
een testsuite	controleren wordt gratis
een ticketsysteem	werk overleeft de sessie
nul afhankelijkheden	geen verloren middag
gereedschap dat zichzelf bouwt	elke verbetering verbetert het gereedschap
elf milliseconden	tien keer proberen mag
een foutenlogboek	dezelfde fout kost minuten
één gedeeld model	één keer schrijven is drie keer hebben
bruikbare foutmeldingen	de lus draait zonder toezicht
het gereedschap gratis	adoptie is geen verkoopgesprek

Negen multipliers, en wat elk van hen goedkoper maakte.

Lees die lijst van boven naar beneden en je ziet iets gebeuren. Hij valt in drie groepen uiteen, en die groepen zijn een volgorde.

De eerste drie maken één persoon sneller. Een testsuite, een plek waar werk blijft staan, en geen afhankelijkheid van andermans versie. Dit is het soort zet waarvan iedereen begrijpt waarom je hem doet, en die je toch uitstelt omdat er die dag niets van komt.

De middelste drie maken een machine bruikbaar. Gereedschap dat zichzelf bouwt, een opstarttijd waarbij tien keer proberen niets kost, en een logboek waardoor dezelfde fout de tweede keer minuten kost. Let op wat er hier gebeurt: deze drie gaan niet over jou. Ze gaan over of er íets anders dan jij aan het werk kan.

De laatste drie laten iets doorlopen zonder dat er iemand kijkt. Eén model in plaats van drie, foutmeldingen waar een machine iets mee kan, en het gereedschap weggeven in plaats van verkopen.

Dat is wat stapelen betekent, en het is de reden dat de volgorde niet vrij is: **elke groep heeft de vorige nodig.** Een machine bruikbaar maken heeft geen zin als controleren nog duur is, en iets laten doorlopen zonder toezicht heeft geen zin als de machine niet bruikbaar is. Niet harder draaien dus, maar **een soort werk mogelijk maken dat er daarvoor niet was** — en dat kan alleen in deze volgorde.

De twee manieren waarop het misgaat

Wie eenmaal doorheeft wat een multiplier is, wil er alleen nog maar bouwen. Dat is ook fout, en op twee manieren.

De eerste: een multiplier die jouw aandacht blijft vragen, is er geen. Dit is eigenschap twee van hierboven en hij wordt het vaakst overtreden. Een script dat je elke ochtend zelf moet starten, een lijst die iemand moet bijwerken, een controle die je moet onthouden — die besparen allemaal tijd en ze vermenigvuldigen niets, want aandacht is de schaarse grootheid en die groeit niet mee. **Tel niet de minuten die het bespaart maar de keren dat het jou nodig heeft.** Is dat getal groter dan nul, dan heb je een gereedschap gemaakt en geen multiplier.

De tweede: alleen maar multipliers bouwen is hetzelfde als er geen bouwen. Er moet ook iets af. Een halfjaar aan gereedschap zonder werk eronder is geen wisselkoers maar een hobby, en je merkt het pas als er niets is om mee te vermenigvuldigen. De verhouding die bij mij werkt is simpel: **een multiplier bouw je pas als je het werk dat hij goedkoper maakt al minstens drie keer met de hand hebt gedaan.** Daarvoor weet je niet wat je bouwt.

Dat is dezelfde eerste stap als in hoofdstuk 3: zelf doen, handen vuil maken. Wie iets automatiseert dat hij nooit zelf heeft gedaan, automatiseert zijn aanname.

Zelf doen · Bundel je losse controles tot één werkwijze

Wat je doet. Je hebt nu drie keer hetzelfde gedaan: een regel gekozen, een controle laten schrijven, hem gedraaid. Dat is drie keer het werk voor drie regels, terwijl er twaalf zijn. Laat er één werkwijze van maken die ze allemaal afloopt — en laat er meteen een tweede naast zetten die kijkt of die ronde zelf volledig was. Reken op vijftien minuten.

TYP DIT

> In regels.md staan twaalf regels. Schrijf een herbruikbare werkwijze die ze alle twaalf afloopt tegen de mappen waar ze over gaan, en alleen de overtredingen meldt met het nummer erbij. Maak daarnaast een tweede, losse werkwijze die controleert of de ronde volledig was, en laat de eerste naar de tweede verwijzen. Draai de eerste daarna en laat me de uitkomst zien.

Wat je ziet. Twee werkwijzen, en een uitkomst die je met de hand nooit had samengesteld: **vijftien overtredingen verdeeld over tien van de twaalf regels.** De regel over vakbekwaamheid springt eruit met vier, en dat is geen toeval — daar zit een certificaat achter dat op één dag afliep en vier opdrachten tegelijk raakte. **Dat je twee regels zonder overtreding ziet, is óók informatie**: die zijn gedraaid en schoon, en dat is iets anders dan niet gedraaid.

WAT ER TERUGKWAM

```
regel  1  1   NLOP-2026-005 ontbreekt
regel  2  1   opdrachtgever bestaat niet
regel  4  1   meerwerk 3950 zonder akkoord
regel  5  4   vakbekwaamheid
regel  7  1   onderaannemer zonder G-rekening
regel  8  2   agenda
regel  9  2   keuring verlopen
--- 15 overtredingen over 10 regels ---
```

Waar het misgaat. Laat hem ook de regels noemen die schoon zijn. Anders weet je niet of regel 10 klopte of overgeslagen werd, en dat verschil is precies waar een controle zijn waarde verliest. **De tweede werkwijze hoort er echt naast en niet in.** Een ronde die zichzelf volledig verklaart, verklaart zichzelf altijd volledig. **En let op de regels die van een datum afhangen.** Regel 5 en 9 moeten gemeten worden tegen de dag waarop het werk gebeurde, niet tegen vandaag — wie tegen vandaag meet, krijgt een ander antwoord zodra hij de controle volgend jaar draait.

EN DAN ZO

> Laat ook de regels zien die nul overtredingen gaven, zodat ik kan zien dat ze gedraaid zijn.

EN DAN DIT

```
regel  6  0   V&G-plannen actueel
regel 10  0   termijnfacturen binnen 14 dagen
regel 11  0   bouwmelding voor de start
regel 12  1   marktsignaal zonder datum
--- 12 van de 12 regels gedraaid ---
```

En dit is niet van software

De lijst hierboven komt uit een technisch project omdat dat het project was. Elk van de negen heeft een tegenhanger in werk dat niets met com-

puters te maken heeft, en meestal is die tegenhanger ouder dan de mijne.

Een aannemer met een vaste controlelijst voor een oplevering heeft de eerste. Een praktijk die noteert wat er met een patiënt is afgesproken heeft de tweede. Een bakker die zijn eigen deeg maakt heeft de derde. Een fotograaf die zijn bewerking vastlegt heeft de vierde. Een restaurantkeuken waar een gerecht binnen de minuut van de pas gaat, heeft de vijfde — en dat is de reden dat er een tweede zaak kán komen.

Wat ze allemaal delen: op de dag dat ze gemaakt werden, leverden ze niets op.

Probeer: streep in je agenda van vorige week alles weg wat je precies zo nog een keer gaat doen. Wat overblijft is het werk dat zich herhaalt — en dat is het enige werk waar een multiplier op kan aangrijpen. Kies er één uit en vraag je af wat er zou moeten bestaan om het de volgende keer half zo duur te maken. Dát bouw je, en niet de taak zelf. Begin bij de taak die het vaakst terugkomt, ook als hij klein is: vermenigvuldigen werkt op het aantal, niet op de omvang.

Heb je er een gevonden, dan komt de vraag die bepaalt of het er ook echt een is. Niet wat hij oplevert, maar of hij je daarna met rust laat.

En toets daarna of het er echt een is. Laat voor iets dat je net gebouwd hebt opsommen op welke momenten het jou nog nodig heeft — starten, bijwerken, opruimen, controleren, beslissen — en tel ze. Een oplossing die tijd bespaart maar aandacht blijft vragen, schaalt niet: jouw aandacht is de enige grootheid die niet meegroeit. Nul is vaak niet haalbaar; van vijf naar één is al een andere orde, als je maar opschrijft wélke ene het is.

Wat dit goedkoper maakte: elke volgende dag. Dat is letterlijk bedoeld en het is het enige wat een multiplier van een besparing onderscheidt.

7. En als je het duizend keer deed?

Twee manieren om hetzelfde te doen, en op de dag zelf zijn ze niet uit elkaar te houden.

In de eerste kijk je naar het resultaat en zegt: dit is goed. In de tweede schrijf je op waaraan je zag dat het goed was, en de volgende keer kijkt iemand anders daarnaar — of iets anders.

Eén keer kost de eerste manier minder tijd. Hij kost altijd minder tijd. En bij duizend keer is de eerste manier niet duurder maar **onmogelijk**, want er is maar één van jou.

Dat is het hele hoofdstuk: **stel bij elke keuze de vraag hoe het eruitziet bij duizend keer, ook als je er maar één doet.** Niet omdat je er duizend gaat doen. Omdat het antwoord je iets anders laat kiezen.

De vraag heeft drie delen, en het derde is de lastige

Werkt het nog? Sommige dingen klappen om bij aantal. Twee dingen die dezelfde plek willen, gaan goed tot ze tegelijk komen. Een naam die uniek leek, botst. Een volgorde die vanzelf klopte omdat er één van was, klopt niet meer. Deze categorie is de bekendste en meestal de goedkoopste om te herstellen.

Is het nog te betalen? Vermenigvuldig de kosten van één keer met duizend en kijk of je schrikt. Een euro per keer is duizend euro. Vijf minu-

ten per keer is drie werkdagen. Dit is rekenwerk en het duurt tien seconden, en bijna niemand doet het.

En heeft het mij nog nodig? Dit is de vraag die alles beslist. Als elk van die duizend keer één minuut van jouw aandacht vraagt, heb je geen systeem — je hebt zeventien uur werk en een illusie. Het aantal mensen groeit niet mee, en dat is niet iets wat je later oplost.

De eerste twee zijn techniek. De derde is een ontwerpvraag, en het antwoord erop bepaalt of je iets bouwt dat kan groeien of iets dat groot wordt en dan stilvalt.



De vraag heeft drie delen, en het derde wordt bijna altijd overgeslagen.

Ik heb die drie vragen ooit in die volgorde beantwoord

Op het callcenter deed ik iets honderden keren per week. Dat is geen duizend, maar het is ver genoeg voorbij één om het verschil te voelen — en het dwong me de drie vragen te beantwoorden zonder dat ik wist dat het vragen waren.

Werkt het nog? Ja. Een gesprek is een gesprek, en het honderdste ging niet stuk omdat het het honderdste was.

Is het nog te betalen? Ook ja, en dat was precies het probleem: het kostte elke keer evenveel. Honderd gesprekken is honderd keer de prijs van één. Er zat geen korting in.

Heeft het mij nog nodig? En daar liep het vast. Alles wat werkte, zat in mijn hoofd. Ik wist welke zin het deed, ik wist wanneer je moest zwijgen, en ik kon het niet aan iemand anders geven zonder erbij te gaan zitten. Ik was honderd keer nodig.

Dat is wat het opschrijven veranderde. Niet de gesprekken — die werden er niet beter van op de dag zelf. Wat veranderde, is dat het antwoord op de derde vraag van *ja* naar *minder* ging: er lag een script, er lag een stappenplan, en het saaie deel liep vanzelf. Pas daarna kon er iets bestaan dat groter was dan mijn week.

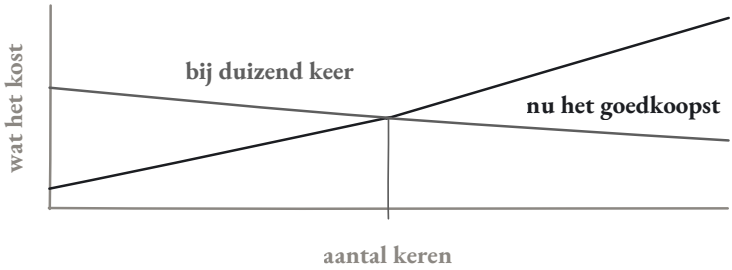
Let op de volgorde, want die is niet toevallig. De eerste twee vragen beantwoordde ik door te kijken. De derde beantwoordde ik door iets te bouwen. Zo is het altijd: de eerste twee zijn rekenwerk, de derde is werk.

En dit gaat over gewoon werk

Een restaurant met één kok kan een gerecht hebben dat afhangt van zijn oordeel. Dat gerecht kan uitstekend zijn. Het kan alleen nooit in een tweede zaak, en ook niet op zijn vrije dag. Een gerecht met een opgeschreven maat en een controlepunt is misschien iets minder — en het bestaat ook als hij er niet is.

Een docent die tweehonderd opstellen nakijkt op gevoel, doet tweehonderd keer hetzelfde werk. Een docent die eerst opschrijft waaraan een goed opstel te herkennen is, doet dat werk één keer en daarna iets anders — en krijgt er bovendien iets voor terug wat het eerste nooit geeft: hij kan uitleggen waarom.

Een adviseur die alles in zijn hoofd heeft, kan nooit twee opdrachten tegelijk en kan niets overdragen. Dat is geen karakterfout; het is een ontwerpkeuze die hij ooit zonder erbij na te denken heeft gemaakt.



Wat bij één keer de goedkoopste keuze is, is bij duizend keer de duurste.

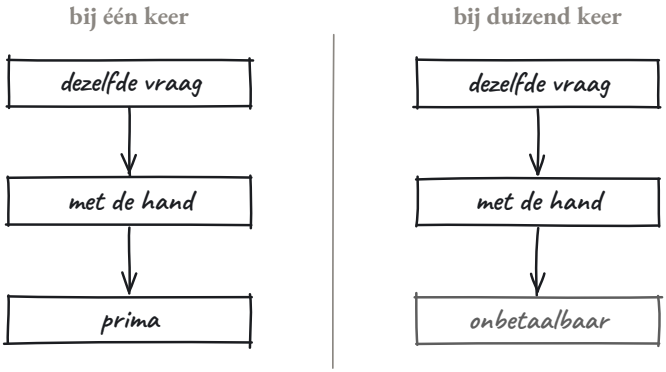
Je komt tot andere besluiten

Dat is het punt waar dit hoofdstuk om draait, en het is groter dan het klinkt. De vraag is niet een controle achteraf. **Hij verandert wat je kiest.**

bij één keer kies je	**bij duizend keer kies je**
kijken of het goed is	opschrijven waaraan je dat ziet
onthouden wat je deed	het vastleggen waar het terug te vinden is
het even zelf doen	zorgen dat het zonder jou kan
"dat merken we wel"	een grens die vanzelf ingrijpt
de beste oplossing	de oplossing die niemand hoeft uit te leggen

Kijk naar de rechterkolom. Dat zijn stuk voor stuk keuzes die op de dag zelf **meer** werk zijn. Het is dezelfde vorm als het eerste hoofdstuk: de zet die telt, is de zet die vandaag niets oplevert.

Het verschil is dat je hier een reden hebt om hem toch te nemen. Niet "het is netter", niet "dat hoort zo", maar: **anders kan het nooit meer dan één.**



Dezelfde beslissing, twee keer genomen: een keer voor nu en een keer voor duizend keer.

En dan nog eens

Wat duizend overleeft, overleeft niet vanzelf duizend keer duizend. Elke sprong breekt iets anders.

Bij tien merk je dat je namen nodig hebt. Bij honderd merk je dat je moet kunnen zien wat er speelt zonder alles te lezen. Bij duizend merk je dat je moet kunnen ingrijpen zonder alles stil te zetten. Bij tienduizend merk je dat het geld op kan zonder dat iemand het zag aankomen.

Dus stel de vraag steeds opnieuw, bij elke stap, en niet één keer aan het begin. Het is geen poortje waar je doorheen moet; het is een gewoonte.

Zelf doen · Laat hem alleen melden wat er veranderd is

Wat je doet. Je ronde uit het vorige hoofdstuk meldt vijftien overtredingen. Bij vijftien opdrachten is dat te overzien. Bij vijftienhonderd niet, en dan gebeurt er iets vervelends: je gaat de lijst niet meer lezen. De vraag van dit hoofdstuk is wat er verandert als je het duizend keer doet — en het antwoord is dat de vorm van de uitkomst moet veranderen, niet de controle. Reken op tien minuten.

TYP DIT

> Pas de ronde aan zodat hij de bevindingen van de vorige keer onthoudt, en meld voortaan drie dingen apart: wat er nieuw bij is gekomen, wat er is opgelost, en hoeveel er ongewijzigd openstaan. Draai hem twee keer: nu, en nadat ik iets heb veranderd.

Wat je ziet. De eerste ronde kan niets vergelijken en zegt dat ook — vijftien bevindingen, geen vergelijking mogelijk. **Daarna is de uitkomst opeens twee regels in plaats van vijftien**, en het zijn de twee regels waar je iets mee moet. **De ongewijzigde vijftien verdwijnen niet**, ze worden geteld. Dat is het verschil tussen iets negeren en iets erkennen.

WAT ER TERUGKWAM

```
eerste ronde: 15 bevindingen
  (geen vergelijking mogelijk)

na een wijziging:
nieuw sinds de vorige ronde   : 1
   + 3: NLOP-2026-013
opgelost sinds de vorige ronde: 0
ongewijzigd                   : 15
```

Waar het misgaat. Laat het aantal ongewijzigde staan. Haal je dat weg, dan lijkt er niets aan de hand zolang er niets nieuws bijkomt — en dan is een probleem dat drie maanden blijft liggen onzichtbaar geworden. **Zorg dat de eerste ronde zegt dat hij niets kan vergelijken.**

Zonder die regel ziet een lege eerste uitkomst er hetzelfde uit als een schone administratie, en dat is precies de fout die dit boek in hoofdstuk 9 behandelt. **En bewaar het geheugen ergens waar je het terugvindt.** Een ronde die zijn vorige uitkomst kwijt is, begint stilletjes weer bij nul en meldt dan alles als nieuw.

EN DAN ZO

> Laat ook zien hoe lang elke ongewijzigde bevinding al openstaat, in dagen.

EN DAN DIT

```
ongewijzigd (15), met hoe lang ze al openstaan:
  1: NLOP-2026-005 ontbreekt        sinds ronde 1
  2: NLOP-2026-006                  sinds ronde 1
  4: NLMW-2026-06                   sinds ronde 1
  9: rolsteiger-B                 sinds ronde 1
  (de teller loopt pas vanaf de eerste ronde,
   niet vanaf het ontstaan)
```

Waarom juist nu

Deze vraag is niet nieuw. Wat nieuw is, is dat het antwoord ertoe doet.

Vroeger betekende duizend keer: duizend mensen inhuren, of een fabriek bouwen. Voor vrijwel iedereen was de vraag daarom theoretisch — je zou het toch nooit duizend keer doen, dus waarom zou je erop ontwerpen.

Dat is weg. Duizend keer is nu een kwestie van aanzetten.

En daar zit precies het verschil tussen een gereedschap en een organisatie. Een gereedschap maakt jou sneller; je blijft ernaast staan en het wacht op jou. Een organisatie doet dingen zonder dat jij erbij bent.

De sprong daartussen is niet een betere machine — het is een reeks keuzes die je vroeg moet maken en die je later niet meer kunt inbouwen.

Wie zijn werk heeft ingericht op één keer, zit vast op één keer, hoe goed het gereedschap ook wordt. Wie de vraag heeft gesteld, kan hem gewoon opendraaien.

Probeer: neem de eerstvolgende beslissing die je toch al neemt, hoe klein ook, en stel er drie vragen bij — werkt dit nog bij duizend keer, is het dan nog te betalen, en heeft het mij dan nog nodig. Schrijf alleen het derde antwoord op: de eerste twee zijn rekenwerk dat je later corrigeert, het derde is een ontwerpkeuze die je later niet meer verandert zonder alles opnieuw te doen. De vraag gaat over de vórm, niet over de omvang — je hoeft niets voor duizend te bouwen, je moet alleen niet iets kiezen dat duizend voorgoed uitsluit.

Wat dit goedkoper maakte: alles wat later opgeschaald moest worden en dat niet hoefde te worden herbouwd, omdat de vorm het al toestond.

8. Waarom doe je dat nog zelf?

Het is geen verwijt. Het is een vraag die je jezelf stelt terwijl je ergens middenin zit, en meestal is het antwoord de eerste keer nog goed.

Omdat ik het sneller doe dan dat ik het uitleg. Vaak waar. *Omdat het maar twee minuten kost.* Ook waar. *Omdat er oordeel bij komt kijken.* Soms waar.

Wat deze vraag bijzonder maakt is niet dat hij het antwoord verandert. Het is dat **hij dezelfde vraag blijft terwijl het antwoord onder je voeten verschuift.** Wat vorige maand niet kon, kan nu. En je merkt dat niet, want je bent bezig.

Waarom het antwoord verschuift zonder dat je het ziet

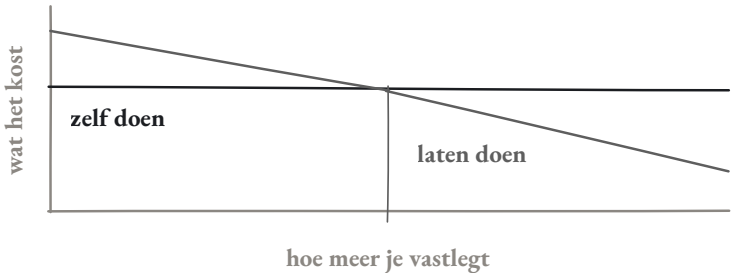
Je hebt op enig moment besloten dat iets jouw werk was. Dat besluit was goed — toen. Daarna is het een gewoonte geworden, en een gewoonte legt geen verantwoording af.

Ondertussen veranderen drie dingen, en geen van de drie klopt aan.

Wat een machine kan, groeit. Niet met sprongen die je opvallen, maar met een beetje per maand. De taak die vorig jaar drie keer opnieuw moest, lukt nu in één keer — en niemand stuurt je daar een bericht over.

Wat het kost om het uit te leggen, daalt. Dat is de stillere van de twee, en de belangrijkste in dit boek. Elke keer dat je iets opschrijft — een regel, een voorbeeld, een fout in een logboek — wordt de volgende uitleg korter. Op een gegeven moment is *uitleggen* goedkoper dan *doen*, en dat moment passeert zonder geluid.

En wat jij aan aandacht overhebt, daalt ook. Naarmate er meer loopt, is jouw minuut meer waard. Dezelfde twee minuten die vorig jaar niets kostten, kosten nu iets — niet omdat je het drukker hebt, maar omdat ze ergens anders meer opleveren.



Wat een machine kan stijgt, wat uitleggen kost daalt — en ergens kruisen ze.

Drie bewegingen, alle drie traag, alle drie dezelfde kant op. Het besluit dat je ooit nam, wordt elke maand een beetje minder waar. En omdat het nooit ineens fout is, blijft het staan.

Daarom is het een vraag en geen regel

Een regel zou zijn: *besteed alles uit wat een machine kan.* Dat is onzin, want dan besteed je dingen uit die je zelf sneller doet, en dingen waar je oordeel bij nodig hebt, en dingen die je nog niet kunt controleren.

De vraag is beter, om drie redenen:

Hij vraagt niet om een systeem. Je hoeft geen inventarisatie te maken van al je taken. Je stelt hem terwijl je bezig bent, aan het ding waar je op dat moment mee bezig bent.

Hij accepteert "omdat het nog niet kan". Dat is een eerlijk antwoord, en het is bruikbaar: het noemt precies wat er zou moeten veranderen. *Nog niet* is een datum zonder getal.

En hij houdt zichzelf actueel. Dezelfde vraag, over een maand, bij hetzelfde werk, kan een ander antwoord geven zonder dat je iets hebt hoeven bijhouden.

De vier antwoorden, en wat je met elk doet

In de praktijk komen er vier soorten antwoorden, en ze vragen allemaal iets anders.

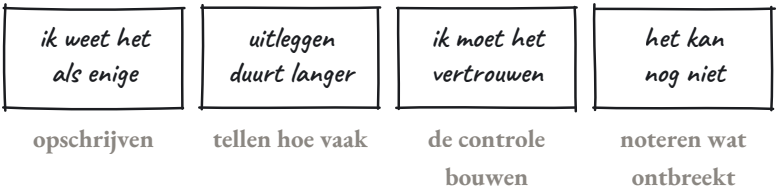
"Omdat ik de enige ben die het weet." Dit is het duurste antwoord en het klinkt het meest als een compliment. Wat je weet zit in je hoofd en reist nergens heen — niet naar een collega, niet naar een agent, niet naar jezelf over drie maanden. **De handeling hier is niet uitbesteden maar opschrijven.** Daarna is de vraag pas te beantwoorden.

"Omdat het uitleggen langer duurt dan het doen." Bijna altijd waar voor één keer, en bijna nooit waar voor tien keer. Reken het uit in plaats van het te voelen: hoe vaak deed ik dit de afgelopen maand? Bij één keer heb je gelijk. Bij vijf keer heb je het al drie keer verloren.

"Omdat ik het moet kunnen vertrouwen." Een goed antwoord, en het wijst naar het echte werk: niet het uitvoeren maar het controleren.

Kun je achteraf zien of het goed ging? Kun je dat goedkoop zien? Zolang dat niet kan, moet je het inderdaad zelf doen — en dan is het bouwen van die controle de eigenlijke taak.

"Omdat het nog niet kan." Het beste antwoord, want het is concreet. Er ontbreekt iets: een toegang, een sleutel, een stuk gereedschap, een voorbeeld. Schrijf op wát er ontbreekt. Dat is een ticket, geen gevoel.



Vier antwoorden, en elk wijst een andere handeling aan.

Let op wat er in geen van de vier staat: "omdat ik het leuk vind". Dat is een geldig antwoord, en het is het enige dat geen handeling nodig heeft. Maar noem het dan zo, in plaats van er een van de andere vier voor in de plaats te zetten.

De ene keer dat ik hem niet stelde

Als ik één ding over mag doen uit die eerste dagen, is het dit, en het is precies de vraag uit dit hoofdstuk die ik toen niet aan mezelf stelde.

Ik heb **de opstartfase zelf zitten uitdenken.** Hoe het gereedschap zichzelf zou moeten kunnen maken, in welke volgorde, met welke tussenstappen — het soort puzzel waar je uren in kunt zitten en waar je aan het eind van de avond trots op bent.

Dat was werk dat ik niet had hoeven doen. **Ik had het moeten formuleren en weggeven.** Niet "los dit op", maar: dit is waar we heen moeten, dit is wat er al is, dit zijn de voorwaarden — en kom maar met een volgorde.

Waarom deed ik dat niet? Omdat het voelde als het interessante deel. En precies dat is het verraderlijke aan deze vraag: **je stelt hem vanzelf bij het saaie werk, en bijna nooit bij het werk waar je plezier in hebt.** Het saaie werk géééf je graag weg. Het puzzelwerk houd je vast, en je noemt het denkwerk.

Soms is het dat ook. Maar de toets is niet of het leuk is; de toets is of jouw hoofd er iets aan toevoegt dat er anders niet zou zijn. Bij die opstartvolgorde was dat antwoord nee — ik had geen kennis die daar doorslaggevend was, ik had een voorkeur.

Dus als je de vraag uit dit hoofdstuk één keer per dag stelt, stel hem dan bij het werk dat je het leukst vindt. Daar staat het antwoord het langst stil, en daar is het het duurst.

Zelf doen · Vraag wat dit rekenwerk nog bij jou doet

Wat je doet. Elke maand zet iemand in dit bedrijf de begrote uren naast de werkelijke. Dat kost een uur, het gaat nooit mis, en daarom is er nooit meer naar gekeken. Laat het narekenen en kijk of het antwoord nog hetzelfde is. Reken op vijf minuten.

TYP DIT

Zet per opdracht de begrote marge naast de werkelijke, uit kostenbewaking/ en nacalculaties/. Doe de lopende opdrachten apart en laat daar zien hoeveel procent van de begrote kosten al gemaakt is.

Wat je ziet. Zes regels. Twee afgeronde opdrachten die keurig binnen de begroting bleven, één die zestien procentpunten marge verloor — **en een lopende opdracht die al 106 procent van zijn begrote kosten heeft gemaakt terwijl hij nog loopt.** Die laatste stond in geen enkel maandoverzicht, omdat een maandoverzicht naar afgeronde opdrachten kijkt.

WAT ER TERUGKWAM

```
NLOP-2026-001 begroot  7.9%  werkelijk  8.5%   +0.6
NLOP-2026-004 begroot 15.2%  werkelijk  8.6%   -6.6
NLOP-2026-010 begroot 19.9%  werkelijk  3.5%  -16.4
NLOP-2026-002 begroot  8.2%  loopt nog (106% van de
                            kosten gemaakt)
NLOP-2026-012 begroot 14.9%  loopt nog (54%)
NLOP-2026-007 begroot  8.7%  loopt nog (60%)
```

Waar het misgaat. Vraag expliciet om de lopende opdrachten apart. Zonder dat verzoek rekent hij de marge uit over alles, en dan verdwijnt een opdracht die op ontsporen staat tussen drie die goed zijn gegaan. **Let op wat er níet in de bron staat.** Deze getallen komen uit `kostenbewaking/`, en daar staan de openstaande crediteurenfacturen niet in — de werkelijke kosten zijn dus hoger dan wat je hier ziet. Dat is geen rekenfout maar een bronfout, en hoofdstuk 14 gaat daarover. **En vraag niet om een oordeel.** Of zestien procentpunten erg is, hangt af van dingen die niet in deze map staan.

EN DAN ZO

> Tel bij de kosten tot nu toe ook de openstaande crediteurenfacturen per opdracht op, en laat zien welk beeld daardoor verandert.

EN DAN DIT

```
met openstaande crediteuren erbij:
  NLOP-2026-007   60% ->  94% van de begroting
  NLOP-2026-012   54% ->  67% van de begroting
  NLOP-2026-002  106% -> 106% (geen openstaande)
en: NLOP-2026-011 heeft wel een crediteur maar
    staat helemaal niet in kostenbewaking/
```

En dit gaat niet over software

De vraag is ouder dan het onderwerp van dit boek, en dat is precies zijn kracht.

- **Een eigenaar die zelf de kas doet** deed dat ooit omdat er niemand anders was. Twee jaar later is er wel iemand, en doet hij het nog steeds — niet uit wantrouwen, maar omdat niemand de vraag stelde.
- **Een arts die zelf afspraken inplant** verdedigt dat met uitzonderingen die er echt zijn. De vraag is niet of die uitzonderingen bestaan, maar of ze de hele agenda moeten sturen.
- **Een monteur die elke storing zelf naloopt** heeft gelijk over de moeilijke gevallen. De makkelijke tachtig procent zit er alleen bij in.
- **Een schrijver die zijn eigen opmaak doet** zegt dat niemand anders het zo doet als hij. Dat is waar, en de vraag is of dat verschil iets waard is voor de lezer.

In alle vier gaat het niet om luiheid en niet om delegeren als deugd. Het gaat erom dat een besluit dat ooit klopte, stil is blijven staan.

En de vraag is beleefd. Dat is geen bijzaak. "Waarom doe je dat nog zelf?" laat ruimte voor een goed antwoord, en daarom durf je hem te stellen — aan een ander, en aan jezelf.

Probeer: één keer per week, tien minuten — laat de vijf dingen opsommen die je het vaakst herhaald hebt, met per ding de voorwaarde die nog ontbreekt voordat je het niet meer zelf hoeft te doen. Vraag om die ontbrekende voorwaarde en niet om een oplossing: een oplossing die je nog niet kunt controleren, is een probleem dat je verplaatst.

Die ronde levert vier soorten antwoord op. Bij drie ervan weet je nu wat de handeling is; het duurste antwoord vraagt iets anders, en dat is geen uitbesteden.

Tip: hoor je bij jezelf *"ik ben de enige die dat weet"*, laat je dan uitvragen tot het als werkwijze op papier staat die iemand anders kan volgen. Laat de vragen stellen in plaats van zelf te dicteren — je slaat automatisch over wat voor jou vanzelfsprekend is, en juist dat is de stap die de ander mist.

En er is één zin die bijna elke blokkade opheft, mits je hem op de goede plek zet.

Zet dit in je vaste instructies: *bij twijfel kies je zelf een plausibele invulling en ga je door; zeg er achteraf bij wat je gekozen hebt en waarom, en vraag alleen als de keuze onomkeerbaar is, geld kost of naar buiten gaat.* De meeste stilstand komt niet van moeilijke vragen maar van kleine — hoe moet dit heten, waar zet ik het neer — en elke keer dat zo'n vraag bij jou terechtkomt, wacht het werk tot jij tijd hebt.

Dat is geen vrijbrief, en hij werkt alleen omdat de rest van dit boek eronder ligt. Er draaien controles mee, er zijn grenzen die in de

uitvoering zitten en niet in een verzoek, en er is een spoor waarin staat wat er besloten is. **Een plausibele keuze is veilig zolang een verkeerde keuze zichtbaar en omkeerbaar is** — en waar dat niet zo is, bakent de laatste regel van die instructie het af. Eis wel dat de keuze achteraf benoemd wordt; anders krijg je voortgang zonder spoor.

> **Wat dit goedkoper maakte:** *elke keer dat de vraag een ander antwoord kreeg, verschoof er werk dat daarna nooit meer op mijn stapel terugkwam.*

9. Zichzelf controleren

Er was iets mis met de iconen en niemand kon zeggen wat.

Ze stonden er, ze hadden de goede vorm, ze stonden op de goede plek. Ze zagen er alleen uit alsof er een matglazen plaatje overheen lag — alsof iemand ze had uitgegrijsd. Dat was precies de beschrijving, en hij klopte.

Er volgden drie rondes. In de eerste gingen de kleuren omhoog naar de waarden die uit het origineel waren gemeten. In de tweede gingen ze nóg verder omhoog, tot volle verzadiging. In de derde werden de lijnen dikker gemaakt en werd er fijner bemonsterd. Na elke ronde: een nieuwe schermafdruk, en precies hetzelfde beeld.

Drie rondes zonder enig verschil is geen pech. Het is een mededeling, en ze luidt: je draait aan de verkeerde knop. Als de kleur omhoog gaat en er verandert niets, dan zit de rem niet in de kleur.

Toen werd er niet meer gekeken maar gemeten. Niet naar het beeld — naar de gegevens waaruit het beeld wordt gemaakt. Een klein programmaatje las de icoondata en drukte per icoon af hoeveel dekking de donkerste plek haalde. Op een schaal van 0 tot 255 kwam eruit: **31**. Twaalf procent. En geen enkele pixel haalde 255, in geen van de iconen.

Dat getal kan geen kleurprobleem zijn en geen vormprobleem. Het is een antwoord.

De oorzaak lag in de lus die de iconen tekent. Die telt per beeldpunt hoeveel van vierenzestig deelmonsters binnen de vorm vallen, en deelt dat door vierenzestig. Er stond een afbreking in: bij de eerste treffer sprong hij uit de binnenste lus. Daardoor kon de teller nooit hoger komen dan acht, terwijl er wel door vierenzestig gedeeld bleef worden. Alles werd getekend op twaalf procent van zijn dekking.

Die ene regel weghalen loste het op. Daarna: maximale dekking 255, en in het eenvoudigste icoon 130 van de 400 beeldpunten volledig dekkend.

"Ziet er goed uit" is geen oordeel. Eenendertig van de tweehonderdvijfenvijftig is er wel een. Drie rondes kijken vonden het niet; één meting vond het in een seconde.

De fout die eruitziet als een smaakkwestie

Waarom dit zo lang duurde, is de interessantere helft. De fout zag er niet uit als een fout. Hij zag eruit als een keuze.

Dat is een klasse op zichzelf, en het is de klasse die het langst overleeft. Een programma dat crasht, meldt zichzelf. Een programma dat iets op twaalf procent tekent, ziet eruit als iemand met een voorkeur voor gedempte kleuren. Er is niets dat piept. Er is alleen iemand die er af en toe naar kijkt en denkt: het kan wel wat sterker.

Er is één zin die dat samenvat, en hij gaat over iets heel anders:

> **Er is niets dat het tegenhoudt of meldt — het gaat stil mis.**

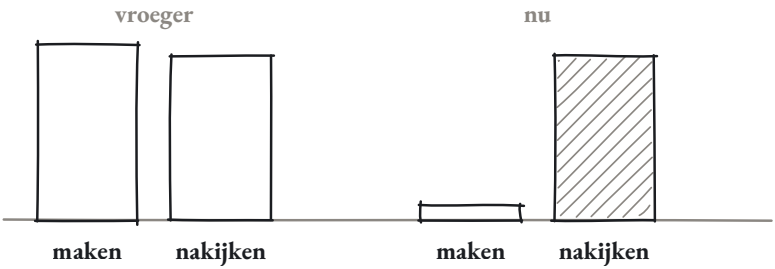
Stil misgaan is de normale toestand van gegenereerd werk. Niet omdat het slecht is, maar omdat er niets is dat het tegenhoudt. En dat brengt ons bij het probleem waar dit hele boek over gaat, nu in zijn scherpste vorm.

Genereren werd gratis, controleren niet

Zolang je zelf alles nakijkt, ben je de flessenhals. Dat is geen ergernis maar een rekensom: honderd dingen per dag laten maken en honderd dingen per dag zelf beoordelen, is honderd beoordelingen per dag, en dat aantal heeft een plafond dat niets met techniek te maken heeft.

Er is maar één uitweg, en die is niet "beter genereren". Het is: **de controle zelf moet goedkoop worden, en ze moet bij de machine komen te liggen.**

Dat klinkt onmogelijk zolang controleren "ernaar kijken" betekent. Het wordt mogelijk zodra controleren "een getal uitrekenen" betekent. Dat is de omslag die dit hoofdstuk beschrijft, en hij bestaat uit vier gewoontes die niets met elkaar te maken lijken te hebben.



Genereren werd bijna gratis. Controleren werd dat niet, en daar zit nu de rem.

Eén: meten in plaats van vragen

Terug naar de iconen, maar nu naar het patroon eromheen.

Bij het bouwen van één onderdeel van de interface noteerde ik acht opmerkingen, en dit zijn ze letterlijk:

> *"te kleinig" — "schraal" — "schaal je iconen iets te klein?" — "iconen bijna niet te zien zo donker" — "niet helder genoeg" — "heb je een soort disabled filter erover" — "de highlight is teveel verschoven omhoog" — "check maat en afstand chevrons"*

Achteraf bleek: **alle acht waren te meten uit een schermafdruk.** Iconen die elf beeldpunten hoog waren waar eenentwintig gemeten was. Een grijs dat lichter hoorde te zijn dan het was. Inkt die achttien boven en vijf onder de tekst zat waar twaalf en elf hoorde. Geen van de acht was een oordeel over smaak. Alle acht waren getallen, en ze stonden in het beeld dat er al was.

Dat is de scheiding waar alles aan hangt: wat te meten is, is je werk; wat werkelijk smaak is, leg je voor. Niet omdat vragen erg is, maar omdat elke vraag een mens nodig heeft en het aantal mensen niet meegroeit. Acht vragen die geen vraag hadden hoeven zijn, zijn acht keer dat het systeem stilstond.

En let op de kant die minder opvalt: dit maakt de vragen die overblijven bruikbaar. Iemand die acht keer gevraagd wordt of iets mooi is, kijkt bij de negende niet meer goed.

Twee: de lus tot het klopt

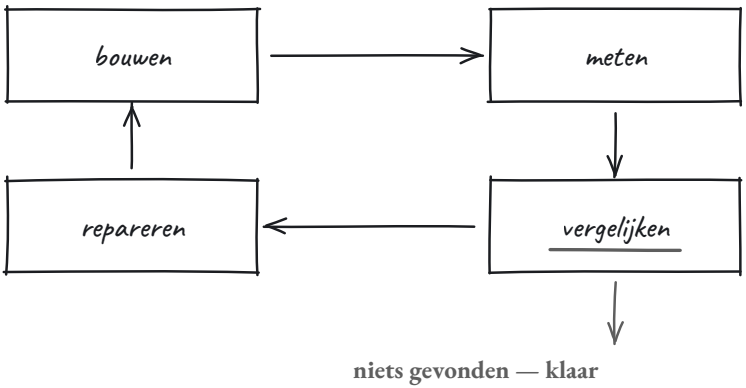
De tweede gewoonte is een vorm, geen inzicht: bouwen, starten, opnemen, vergelijken met de referentie, één oorzaak aanwijzen, repareren, opnieuw.

Vijf tot tien rondes achter elkaar, zonder tussentijds te vragen. Twee regels houden die lus gezond:

Eén oorzaak per ronde. Vindt een ronde niets meer, dan is de lus klaar en pas dán wordt er iets getoond. Vindt een ronde er vijf, dan is de kans groot dat het één fout is met vijf symptomen — en dan zoek je eerst de gemene deler, want vijf reparaties op één oorzaak leveren vier nieuwe fouten op.

Twee rondes bijstellen zonder effect betekent stoppen met bijstellen en beginnen met meten. Dat is de regel die de iconen drie rondes eerder had kunnen oplossen, en ze staat er nu omdat hij drie rondes gekost heeft.

Dit werkt alleen als de rondes goedkoop zijn. Bij een bouwstap van elf milliseconden kan een machine het tien keer proberen; duurt hij dertig seconden, dan probeert hij het één keer en gokt hij de rest. Snelheid is hier geen comfort maar de voorwaarde waar de hele lus op staat — daarover gaat het volgende hoofdstuk.



De lus: bouwen, meten, vergelijken, repareren — en pas stoppen als er niets meer te vinden is.

Drie: de controle die zichzelf wantrouwt

Op een dag klapte de editor eruit bij het aanklikken van een bestand.

De eerste reflex is altijd dezelfde: *ik heb het net gesloopt.* De tweede reflex is de waardevolle: bouw de vorige versie opnieuw op en probeer het daar. Zelfde crash. Hij zat er al dagen in.

De oorzaak waren twee beveiligingen die elkaar omkeerden. De eerste zorgt dat de cursor nooit onder nul komt. De tweede zorgt dat hij nooit voorbij de laatste regel komt. Bij een bestand van nul regels zet de tweede hem op min één — de beveiliging was de bron van de fout.

Een bevinding is pas een bevinding als je hem tegen de vorige versie hebt gehouden. Anders is het een bewering, en drie van de vier keer de verkeerde. Het verschil kost één herbouw, en het is het verschil tussen "ik heb een fout gevonden" en "ik heb iets stukgemaakt" — twee conclusies die tegengesteld werk opleveren.

Er zit nog een tweede les in en die is ongemakkelijker: **niemand had er ooit op geklikt.** Weken aan een interface bouwen, en de meest voor de hand liggende handeling die een gebruiker verricht, was nooit uitgevoerd. Wie test, bepaalt wat er gevonden wordt. Een machine die zichzelf controleert op precies datgene wat ze zelf heeft bedacht om te controleren, vindt precies dat.

Vier: de meetopzet is ook verdachte

Op diezelfde dag leek er twee keer een fout in de code te zitten, en zat hij twee keer in de proef.

De eerste keer stonden er twee vensters open met dezelfde titel. Het gereedschap dat een venster opzoekt geeft ze in willekeurige volgorde. De toetsaanslag ging naar het ene venster en de schermafdruk kwam van het andere. Conclusie: het paneel opent niet. Die conclusie was fout, en ze zag er volkomen solide uit — er wás een schermafdruk, en daar stond het paneel niet op.

De tweede keer deed een toetscombinatie niets. Vijf minuten zoeken in de tabel waar die combinatie in hoort te staan. Tot bleek dat het gereedschap *synthetische* toetsaanslagen verstuurt, en dat de omgeving waarin het programma draaide die categorie negeert. Met een echte toetsaanslag werkte het meteen.

De derde keer was de scherpste, en hij kwam veel later. Een toets die de bijschriften in een tekening nakijkt, stond groen. Ik keek naar de tekening en zag twee bijschriften die elkaar inliepen — het eerste woord van het tweede label verdween achter het laatste van het eerste.

De eerste verdenking was de meting: als de breedte van een stuk tekst uit het lettertypebestand tien procent te laag uitvalt, staat élk bijschrift in het hele boek te dicht op zijn buurman, en ziet geen enkele toets het. Dus heb ik het nagemeten — dezelfde teksten langs precies de weg van het boek gerenderd en de inkt in de afbeelding geteld. **De meting klopte, binnen een of twee procent.**

Wat er dan wél aan de hand was, is een subtieler soort fout. De toets vroeg: *overlappen deze twee?* Het antwoord was nee, en dat was correct. De twee stonden ongeveer een halve millimeter uit elkaar: ze raakten elkaar niet en ze lazen als één woord.

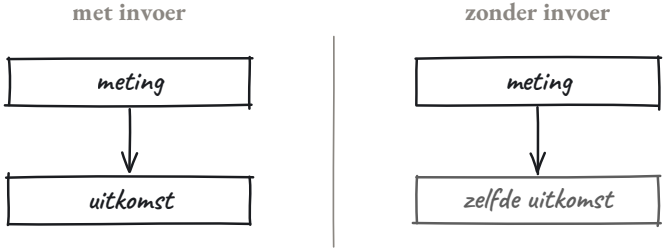
> **Er was een grens, maar geen ondergrens.** *En een grens zonder ondergrens keurt alles goed wat er nét aan voldoet.*

Dat is niet de meetopzet die stuk is — die was in orde. Het is de **vraag** die te smal was. Een groene toets is nooit meer waard dan de vraag die hij stelt, en die vraag staat nergens in het antwoord.

Wie meet, controleert eerst zijn meetlat. Dat is oud advies, en het wordt belangrijker naarmate je meer aan machines overlaat — want een machine twijfelt niet aan haar eigen opstelling. Ze rapporteert wat de opstelling zegt, met dezelfde stelligheid als wanneer het klopt. Een mens die twee vensters ziet staan, voelt iets. De machine telt ze niet eens.

En bij een toets die groen staat terwijl je iets ziet: **geloof je ogen en zoek de vraag.** Niet het antwoord is fout, maar wat er gevraagd werd. Het herstel is bijna altijd een ondergrens erbij — niet *"ze mogen elkaar niet raken"* maar *"er hoort zoveel tussen te zitten"*. Die nieuwe regel vond meteen een tweede geval dat ik niet had bekeken, en dat is het bewijs dat het er een was.

Praktisch betekent dit: de eerste stap van een diagnose is niet het zoeken naar de oorzaak maar het tellen van wat er draait. Hoeveel processen, hoeveel vensters, welke van de twee heeft de gate. Dat is drie regels werk, en het is het verschil tussen een middag zoeken en een minuut.



Voordat je de meting gelooft: meet eerst wat je meet als er niets gebeurt.

Een toets die je nooit rood hebt gezien

Nog één gewoonte, en hij is de goedkoopste van allemaal. Hij kost twee minuten en hij bepaalt of al het bovenstaande iets waard is.

> **Een groene test die je nooit rood hebt gezien, is geen bewijs.**

Een toets die groen staat, zegt twee dingen tegelijk en je kunt ze niet uit elkaar houden: *het klopt*, of *ik meet niets*. Een toets die per ongeluk naar de verkeerde plek kijkt, een assertie die altijd waar is, een bestand dat niet eens geladen wordt — ze zien er alle drie precies zo uit als een toets die zijn werk doet. Sterker: ze zijn *rustgevender*, want ze gaan nooit stuk.

De proef is dat je de fout terugzet. Breek expres wat de toets hoort te vangen, draai hem, en kijk of hij rood wordt. Daarna herstel je het. Dat is het hele ritueel, en het duurt korter dan het schrijven van de toets zelf.

Een voorbeeld. Er is een controle op de dekking van een set pictogrammen, want een fout in de rasteriser had ze ooit allemaal bleek gemaakt — ze stonden er, ze hadden de goede vorm, en ze waren zo flets dat het op een stijlkeuze leek. De nieuwe controle stond groen. Toen heb ik de fout teruggezet, en er vielen **negenentachtig van de zesennegentig** controles om. Pas op dat moment wist ik dat hij hem zou vangen.

Wie dit overslaat, bouwt een alarm en test nooit of de bel het doet. En dat is precies het soort zekerheid dat duurder is dan geen zekerheid, want je gaat erop leunen.

En de andere kant: een toets die te veel vangt, wordt uitgezet

Er is een tweede manier waarop een controle waardeloos wordt, en die voelt onschuldiger dan hij is.

Een toets kan **te smal** kijken: dan laat hij door wat hij hoorde te vangen, en je ziet groen terwijl het fout is. Dat is de fout hierboven. Maar hij kan ook **te ruim** kijken: dan meldt hij dingen die er niets mee te maken hebben, en je ziet rood terwijl het goed is.

De eerste kost je een fout. De tweede kost je de toets, want een controle die vals alarm geeft wordt binnen een week genegeerd of weggehaald. **Een gate die het halve werk afkeurt, is geen strenge gate maar een afwezige.**

Beide vormen komen voor, en soms binnen hetzelfde uur. Twee controles keken te smal — en een derde keek te ruim: hij moest een verboden woord tegenhouden en keurde meteen twee volstrekt normale zinnen af, omdat de toegestane uitzondering toevallig over een regeleinde was

afgebroken. Nog vijf van dat soort meldingen en niemand draait hem meer.

> **Kijk dus naar allebei de kanten: wat laat hij door, en wat meldt hij ten onrechte.**

Het tweede is bovendien makkelijker te meten dan het eerste, en dat wordt bijna nooit gedaan: laat hem één keer over álles lopen wat er al goed is, en tel hoeveel hij er afkeurt. Elk getal boven nul is werk voordat hij aan mag.

Zelf doen · Wantrouw je eigen controle

Wat je doet. Je controle uit het vorige hoofdstuk meldt vier overtredingen op regel 5. De vraag van dit hoofdstuk is niet of dat er vier zijn, maar of de controle zelf klopt. **Een meetopzet is ook een verdachte**, en je toetst hem door hem expres te laten falen. Reken op tien minuten.

TYP DIT

```
Verander in personeel/m-de-lange.md het certificaat "VOL-VCA" in
"VOL VCA", met een spatie in plaats van een streepje. Draai daarna re-
gel 5 opnieuw en vertel me wat er verandert en waarom.
```

Wat je ziet. Van vier overtredingen naar acht. **En de vier nieuwe zien er volkomen geloofwaardig uit**: *m-de-lange heeft geen VOL-VCA*, vier keer, netjes met opdrachtnummer. Er is niets aan te zien dat ze onzin zijn. De controle vergelijkt de gevraagde naam letterlijk met de naam in het personeelsdossier, en één spatie is genoeg om die vergelijking te laten mislukken zonder dat er iets misgaat.

WAT ER TERUGKWAM

```
regel 5 meldt nu 8 overtredingen:
  NLOP-2026-002: m-de-lange heeft geen VOL-VCA
  NLOP-2026-004: verlopen
  NLOP-2026-007: j-brinkman heeft geen Hijsbewijs
  NLOP-2026-008: verlopen
  NLOP-2026-009: m-de-lange heeft geen VOL-VCA
  ... en nog drie van dezelfde soort
```

Waar het misgaat. Zet de spatie daarna weer terug. Vergeet je dat, dan draagt je administratie vanaf nu een fout die je zelf hebt gemaakt, en die vind je over drie maanden niet meer terug. **Denk niet dat dit een kunstmatige fout is.** Een certificaat dat met de hand is ingetypt, een naam die de uitgever heeft gewijzigd, een spatie uit een geplakte tabel: het gebeurt vaker dan een verlopen certificaat. **En kijk naar de KANT waarop de fout uitpakt.** Deze controle meldt te veel, en dat merk je. Een controle die te weinig meldt, merk je niet — en dat is de gevaarlijke kant.

EN DAN ZO

Zet de spatie terug en laat de controle voortaan ook melden welke certificaatnamen hij niet heeft kunnen koppelen aan een naam in personeel/, apart van de echte overtredingen.

EN DAN DIT

```
regel 5  overtredingen: 4
regel 5  niet gekoppeld: 0
  (elke gevraagde naam is teruggevonden in
   personeel/, dus de vier zijn echt)
```

De gates: het mechanisme tegen ongemerkt afdwalen

De vier gewoontes hierboven vinden fouten. Er is nog een vijfde nodig, en die voorkomt iets anders: dat er ongemerkt iets verandert dat al goed was.

Een machine die zelfstandig doorwerkt, heeft geen gevoel voor wat er duur was om te bereiken. Ze ziet een onderdeel dat werkt en getest is als materiaal, net als al het andere. Zo verdwijnt er stilletjes iets dat drie dagen heeft gekost, in een wijziging die ergens anders over ging.

Een **gate** is daartegen de goedkoopste maatregel die er is: een verplichte zelfvraag vóór je iets aanraakt dat werkt en getest is. Niet meer regels, geen verbod — één moment waarop er gezegd moet worden waarom.

Twee eigenschappen maken hem bruikbaar in plaats van irritant:

Hij is niet absoluut. Overrulen mag. Werkende dingen moeten soms wijken, en een regel die dat verbiedt, wordt binnen een week omzeild in plaats van gevolgd.

De reden wordt opgeschreven. Dat is het hele mechanisme. Niet de toestemming is de waarde maar het spoor: over twee weken is er een zin die zegt waarom dit werkende ding is weggehaald, en die zin is het verschil tussen een besluit en een ongeluk.

Een afspraak die niemand controleert, geldt niet. Een gate is de kleinste vorm van controle die nog wel geldt, en hij kost één vraag.



rood: terug, en de lus is kort

Een gate laat niets door tot de controle groen is — en houdt dus fouten klein.

Wat de machine kan lezen, kan ze controleren

Er is één ding dat al het bovenstaande mogelijk maakt en dat makkelijk over het hoofd wordt gezien: **het werk moet sporen achterlaten in een vorm die een machine kan lezen.**

Een schermafdruk is voor een mens. Een getal is voor allebei. En daar zit de hele omslag in, als je hem toepast op het vastleggen van wat een agent doet:

> **Sla het op in een vorm die je kunt uitlezen, en je hebt geen schermafdruk meer nodig om te zien of het goed ging.** *Zolang het resultaat van een handeling alleen te zien is, kan alleen een mens het beoordelen. Zodra het ook te lézen is — een getal, een lijst, een gestructureerd bestand — kan de machine het zelf nakijken, en dan is de mens er niet meer voor nodig. Dat is niet een kwestie van betere modellen. Het is een kwestie van in welke vorm je je eigen werk opslaat, en dat is een keuze die je vandaag maakt.*

Probeer: zet in je opdracht niet alleen wát er gemaakt moet worden, maar ook waarmee het gecontroleerd wordt en dat er doorgegaan moet

worden tot het klopt — en noem het commando letterlijk. Verificatie is de flessenhals, niet generatie: zolang jij degene bent die nakijkt, schaalt er niets. Van acht koud gegeven taken kwam er één in één keer door de controle, dus de eerste poging is niet de oplevering. Het werkt alleen als controleren snél is: bij elf milliseconden probeert hij het tien keer, bij dertig seconden probeert hij het één keer en gokt hij.

Nakijken laten doen werkt pas als er iets te tellen valt. Anders krijg je een oordeel terug in plaats van een controle, en dan ben je niets opgeschoten.

Probeer: bedenk vóórdat je zelf gaat beoordelen welke eigenschap van het resultaat je kunt *tellen*, en laat die schifting eerst doen. Een fotograaf schift tweeduizend opnamen op scherpte en op ogen dicht voordat hij er één bekijkt; een vertaler laat eerst nalopen of alle getallen, namen en eenheden overeenkomen. Elk oordeel dat je zelf velt kost een mens, en het aantal mensen groeit niet mee. Van de acht opmerkingen hierboven waren er acht te meten uit een schermafdruk — geen enkele was smaak. Pak je laatste tien afkeuringen en vraag je bij elke af of er een getal onder zat; het antwoord verrast bijna iedereen.

Wat na dat schiften overblijft, is de moeite van het bewaren waard — want dezelfde fouten komen terug, en de tweede keer hoeven ze niets meer te kosten.

Tip: hou een logboek bij van elke fout die een model maakt — wat het schreef, waarom dat fout was, wat de goede vorm is — en maak daar de lijst van die je voortaan meestuurt. Sorteer op hoe váák, niet op hoe interessant. Een model vult bij iets wat het niet kent aan uit de buurtalen, en precies dat vangt zo'n lijst af. Eén regel per fout, in de gebiedende

wijs, met de goede vorm erbij: *"let op met types"* leert niets, *"declareer elke variabele vóór het begin"* wel.

En let op wat je in zo'n primer zet. Alles wat erin staat wordt nagedaan, ook het deel dat je er niet mee bedoelde.

Let op: zet voorbeeldsyntaxis in je opdrachten nooit zo neer dat een gereedschap hem als echte syntaxis kan lezen. Een opdracht is tekst in een bestand, en een bestand wordt gelezen door meer dan één ding — wat voor het model een voorbeeld is, kan voor een ander gereedschap een opdracht zijn. Het is echt gebeurd: de opdrachttekst bevatte als voorbeeld het woord waarmee je een ander bestand binnenhaalt, het gereedschap dat die tekst las zag het staan en voerde het uit, en sloeg de échte verwijzing verderop over als dubbel. Overal waar tekst voor twee lezers bedoeld is, kan de ene lezen wat voor de andere bedoeld was.

> **Wat dit goedkoper maakte:** *elke controle die een getal werd, hoefde daarna nooit meer door een mens gedaan te worden.*

10. Snelheid is geen comfort

De agent had het al vier keer geprobeerd toen ik de regel las waar hij op vastliep.

Hij schreef een stuk code, liet het bouwen, kreeg een foutmelding terug, las die, veranderde één ding en liet het opnieuw bouwen. Vier rondes, en ze pasten samen in de tijd die ik nodig had om te zien wát er misging. Bij de vijfde ronde was het klaar. Ik heb er niets aan gedaan.

Dat is geen verhaal over een slimme agent. Het is een verhaal over een getal: de bouwstap duurde **elf milliseconden**. Daardoor kostte vier keer proberen minder dan een oogwenk, en was "gewoon nog een keer" een redelijke zet.

Waarom snelheid de lus mogelijk maakt

Snelheid wordt bijna altijd verkocht als comfort. Minder wachten, prettiger werken, een fijnere dag. Dat is het ook, en dat is niet waarom het hier staat.

Snelheid is wat de lus mogelijk maakt. De werkwijze uit het vorige hoofdstuk — bouwen, meten, vergelijken, repareren, en pas stoppen als er niets meer te vinden is — is een lus die tien of twintig keer rond moet kunnen. Wat één ronde kost, bepaalt of die lus bestaat.

Reken het uit en het is niet ingewikkeld. Tien rondes van elf milliseconden is ruim een tiende seconde: je merkt het niet eens. Tien rondes van dertig seconden is vijf minuten, en dat is voor één klein probleem een prijs die iemand gaat wegen. Vijftig rondes is bij het ene getal nog steeds een halve seconde, en bij het andere een halve werkdag.

Daarmee is een prestatiegetal iets anders geworden dan een prestatiegetal. Het is een **ontwerpvoorwaarde**: niet *sneller is prettiger*, maar *onder deze grens werkt deze werkwijze en erboven niet*.



Onder de drempel probeert een machine het tien keer. Erboven probeert hij het één keer en gokt de rest.

De grens is geen helling maar een trap

Hier zit het deel dat verrast. Je zou verwachten dat het langzaam slechter wordt: hoe trager de stap, hoe minder rondes, netjes evenredig. Zo gaat het niet.

Wat er gebeurt is dat het **gedrag omslaat**. Onder een bepaalde grens is nog een keer proberen goedkoper dan nadenken over wat het zou kunnen zijn, dus wordt er geprobeerd. Boven die grens wordt nadenken goedkoper dan proberen — en "nadenken" heet hier, bij een machine én

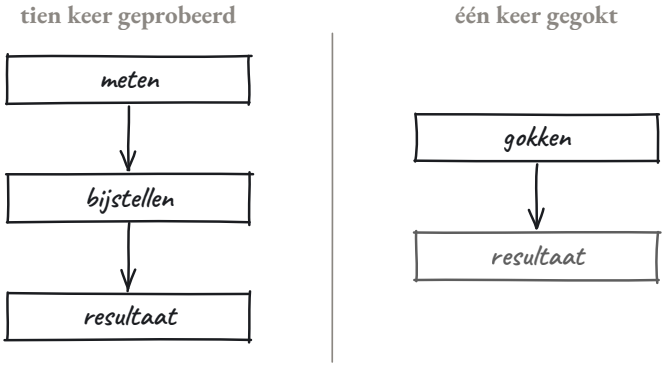
bij een mens, **gokken**. Niet één keer per ronde iets langzamer, maar een andere manier van werken.

Iedereen die weleens op een trage bouwstap heeft gewacht, herkent dit van zichzelf. Je verandert niet één ding en kijkt wat er gebeurt. Je verandert drie dingen tegelijk, want je wilt niet drie keer wachten. En dan werkt het, en weet je niet welk van de drie het was.

Het ergste is dat het niets zegt

Een lus die te duur is geworden, valt niet om. Hij geeft geen fout, hij stopt niet, er komt geen melding dat er iets is opgegeven. Hij levert gewoon een antwoord dat er precies zo uitziet als een antwoord dat wél is nagekeken.

Dat is de stilste vorm van kwaliteitsverlies die er is, en hij is aan de uitkomst niet te zien. Je ziet hem alleen aan het getal ernaast: hoeveel rondes zijn er werkelijk gedraaid? Nul rondes en vijftien rondes leveren allebei een resultaat op. Alleen de tweede is gecontroleerd.



Tien keer proberen en één keer gokken leveren allebei een antwoord op. Aan het antwoord is het verschil niet te zien.

En dit gaat niet over software

De grootheid die hier telt is niet *snelheid*. Het is **de kosten van één keer proberen**, en die bestaat in elk vak.

- **Een goedkeuring die een week duurt**, betekent dat niemand een half idee indient om te kijken wat ervan komt. Er worden alleen nog uitgewerkte voorstellen ingediend, en die zijn per stuk duurder en per stuk vaker mis. Zakt diezelfde goedkeuring naar een uur, dan verandert niet het tempo maar de sóórt voorstellen die er binnenkomt.
- **Een proef die een dag kost**, wordt één keer gedaan met de beste gok. Kost hij tien minuten, dan worden er vijf varianten geprobeerd en weet je bovendien welke variabele het deed.
- **Een kok die tussendoor proeft**, stuurt twintig keer bij per gerecht. Een kok die pas aan tafel hoort hoe het was, stuurt één keer bij per week.
- **Een offerte die twee dagen kost om op te stellen**, wordt niet uitgebracht bij twijfel. Dat is geen luiheid; dat is dezelfde rekensom.

In alle vier de gevallen is de vraag niet "kan het sneller" maar: **ligt de prijs van één ronde onder de grens waar mensen ze nog durven te doen?** Ligt hij eronder, dan krijg je vanzelf meer rondes en dus een beter resultaat, zonder dat er iemand beter zijn best doet.

En als je de duizend-keer-vraag erbij pakt, wordt het scherper. Wat bij een enkele ronde een ongemak is, is bij duizend rondes het verschil tussen "dit draait vannacht door" en "hier moet iemand bij blijven".

Zelf doen · Meet hoe lang je ronde duurt

Wat je doet. Je ronde uit hoofdstuk 6 loopt de hele administratie af. Voordat je hem vaker gaat draaien, moet je weten wat hij kost — want dat bepaalt of je hem tussendoor draait of één keer per maand, en dat verschil bepaalt weer wanneer je iets vindt. Reken op drie minuten.

TYP DIT

> Draai de ronde drie keer achter elkaar en meet elke keer hoe lang hij erover doet. Zeg daarna wat dat betekent voor hoe vaak ik hem kan draaien.

Wat je ziet. Drie keer twee honderdste seconde. **Dat is geen prestatie maar een drempel**: onder een seconde draai je iets tussendoor, zonder erover na te denken. Boven de tien seconden doe je het als je eraan denkt, en boven de minuut alleen nog als het moet. De controle is niet beter geworden — hij is *goedkoper* geworden, en daardoor draai je hem tien keer vaker.

WAT ER TERUGKWAM

```
ronde  0.02 s
ronde  0.01 s
ronde  0.02 s
15 opdrachten, 12 regels
(en dat is inclusief het opstarten: de ronde
 zelf viel onder de meetgrens)
```

Waar het misgaat. Meet met de echte hoeveelheid, niet met een voorbeeldje. Vijftien opdrachten zeggen niets over vijftienhonderd; vraag erbij hoe de tijd meegroeit. **Meet drie keer en niet één keer.** De eerste keer zit er opstarttijd in die er daarna niet meer is, en dan meet je iets anders dan je denkt. **En let op wat je níet meet.** Dit is de tijd van de controle, niet van het lezen van de uitkomst — en dat laatste is bij vijftienhonderd opdrachten het echte probleem.

EN DAN ZO

Verdubbel de administratie een paar keer door de map te kopiëren, en laat zien hoe de tijd meegroeit met het aantal opdrachten.

EN DAN DIT

```
     15 opdrachten   onder de meetgrens
    240 opdrachten   0.002 s
    960 opdrachten   0.007 s
   3840 opdrachten   0.028 s
  15360 opdrachten   0.124 s
de tijd groeit recht mee: duizend keer zoveel
werk kost duizend keer zoveel tijd, en dat is
nog altijd een achtste seconde
```

Wat het kost om het snel te houden

Snelheid heeft een prijs. Elke keuze die hem opleverde, sloot iets anders af — en die prijs betaal je op een dag dat je er niet op zat te wachten.

Snelheid is dus geen kenmerk dat je later toevoegt. Hij is de optelsom van dingen die je niet gedaan hebt, en daarom valt hij ook niet achteraf in te bouwen: dan zit alles wat je wél deed er al in.

De praktische kant daarvan: als je vroeg in een project één eigenschap wilt beschermen tegen gemak, kies dan deze. Bijna alles is later nog te repareren. De kosten van één ronde niet.

Probeer: zet een klok op de stap die je het vaakst herhaalt — tien keer achter elkaar, en vraag om de mediaan én de langste, niet het gemiddelde. De langste is het getal dat bepaalt of iemand het nog een keer doet; één ronde van dertig seconden tussen negenennegentig snelle in bepaalt het gedrag. Vraag daarna wat de drie grootste posten in die tijd zijn en wat er weg kan, met de prijs erbij aan functionaliteit — anders krijg je

alleen de mooie helft. De vraag is nooit *is dit snel* maar *past er tien keer proberen in de tijd die iemand ervoor over heeft*.

Zodra je dat getal hebt, is de volgende vraag hoe je het vasthoudt. Snelheid gaat namelijk niet in één keer weg.

En leg de grens vast voordat je hem nodig hebt: schrijf op wat de duurste stap mag kosten voordat je werkwijze niet meer werkt, en laat een test meedraaien die faalt zodra hij daaroverheen gaat. Snelheid gaat niet in één keer weg maar in stapjes die elk verdedigbaar zijn, en er is geen moment waarop iemand zegt: hier stoppen we. Kies een grens die je nu ruim haalt — een grens die je vandaag al nét haalt, staat morgen rood om een reden die niets met snelheid te maken heeft, en wordt dan uitgezet.

> **Wat dit goedkoper maakte:** *proberen werd goedkoper dan gokken, en daarmee werd elke controle uit het vorige hoofdstuk iets dat een machine er gewoon bij kan doen.*

11. Niet op schijf, niet gebeurd

Toen ik voor dit boek terugkeek, had ik alles nog. Elke wijziging met een tijdstempel. Elke taak met de reden erbij. Ruim twaalfhonderd stukjes werk waarvan ik kon nalezen waarom ze bestonden.

En van de eerste vier dagen had ik niets.

Niet omdat er iets verloren was gegaan. De wijzigingen staan er tot op de minuut; ik kan je precies vertellen wat er die dagen gebeurde. Wat er niet staat, is waarom. Geen enkele overweging, geen enkele twijfel, geen enkele keuze met een reden erbij.

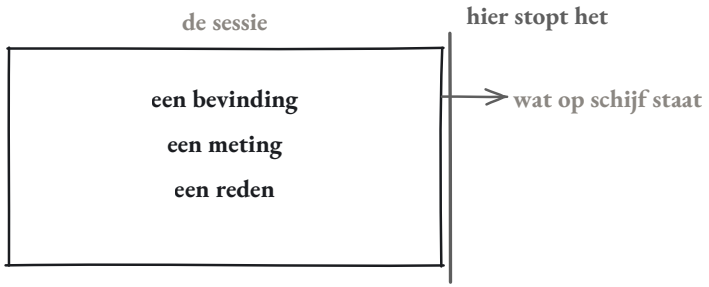
Er werd in die dagen gebouwd en niet overlegd, en dus schreef niemand iets op. Dat was op dat moment volkomen logisch. **Het waren de vier dagen waarin alles ontstond.**

Wat daar verdween

Het gekke is dat ik het toen allemaal wist. Elke keuze had een reden, en die reden was zo vanzelfsprekend dat opschrijven verspilling leek. Je schrijft niet op waarom je links afslaat als je de weg kent.

Twee weken later is de weg weg. Niet de route — die staat er nog — maar het landschap eromheen: wat je overwoog, wat je verwierp, waarom dit eerst moest en dat later. Dat zat in mijn hoofd en in mijn hoofd alleen, en het is er nu niet meer.

Dit is de meest onderschatte vorm van verlies die er is, want je merkt hem niet op het moment dat hij optreedt. Er gaat niets kapot. Er valt niets om. Het enige dat gebeurt is dat je over een maand naar je eigen werk kijkt en denkt: waarom heb ik dat zo gedaan.



Wat je zelf onthoudt is weg zodra je het niet meer paraat hebt; wat opgeslagen is blijft.

Waarom dit een multiplier is en geen administratie

Vastleggen voelt als overhead. Het is werk dat niets maakt, en het moet gebeuren op het moment dat je juist bezig bent met iets dat wél iets maakt.

Maar kijk waar het op aangrijpt:

- **Het maakt werk overdraagbaar.** Iets dat alleen in jouw hoofd zit, kan alleen jij doen — en dan is er altijd precies één van, hoeveel handen er ook beschikbaar zijn.
- **Het maakt herhaling goedkoop.** Dezelfde vraag voor de tweede keer beantwoorden kost bijna niets als het antwoord ergens staat, en evenveel als de eerste keer als dat niet zo is.

- **Het maakt je fouten bruikbaar.** Een fout die je onthoudt, maak je misschien niet nog een keer. Een fout die is opgeschreven, maakt ook niemand anders nog een keer.
- **En het is de enige manier waarop iets zonder jou verder kan.** Dat is niet een bijkomstigheid maar de hele voorwaarde voor alles wat in dit boek nog komt.

Het is dus precies de vorm uit het hoofdstuk over multipliers: op de dag zelf verlies, daarna vermenigvuldigend.

Achteraf werkt niet, en dat is geen kwestie van discipline

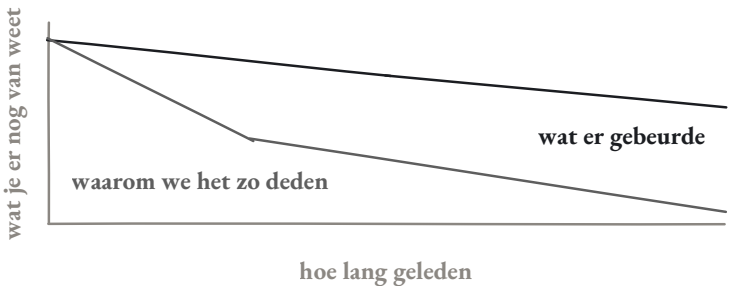
De voor de hand liggende oplossing is om het aan het eind te doen. Aan het eind van de dag, aan het eind van de week, aan het eind van het project — dan weet je immers hoe het is afgelopen.

Dat werkt niet, om twee redenen die allebei onoverkomelijk zijn.

De eerste: aan het eind is de helft van het waarom al weg. Je schrijft dan op wat je hebt gedaan, en dat staat meestal al ergens. Wat je niet meer opschrijft is wat je bijna had gedaan en waarom niet — en juist dat is het stuk dat je later nergens anders kunt terugvinden.

De tweede: er ís vaak geen eind. Een dag loopt af zonder afronding. Een project verandert van richting. Een gesprek wordt onderbroken. En elke keer dat er geen eind is, is er ook geen moment waarop het alsnog gebeurt.

Dus schrijf je het op terwijl het gebeurt. Niet netjes, niet volledig, niet in zinnen die iemand anders mooi vindt. Ruw is prima — ruw en aanwezig is oneindig veel meer waard dan verzorgd en er niet.



Het waarom slijt sneller dan het wat.

De reden is meer waard dan het besluit

Er is een reden dat ik dit hoofdstuk scherper voel dan de meeste, en die komt uit de begeleiding.

Daar werd van alles vastgelegd: wat er was afgesproken, wat er gedaan was, hoe het ging. Wat er bijna nooit stond, was **waarom** iets zo besloten was. En dat is precies het stuk dat je nodig hebt, want zonder reden kun je een besluit alleen maar opvolgen of overtreden — je kunt het niet **herzien**.

Neem iets simpels: er is afgesproken dat iemand op vaste dagen komt. Staat er alleen dát, dan is het over een half jaar een regel waar niemand meer van weet waar hij vandaan komt. Staat erbij waaróm — omdat het anders te vol werd, of omdat er iets samenviel — dan kun je zien of die reden er nog is. In het eerste geval heb je een gewoonte. In het tweede een besluit dat mee kan bewegen.

En wat er niet staat, wordt opnieuw uitgevonden op de persoon zelf. Dat is de echte kosten, en hij valt bij iemand anders. De volgende begeleider begint opnieuw: opnieuw uitvragen, opnieuw uitproberen, opnieuw ontdekken wat al drie keer ontdekt was. Voor de organisatie is

dat verloren tijd. Voor degene die het overkomt, is het de derde keer hetzelfde gesprek over zichzelf.

Bij een agent is het dezelfde vorm zonder die last: wat er niet staat, wordt opnieuw uitgezocht. Alleen betaal je daar in tokens en minuten, en voel je het dus nauwelijks — wat precies de reden is dat het daar nog makkelijker wordt overgeslagen.

De regel die ik eruit meenam, geldt in beide vakken: leg niet vast wat je besloot, leg vast waaróm, en doe het op het moment dat je het weet. Een uur later ken je je eigen overweging al niet meer, en dan schrijf je op wat er gebeurde in plaats van waarom het gebeurde.

En dit is geen software

Een adviseur die alles in zijn hoofd heeft, kan nooit twee opdrachten tegelijk, kan niemand inwerken, en kan niet ziek worden. Hij noemt dat ervaring. Het is ook een ontwerpkeuze, en hij heeft hem waarschijnlijk nooit bewust gemaakt.

Een werkplaats waar de monteur weet welke auto's dit mankement hebben, weet dat kwijt op de dag dat hij vertrekt. Een werkplaats die het bijhoudt, weet het daarna nog steeds — en ziet bovendien iets wat niemand kon zien toen het in één hoofd zat: dat het bij dat ene bouwjaar vaker voorkomt.

Een school ontdekt elk jaar opnieuw waar leerlingen struikelen. Elk jaar opnieuw, want het stond nergens. **In alle drie de gevallen is het probleem niet dat men het niet weet, maar dat het weten nergens staat.**

Zelf doen · Leg het besluit vast, niet alleen de uitkomst

Wat je doet. Op 15 april week de maatvoering van de aanbouw vier centimeter af van de tekening. Dat is opgelost, en het besluit staat in de notulen. **Maar de reden staat er niet**, en over een jaar is dat het enige wat iemand nodig heeft. Laat dat zoeken en laat er meteen een werkwijze van maken. Reken op tien minuten.

TYP DIT

> Zoek in uitvoering/ de plekken waar een besluit is genomen, en kijk per besluit of de reden erbij staat of alleen de uitkomst. Schrijf daarna een herbruikbare werkwijze voor het vastleggen van een besluit, met bovenaan één regel over wanneer hij van toepassing is.

Wat je ziet. Vier notulen met besluiten, en vier dagrapporten waarin iets wordt besloten zonder dat het een besluit heet — *gemeld bij werkvoorbereiding*, *doorgezet*, *opnieuw plannen*. **In geen van de acht staat waaróm.** Bij de maatvoering staat wél de uitkomst (*afwijking 4 cm akkoord na aanpassing tekening*) en niet de afweging: waarom aanpassen goedkoper was dan verplaatsen.

WAT ER TERUGKWAM

```
besluiten met een reden erbij : 0 van de 8
besluiten met alleen uitkomst : 8 van de 8

verborgen besluiten in dagrapporten:
  "gemeld bij werkvoorbereiding"
  "doorgezet"
  "opnieuw plannen"
  "meerwerk aangekondigd"
```

Waar het misgaat. Zoek ook op plekken die geen besluitenlijst heten. De helft van wat hier besloten wordt, staat in een dagrapport in een halve zin. Vraag je alleen om notulen, dan mis je die helft en denk je dat de administratie klopt. **Vraag om de REDEN als apart veld.** Zeg je dat

niet, dan komt er een werkwijze die de uitkomst netjes vastlegt, en dan heb je over een jaar precies hetzelfde probleem. **En schrijf het tijdens het werk, niet erna.** Een reden die je aan het eind van de week opschrijft is een reconstructie, en die klinkt altijd verstandiger dan hij was.

EN DAN ZO

> Zet in de werkwijze de reden als verplicht veld, en laat me zien wat er gebeurt als iemand hem leeg laat.

EN DAN DIT

```
besluit-vastleggen
  "Als er in een gesprek of op de bouwplaats een
   besluit valt -- legt vast wat, wanneer, door
   wie en WAAROM."
  reden leeg -> het besluit wordt niet
                weggeschreven, met de melding
                "een besluit zonder reden is een
                 uitkomst, geen besluit"
```

Drie dingen die echt de moeite waard zijn

De meeste dingen zijn later prima af te leiden uit wat er ligt. Wat je nergens uit kunt afleiden, is dit:

1. **Wat je hebt besloten, en waarom.** Vooral het waarom, want de beslissing zelf is meestal zichtbaar aan het resultaat.
2. **Wat er misging, en wat de oorzaak was.** Niet dat het misging — dat weet je nog wel — maar waar het aan lag, want dat is precies wat je de volgende keer had willen weten.
3. **Waar je was toen je stopte.** Eén regel. Wat je aan het doen was en wat het volgende is.

Dat derde punt lijkt het minst belangrijke en het is het meest gebruikte. Iemand die morgen verdergaat — jij, of iemand anders, of iets anders — hoeft dan niet te reconstrueren waar het verhaal was gebleven.

Probeer: zet boven in het document waar je werk in staat één regel — wat je aan het doen bent en wat het volgende is. Werk hem bij op het moment dat je ergens anders bent, niet aan het eind, en haal hem weg als het klaar is: een regel die naar afgerond werk wijst is erger dan geen regel. Hou het bij een verwijzing en niet bij een verslag — *"bezig met de offerte voor X, wacht op de prijzen van Y"* is genoeg. Wie er meer in zet, gaat hem niet bijhouden.

Wat dit goedkoper maakte: elke keer dat iemand verder moest waar een ander was gestopt — inclusief ik, twee weken later, die zijn eigen werk niet meer herkende.

12. Fratsen

In het werk met mensen naar werk hadden we er een woord voor, en ik ben het nooit meer kwijtgeraakt.

Fratsen. Kleine, onverwachte, onvoorspelbare of onhandige dingetjes met grote gevolgen. Geen rampen. Geen onwil. Meestal iets dat op zichzelf te klein is om over te beginnen: net te laat binnenkomen, een vraag niet stellen, een formulier half invullen, een afspraak wel nakomen maar zonder de spullen die erbij horen.

Een keer is het niets. Dat is precies waarom er niets aan gedaan wordt.

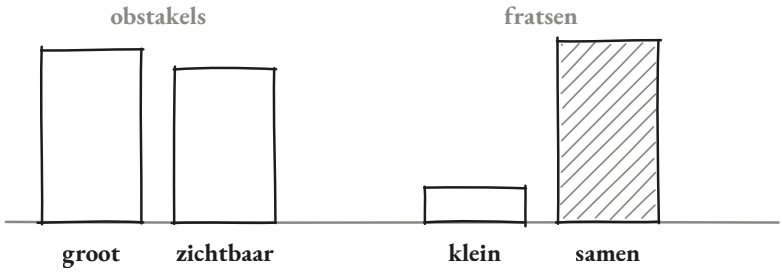
Waarom juist het kleine het knelpunt is

Het grote gaat niet mis. Dat klinkt vreemd en het is gewoon waar.

Grote obstakels zijn zichtbaar, en zichtbare dingen krijgen aandacht. Er wordt over vergaderd, er komt een plan, er is budget. **Fratsen liggen onder de drempel waarop iemand ingrijpt** — ze zijn te klein om een gesprek waard te zijn, en tegen de tijd dat ze bij elkaar groot genoeg zijn, is het patroon al gewoonte geworden.

En ze doen iets specifieks. Ze zorgen er niet voor dat het misgaat; ze zorgen ervoor dat het **net niet lukt.** De plaatsing die bijna slaagde. De baan die na zes weken toch weer eindigde. Niet om één reden die je kunt aanwijzen — om vijf kleine dingen die elk op zichzelf niets waren.

Dat was het patroon dat ik zag, keer op keer. Niet gebrek aan capaciteit, niet gebrek aan willen. Fratsen.



Grote obstakels krijgen aandacht. Fratsen liggen eronder en stapelen op.

Wat een frats onderscheidt van een fout

Dit onderscheid is de kern van dit hoofdstuk, en zonder dat onderscheid doe je het verkeerde.

Een **fout** gaat mis en stopt. Er komt een melding, iets werkt niet, iemand merkt het. Een fout is vervelend en hij is bruikbaar: hij leert je iets, en het volgende hoofdstuk gaat daarover.

Een **frats** gaat niet mis. Hij levert een resultaat op dat er in orde uitziet en het net niet is. Niemand merkt het op het moment zelf, want er is niets te merken. Je merkt hem pas aan de uitkomst een paar stappen verderop, en dan is de oorzaak al niet meer te vinden.

	een fout	een frats
wat je ziet	iets werkt niet	iets lijkt te werken
wanneer	meteen	een paar stappen later, of nooit
één keer	hinderlijk	onzichtbaar
duizend keer	duizend meldingen	het lukt structureel net niet
wat je ermee doet	oplossen en opschrijven	eerst zichtbaar maken

Fouten zijn je beste leraar. Fratsen leren je niets, want ze vertellen je niet dat ze er zijn.

En dit is precies wat een hallucinatie is

Toen ik met AI ging werken, herkende ik het meteen, en dat hielp me meer dan welke uitleg ook.

Een model dat crasht, is een fout. Een model dat een bibliotheek verzint die niet bestaat, of een bron noemt die er niet is, of een getal geeft dat nergens vandaan komt — dat is een frats. Het antwoord ziet er uit als een antwoord. De vorm klopt. De toon klopt. Er is niets dat piept.

En het gevolg is hetzelfde als op de arbeidsmarkt: **het zit de schaal in de weg.**

Niet omdat het zo vaak gebeurt. Juist omdat het weinig gebeurt. Als één op de twintig antwoorden een frats bevat, en je kunt niet zien welke, dan moet je ze alle twintig nakijken. Daarmee is de winst weg. Honderd taken tegelijk laten lopen heeft geen zin als er honderd nagekeken moeten worden door iemand die dat niet kan bijhouden.

Een frats maakt niet het werk duur. Hij maakt het vertrouwen duur — en vertrouwen is precies wat je nodig hebt om iets zonder jou te laten draaien.

Waarom er zelden over gesproken wordt

Er is nog een reden dat fratsen blijven bestaan, en die heeft niets met opmerkzaamheid te maken. In de begeleiding was het de belangrijkste van allemaal.

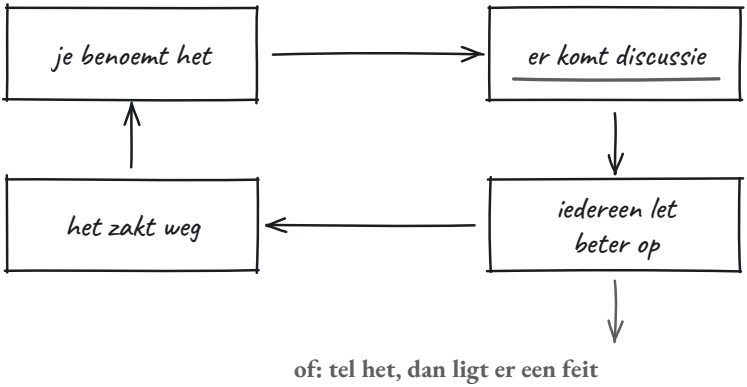
Een frats is geen objectieve fout, en daardoor ligt hij sociaal gevoelig. Een fout kun je aanwijzen: hier hoorde iets te gebeuren, het gebeurde niet, dat is na te lopen. Een frats kun je alleen benoemen als indruk — *het lukt net niet*, *het loopt niet lekker*, *er zit iets scheef*. En een indruk nodigt uit tot een tegenindruk.

Dus gebeurt er een van drie dingen zodra je het aan de orde stelt, en geen van de drie helpt:

- **er komt discussie**, want er ligt geen feit op tafel maar een oordeel, en tegen een oordeel is een ander oordeel een geldig antwoord
- **er komt onbegrip**, want de ander hoort iets over zichzelf dat hij nergens terugziet — en dat is niet onwil, er ís niets te zien
- **er komt tijdelijke onderdrukking van de symptomen**: iedereen let twee weken beter op, het lijkt beter, en daarna komt het terug. Er is niets opgelost, want er is niets veranderd

Het echte verlies zit niet in het ongemakkelijke gesprek. Het zit erin dat je na afloop **niets meetbaars hebt dat je systematisch kunt oplos-**

sen. Je hebt een gevoel gedeeld en een belofte gekregen. Dat is geen aangrijpingspunt, en over een maand sta je er weer.



Het gesprek zonder feit draait rond. De uitgang is tellen.

Hier staat dus dezelfde zet als in hoofdstuk 4, en hij is hier geen stijlkeuze maar de enige manier om het onderwerp bespreekbaar te maken: **maak er eerst een feit van.** Niet *"je komt vaak net te laat"*, maar *"van de twintig keer waren er zeven anders dan de andere dertien, en dit was het verschil."* Dat is geen verwijt meer. Het is een waarneming, en over een waarneming valt niet te twisten — daar valt alleen naar te kijken.

Tellen is dus niet alleen hoe je een frats vindt. **Tellen is wat het gesprek erover mogelijk maakt.**

Bij een machine is die sociale prijs nul, en dat is een groter voordeel dan het lijkt. Je kunt tegen een agent zeggen wat er net niet klopt, twintig keer achter elkaar, zonder dat iemand zich aangesproken voelt en zonder dat er een verhouding op het spel staat. Daardoor kun je in een middag doen wat in mensenwerk maanden kost — en daardoor zag ik

het patroon dat ik uit die andere wereld kende opeens in het klein terug, en versneld.

Hoe je grip krijgt op iets dat onder de radar zit

Eerst de scherpste formulering van het verschil, want die bepaalt de hele aanpak:

> **Fouten kun je meten. Fratsen zitten onder de radar.**

Een fout meldt zich. Je kunt hem tellen, sorteren, en er een grafiek van maken die naar beneden loopt. Een frats meldt zich niet, en daarom helpt geen enkel instrument dat op meldingen wacht.

Wat niet werkt is beter opletten. Dat is het antwoord dat iedereen als eerste geeft, en het faalt om dezelfde reden in allebei de werelden: fratsen zitten onder de drempel van de aandacht, en aandacht is precies waar in de aanloop het minst van is. Erger nog — je oordeel is juist het filter dat ze heeft weggelaten. Wie beter oplet, let beter op dezelfde dingen.

Er zijn vijf zetten die wél grip geven, en ze staan in deze volgorde omdat elke volgende de vorige nodig heeft.

Eén: herhaal en vergelijk. Dit is het enige detectieprincipe dat werkt. Een frats heeft geen eigen kenmerk waaraan je hem in één geval herkent — zijn kenmerk is **verschil tussen keren die hetzelfde hadden moeten zijn.** Eén uitkomst zegt niets. Twintig uitkomsten naast elkaar wijzen hem aan, en je hoeft er niet eens slim voor te zijn.

Twee: leg vast hoe het eruitziet als er niets aan de hand is. Zonder ijkpunt is elk verschil evenveel waard, en dan verdrink je in ruis. Mét ijk-

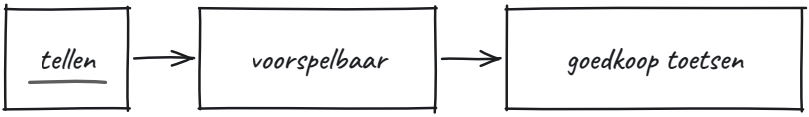
punt is afwijking het signaal. Dat ijkpunt hoeft niet mooi te zijn; het moet alleen bestaan en niet meebewegen.

Drie: tel, en oordeel niet. Noteer wat er anders was, niet of het erg was. De ernst schat je later in; op het moment van noteren is je oordeel juist het probleem, want dat is het filter dat ze onzichtbaar hield.

Vier: verklein de speelruimte. De meeste fratsen verdwijnen niet doordat iemand zich meer inspant, maar doordat de situatie minder ruimte laat om af te wijken. Dezelfde volgorde, dezelfde vorm, dezelfde woorden. Wat er dan nog anders gaat, gaat anders om een echte reden.

Vijf: laat de drempel zakken. En hier zit het echte antwoord. Fratsen bestaan omdat er een grens is waaronder niemand ingrijpt — dat is geen slordigheid maar een economische keuze, en zolang het werk handmatig ging was die keuze terecht. **Je krijgt pas grip als die grens omlaag kan, en die kan alleen omlaag als controleren goedkoop is.**

Daarmee valt dit hoofdstuk samen met de rest van het boek. Snelheid, automatische controles, een lus die zichzelf nakijkt: dat is niet netheid. Dat is wat de drempel verlaagt tot onder het niveau waarop fratsen leven.



beter opletten staat er niet bij

Niet beter opletten, maar zichtbaar maken, voorspelbaar maken en goedkoop toetsen.

De frats zelf, of iets ervoor

Dan de vraag die alles bepaalt, en het is de vraag die ik mezelf inmiddels bij elke frats stel:

> **Los je de frats op, of los je iets op dat eraan voorafgaat, zodat er minder fratsen ontstaan?**

Want een frats staat er zelden op zichzelf. **Er ligt bijna altijd een oorzaak vóór.** Wie net te laat binnenkomt heeft misschien geen wekker, of een bus die niet aansluit, of een avond die niet goed eindigde. Een model dat een bibliotheek verzint, heeft geen lijst gekregen van wat er wél bestaat. Het zichtbare dingetje is het uiteinde van iets dat eerder begon.

En dat betekent dat je **terug** moet, en dat voelt verkeerd. Alles in een project duwt vooruit: de volgende stap, de volgende taak, de volgende week. Teruggaan naar iets dat al gepasseerd is, ziet eruit als oponthoud. **Het is het tegenovergestelde van vooruitgaan, en het is toch het enige dat werkt** — want vooruit los je dit geval op, en terug los je de soort op.



terug is de enige richting die een hele soort wegneemt

Niet het geval, maar wat ervoor ligt. Daar verdwijnt een hele soort.

Er zijn drie hoogtes waarop je kunt ingrijpen, en het verschil tussen die drie is het hele verhaal:

	wat je aanpakt	wat je eraan overhoudt
de frats	dit ene geval	het kost de volgende keer weer evenveel
de oorzaak ervoor	waaróm dit soort gevallen ontstaat	een hele soort verdwijnt
wat die oorzaak toelaat	de situatie die er ruimte voor liet	ze ontstaan niet meer

De eerste hoogte is niet waardeloos. Hij koopt je tijd, en soms is tijd precies wat je nodig hebt om aan kijken toe te komen. Maar als je daar blijft, ben je aan het dweilen: elk geval kost hetzelfde, er komt niets bij, en over een maand doe je het nog steeds.

De derde hoogte haal je niet altijd. Sommige oorzaken liggen buiten je bereik — bij een ander, bij een systeem dat niet van jou is, bij een gewoonte die ouder is dan jij. Dan is de tweede goed genoeg, en het is eerlijker om op te schrijven dat je daar bent blijven staan dan te doen alsof je bij de wortel bent geweest.

Dit is dezelfde zet als *zoek de sleutel, niet de deuren*, verderop in dit boek: niet het zichtbare geval aanpakken maar de voorwaarde eronder. En het is ook waarom *verklein de speelruimte* hierboven zo hoog staat. Minder ruimte om af te wijken is niets anders dan een voorliggende oorzaak wegnemen, voordat er iets uit ontstaat.

Een frats uit dit boek zelf

Tijdens het maken hiervan stond er weken lang geen enkel vetgedrukt woord vet.

De letter waarin dit boek gezet is, bestaat in twee sneden; de ene heeft geen vette variant. Het zetprogramma koos die, en viel bij vet stilletjes

terug op gewoon. Geen melding, geen waarschuwing, geen verschil dat je ziet als je niet weet dat je ernaar moet kijken. Elke nadruk in elk hoofdstuk was weg, en het zag er alleen uit als een boek waarin de nadruk niet zo goed werkte.

Dat is een frats in zijn zuiverste vorm. Klein, stil, geen fout, en aanwezig op elke bladzijde. Hij is niet gevonden door beter te kijken maar doordat iemand een bladzijde naast een andere legde en zich afvroeg waarom het zo grijs oogde.

Zelf doen · Groepeer op soort in plaats van op project

Wat je doet. In `opleverpunten/` staan zes punten uit vier opleveringen. Per oplevering zijn het er één of twee, en dan lijkt het een kwestie van pech. Laat ze groeperen op soort en kijk of dat nog steeds zo is. Reken op vijf minuten.

TYP DIT

> Groepeer de opleverpunten in opleverpunten/ naar soort in plaats van naar opdracht, en zeg welk soort het vaakst terugkomt. Zet er per punt bij of het hersteld is of nog openstaat.

Wat je ziet. Drie soorten. **En dan valt er iets op dat per project onzichtbaar was: alles wat nog openstaat, zit in dezelfde soort.** Wat lelijk is — een kitrand, een zijwang niet in lijn — wordt ter plekke hersteld. Wat *ontbreekt* moet besteld, gepland en teruggekomen worden, en dat blijft dus liggen. Geen van deze zes is een fout; samen houden ze twee opleveringen op.

WAT ER TERUGKWAM

```
ontbreekt      3
  ventilatierooster keuken      hersteld
  antislip treden               open
  hemelwaterafvoer              open
afwerking      2
  kitrand slaapkamer 2          hersteld
  zijwang dakkapel              hersteld
maat of norm   1  (leuning te laag, hersteld)
```

Waar het misgaat. Laat hem de soorten zelf verzinnen en geef ze niet op. Geef je de indeling mee, dan bevestigt hij jouw beeld en vindt hij niets nieuws. **Zet de status erbij.** Zonder *hersteld* of *open* had je alleen geteld hoe vaak iets voorkomt, en dan was het patroon — het openstaande zit allemaal in één soort — er niet uit gekomen. **En reken op een categorie "overig".** Die is niet mislukt; hij is de plek waar volgende keer een nieuw soort uit tevoorschijn komt.

EN DAN ZO

Kijk nu of er een verschil is in hoe lang de punten per soort openstaan, en wat dat verklaart.

EN DAN DIT

```
afwerking      gemiddeld 2 dagen tot herstel
maat of norm   1 dag
ontbreekt      1 hersteld na 5 dagen, 2 nog open
                na 30+ dagen
verschil: afwerking doe je met wat er in de bus
ligt, ontbreekt moet je bestellen
```

En dit geldt ver buiten dit onderwerp

- **Een keuken waar één stap in de voorbereiding wisselt** levert gerechten die allemaal net iets anders zijn. Geen enkel gerecht is fout. De zaak wordt er alleen nooit voorspelbaar van.
- **Een team waar de overdracht meestal mondeling gaat** werkt prima tot iemand ziek is. Dan blijkt dat het al maanden net niet klopte, alleen merkte niemand dat omdat het steeds werd rechtgezet.
- **Een formulier waarin één veld dubbelzinnig is** levert een paar procent verkeerde invullingen op. Bij honderd per jaar is dat ruis. Bij honderdduizend is het een probleem met een eigen afdeling.
- **Een machine die zelden vastloopt maar vaak nét anders doseert** produceert niets dat afgekeurd wordt, en toch haalt de partij de specificatie niet.

In alle vier de gevallen is er geen schuldige en geen incident. Er is een patroon dat onder de drempel van ingrijpen ligt, en dat pas zichtbaar wordt als je het telt.

Probeer: laat een taak die je vaak uitvoert twintig keer draaien met dezelfde invoer, en vraag om elk verschil tussen de uitkomsten — ook de kleine, vóóral de kleine — zonder oordeel over de ernst. Je zoekt geen mislukkingen maar variatie: wat bij twintig keer verschilt, verschilt bij duizend keer duizend keer. Vraag uitdrukkelijk om géén oordeel, anders krijg je de grote dingen terug en die had je al gezien.

Tellen maakt ze zichtbaar, en zichtbaar is nog niet weg. De eerste neiging is dan om het geval dat je ziet op te lossen, en dat is precies een stap te ver naar voren.

En ga er niet meteen op af. Vraag voor de frats die het vaakst voorkwam eerst welke drie dingen eraan voorafgaan en hem mogelijk maken, wat er zou veranderen als die weg waren, en welke van de drie de meeste gevallen wegneemt — nog zonder oplossing. Een frats oplossen kost de volgende keer weer evenveel; zijn oorzaak wegnemen laat een hele soort verdwijnen. Haal je de derde hoogte niet, schrijf dat dan op: doen alsof je bij de wortel was, is hoe dezelfde frats over een half jaar opnieuw onderzocht wordt.

Pas als je weet waar je ingrijpt, heeft een controle zin. En die moet goedkoop zijn of hij wordt uitgezet.

Tip: vraag daarna om een controle die het vangt en die mínder kost dan de handeling zelf, met erbij wat hij *niet* vangt. Een dure controle wordt uitgezet zodra het druk wordt, en weten wat een toets niet vangt is het verschil tussen een controle en een vals gevoel van veiligheid. Negen van de tien is genoeg om te beginnen — wachten op een toets die alles vangt is hoe je eindigt met geen enkele toets.

> **Wat dit goedkoper maakte:** *elke frats die zichtbaar werd, hoefde niet meer door mij te worden opgevangen — en pas daarna had het zin om meer tegelijk te laten lopen.*

13. Fouten zijn je beste leraar

Acht opdrachten, koud gegeven. Eén ervan kwam er door.

Het was een eerlijke proef: acht kleine, gewone taken in een omgeving met eigen afspraken, gegeven aan een model zonder hulp, zonder voorbeelden, zonder dat er iemand meekeek. Zeven van de acht werkten niet. **Twaalf en een half procent.**

De reflex die daarbij hoort is: *het model kent dit niet.* Die reflex is begrijpelijk, hij is deels waar, en hij leidt nergens heen. Want als dat het antwoord is, dan is er ook maar één oplossing — wachten tot jouw manier van werken vaak genoeg ergens beschreven staat om in een training te belanden — en dat is geen oplossing maar een wens.

En dat geldt niet alleen voor code. Elk vak heeft zijn eigen afspraken: een dossier dat een vaste opbouw heeft, een offerte waarin de volgorde vastligt, een formulier waarin één veld iets anders betekent dan het lijkt. Wie een model in zo'n omgeving loslaat, krijgt hetzelfde percentage en dezelfde soorten fouten.

Dus heb ik iets anders gedaan. Ik heb de zeven fouten op een rij gezet en gekeken of ze op elkaar leken.

Zeven fouten, drie soorten

Ze leken op elkaar. Ze leken zelfs zó op elkaar dat de hele stapel in drie bakjes paste.

Een verzonnen onderdeel. Het model gebruikte dingen die hier niet bestaan, maar wel in de omgevingen die het goed kent. Namen die logisch klinken, die ergens anders echt bestaan, en die hier niets zijn. Dit was de grootste bak, en het is ook de grootste categorie in het foutenlogboek dat ik al maanden bijhield.

Een afspraak die nergens anders zo is. Eén kleine schrijfwijze die hier anders is dan in de twee omgevingen die er het meest op lijken. Niet fout te noemen, niet te raden, en precies daarom telkens mis.

Een vorm die per geval verschilt. De derde soort was de vervelendste, en niet omdat hij moeilijk is. Twee functies die naast elkaar in hetzelfde bestand staan, willen hun gegevens op een verschillende manier aangeleverd krijgen. Dat is niet te raden en nauwelijks te onthouden — en dat is dus geen fout van het model maar een fout van mij.

de soort fout	hoe vaak
een verzonnen onderdeel	de grootste bak
een afwijkende afspraak	twee van de zeven
een vorm die per geval verschilt	en die is van mij

Drie soorten fouten, en hoe vaak elke soort voorkwam. Sorteren op aantal, niet op hoe interessant het is.

Toen ik ze zo naast elkaar had staan, kwam er een zin uit die ik niet had verwacht:

Drie regels zouden zeven van de zeven fouten hebben voorkomen.

Het waren alle zeven keer drie afspraken die nergens stonden waar ze gelezen konden worden.

Het is geen kennisprobleem maar een aanwezigheidsprobleem

Hier zit het deel dat overdraagbaar is, en het geldt ver buiten dit voorbeeld.

Alle drie die afspraken stonden op schijf. Correct opgeschreven, in documentatie die klopte en die niemand hoefde te zoeken. **Ze stonden alleen niet in beeld op het moment dat de code ontstond.**

Dat is een ander soort tekort dan "hij weet het niet". Hij wist het inderdaad niet, maar de reden dat hij het niet wist is niet dat het nergens stond — het is dat er niets was dat het op het juiste moment aanreikte. En dat is te repareren zonder dat er iets slimmer hoeft te worden.

Dit onderscheid is het waard om vast te houden, want het beslist waar je je geld en je tijd in steekt:

- **een kennisprobleem** los je op met beter materiaal, meer training, een groter model — traag, duur, en meestal niet van jou afhankelijk
- **een aanwezigheidsprobleem** los je op door het juiste stuk tekst op het juiste moment mee te sturen — goedkoop, vandaag, en helemaal in eigen hand

Bijna alles wat eruitziet als het eerste, is bij nader inzien het tweede.

De lijst die van je eigen fouten gemaakt is

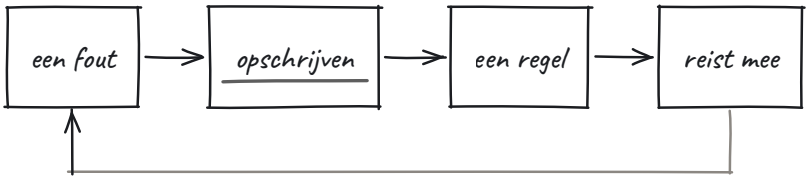
Wat daaruit volgt is bijna te simpel om op te schrijven, en dat is precies waarom bijna niemand het doet.

Je houdt een logboek bij van elke fout die er gemaakt wordt. Wat er geschreven werd, waarom dat fout was, en wat de goede vorm is. Dat kost per fout een halve minuut, op een moment dat je toch al naar die fout aan het kijken bent.

Dan sorteer je die lijst — en hier zit de enige stap waar iets in te kiezen valt.

Sorteer op hoe vaak, niet op hoe interessant. De verleiding is groot om te beginnen bij de fout die je het meest verbaasde, want die is het leukst om over na te denken. Maar de fout die je verbaasde kwam één keer voor. De saaie fout die je drie keer per dag ziet, is degene die je tijd opeet.

En dan stuur je de bovenste regels mee, elke keer: de vijf of tien regels waar het werkelijk op stuk gaat, plus een volledige lijst van wat er echt bestaat — want tegen verzonnen onderdelen is een lijst het hele antwoord.



Van fout naar regel naar niet-meer-voorkomen: elke ronde maakt de volgende ronde korter.

Bedenk, fix en bewaak

Er is een korte manier om dit hele hoofdstuk aan een agent door te geven, en hij bestaat uit drie woorden.

> *"Bedenk, fix en bewaak."*

Bedenk — verzin hoe het opgelost kan worden, en zeg welke aanpak je kiest. **Fix** — voer het uit. **Bewaak** — zorg dat het niet terug kan komen: een test, een controle, een regel die meereist.

De eerste twee doet iedereen. De derde is waar het op stukloopt, en zonder die derde is de lus niet rond: je hebt dan een fout opgelost in plaats van een soort fouten.

Wat deze drie woorden vervangen, is een heel gesprek. Zonder die afspraak beschrijf je elke keer opnieuw wat je verwacht — en dat is precies de uitleg die bij elke ronde herhaald moet worden. Met die afspraak staat er één regel, en je krijgt het hele rondje terug.

En dat is ook waar ze voor bedoeld zijn. **Niet dat hij het één keer doet, maar dat hij het blijft doen** — zodat de volgende keer dat hetzelfde langskomt, er al iets staat dat het tegenhoudt en jij er niet meer naar hoeft te kijken.

De gate die je bouwt, vangt jou

Dit is het meest ongemakkelijke wat ik over dit hoofdstuk kan zeggen, en het is ook het meest bruikbare.

Elke controle die ik heb gebouwd, is uiteindelijk afgegaan op mijn eigen werk. Niet op dat van een ander, niet op dat van een agent

— op dat van mij, de persoon die de controle had bedacht en precies wist waar hij op lette.

Het patroon is steeds hetzelfde. Ik bouw een controle omdat ik een soort fout heb zien gebeuren. Hij staat er. Ik werk verder, in de overtuiging dat ik díe fout nu niet meer maak, want ik ken hem. En een week later gaat hij af, op mij.

Een greep uit wat dat opleverde, allemaal bij het maken van dit boek:

- een controle op tekstverlies vond dat mijn eigen zetter honderdzevenentachtig regels weggooide
- een grens op de lengte van een stukje voorbeeldtekst hield veertien keer mijn eigen tekst tegen
- een verbod op productnamen ving een productnaam in een passage die ik zelf had geschreven
- een controle op namen die door elkaar lopen, vond twee namen die ik zelf had verzonnen
- en een ander systeem, waar ik niets over te zeggen had, vond vier dingen die hier al maanden scheef stonden

Nul keer is er iets afgegaan op iemand anders. Dat komt niet doordat anderen beter werken. Het komt doordat ik degene ben die hier het meeste doet, en dus ook degene die de meeste fouten maakt.

Waarom dit ertoe doet: het verandert waarvoor je een controle bouwt. Niet om iemand te betrappen, en ook niet om een agent in toom te houden. **Je bouwt hem tegen jezelf over een week** — tegen de versie van jou die haast heeft, die dit stuk al kent, en die daarom niet meer kijkt.

Dat is ook het antwoord op de tegenwerping dat het te veel gedoe is voor de fouten die jij toch niet maakt. Je maakt ze wel. Je bent alleen niet aanwezig op het moment dat het gebeurt, en dat is precies wat de vorige paragraaf zegt.

Het cijfer waar je naar kijkt, is het verkeerde

Nog iets dat die acht taken opleverden, en het is misschien het nuttigste van alles.

Twaalf en een half procent klinkt als een slecht cijfer, en dat is het ook — als je meet hoe vaak het in één keer goed gaat. Maar waarom zou je dat meten?

In dit geval was er een controle die streng is, die snel is, en die uitgebreid vertelt wát er mis is. Een poging kost dan bijna niets, en elke mislukte poging levert een zin op waar je iets mee kunt. **De vraag is dus niet hoe vaak het in één keer lukt, maar hoe vaak het binnen drie pogingen lukt mét de foutmelding erbij.** Dat is een heel ander getal, en het is het getal dat er werkelijk toe doet, want zo werkt het in de praktijk ook.

Dit is dezelfde gedachte als in hoofdstuk 10, van de andere kant bekeken. Als proberen goedkoop is, is *in één keer goed* een luxe en geen eis. En als je toch op *in één keer goed* stuurt, ga je dingen verbeteren die er niet toe doen.

En dit gaat niet over programmeren

De vorm hiervan zit overal waar mensen — of machines — hetzelfde soort werk herhalen.

- **Een keuken die klachten bijhoudt op soort** in plaats van op hoe erg ze waren, ontdekt na een maand dat driekwart over hetzelfde gaat, en dat het vaak één handeling in de voorbereiding is.
- **Een docent die de tien fouten opschrijft die elke leerling maakt** hoeft die tien nooit meer tien keer los uit te leggen. De lijst gaat vooraf mee, en de les gaat over de rest.
- **Een administratie waar formulieren op dezelfde regel stuklopen** heeft geen strengere controle nodig maar een duidelijker formulier. De fout zit niet bij degene die hem maakt.
- **Een ploeg die na elke storing één regel opschrijft** — wat het was, wat we dachten, wat het bleek — heeft na een half jaar iets dat geen enkele cursus kan vervangen, omdat het over deze installatie gaat en niet over installaties in het algemeen.

In alle vier gevallen is de winst niet dat er minder fouten gemaakt worden. De winst is dat dezelfde fout de tweede keer minuten kost in plaats van uren, en de derde keer niet meer voorkomt.

Zelf doen · Voeg de derde stap toe die altijd wordt vergeten

> **Wat je doet.** Er staan drie incidenten in `kam/incidenten/`, alle drie netjes met een maatregel erbij. Bedenken en repareren zijn dus gebeurd. De vraag van dit hoofdstuk is de derde stap: **wie zorgt dat het niet terugkomt?** Reken op tien minuten.

TYP DIT

> Loop de incidenten in kam/incidenten/ na. Kijk per incident of de maatregel ergens is vastgelegd als een terugkerende controle, of dat hij alleen als tekst in het incident staat. Schrijf daarna een werkwijze voor het afhandelen van een incident, met als verplichte laatste stap welke controle voorkomt dat dit soort incident terugkomt.

Wat je ziet. Drie incidenten, drie maatregelen, **en nul controles.** Snoeren ophangen, afzetting verbreden, handschoenen verstrekken: alle drie verstandig, alle drie eenmalig, en geen ervan gebeurt volgende maand vanzelf opnieuw. **En er is er één niet gemeld** — dat is een vierde bevinding die je er gratis bij krijgt.

WAT ER TERUGKWAM

```
2026-03-18 verlengsnoer   maatregel: ja  controle: nee
2026-04-16 dakpan         maatregel: ja  controle: nee
2026-04-25 snijwond       maatregel: ja  controle: nee
                          (en niet gemeld)
maatregelen zonder controle: 3 van de 3
```

Waar het misgaat. Vraag niet of de maatregel goed is. Dat is een vakinhoudelijk oordeel en daar heb je hier niets aan; de vraag is of hij blijft bestaan als niemand eraan denkt. **Maak de laatste stap verplicht en niet aanbevolen.** Is hij optioneel, dan wordt hij overgeslagen op precies de dag dat het druk is — en dat is dezelfde dag waarop het incident gebeurde. **En laat "geen controle nodig" een geldig antwoord zijn, mits er een reden bij staat.** Anders verzint iemand een controle om van het veld af te zijn, en dan heb je een controle die niemand draait.

EN DAN ZO

> Voeg de laatste stap toe aan de werkwijze en laat zien wat er dan voor het snoer-incident had moeten gebeuren.

EN DAN DIT

```
incident-afhandelen, stap 4 (verplicht):
  welke terugkerende controle vangt dit voortaan?
  antwoord mag ook zijn: geen, want <reden>

toegepast op 2026-03-18:
  controle: "loopt er een snoer over een pad?"
  toegevoegd aan de rondgang op de bouwplaats
```

Niet elke les komt uit iets dat misging

Eén aanvulling, want anders wordt dit hoofdstuk eenzijdig.

Er is een soort kennis die er precies zo uitziet als een mislukking en het niet is: het besluit om iets *niet* te doen. Je hebt iets onderzocht, je hebt eraan gewerkt, en de uitkomst is dat het niet moet gebeuren.

Dat is een resultaat, geen mislukking. Er is werk gedaan en er is iets vastgesteld. Het hoort dus op *afgerond* te staan en niet op *afgeblazen* — en het hoort te worden opgeschreven, met de reden erbij, want anders komt hetzelfde voorstel over een half jaar opnieuw langs en wordt hetzelfde onderzoek opnieuw gedaan.

Dat is dezelfde regel als hierboven, alleen op iets dat niet pijn deed. En daardoor wordt hij nog vaker vergeten.

Probeer: zet vanaf nu elke fout in één bestand, op het moment dat je hem oplost, met drie velden — wat er stond, waarom dat fout was, wat de goede vorm is. Laat na een regel of twintig het logboek groeperen naar soort, gesorteerd op aantal, met per soort één regel die hem zou voorkomen. Je zoekt niet de interessantste fout maar de vaakste; dat zijn zelden dezelfde. Schrijf op wat er werkelijk gebeurde en niet wat je ervan

vond — een logboek met meningen erin is over een maand niet meer te sorteren.

Een logboek dat je niet meestuurt, is een archief. De volgende stap is het onderweg meegeven — en dan pas werkt het.

Stuur ze daarna mee in plaats van ze uit te leggen: maak van de regels die je vond één blok van hooguit tien regels dat bij elke opdracht meegaat, met de volledige lijst erbij van wat er écht bestaat. Tegen een verzonnen onderdeel is zo'n lijst het hele antwoord. Kort houden is geen zuinigheid maar noodzaak: een blok van drie bladzijden wordt gelezen als achtergrond, tien regels worden gelezen als instructie.

Meet daarna opnieuw. Maar dan het getal dat er werkelijk toe doet, want het cijfer waar je vanzelf naar kijkt is het verkeerde.

En meet het getal dat ertoe doet: niet hoe vaak het in één keer goed gaat, maar hoe vaak het binnen drie pogingen goed gaat mét de foutmelding erbij. In de praktijk krijgt niemand één poging, en sturen op *in één keer goed* verbetert dingen die er niet toe doen. Dit werkt alleen als je foutmeldingen bruikbaar zijn — zijn ze dat niet, dan meet je je meldingen en niet je model, en dan weet je meteen waar het echte werk ligt.

> **Wat dit goedkoper maakte:** *elke fout werd één keer gemaakt in plaats van telkens opnieuw, en wat hij opleverde reisde voortaan mee naar de volgende.*

14. Mijn gereedschap loog tegen me

De foutmelding was een keurige zin. Hij noemde de regel, hij noemde het probleem, en hij stelde een oplossing voor. Hij was op twee punten onwaar.

Wat er stond, kwam erop neer: *die naam ken ik niet — hij hoort in dat andere bestand, voeg dat toe.* Dat andere bestand stond drie regels hoger al toegevoegd. En de naam hoorde daar helemaal niet in: hij bestond nergens, hij was verkeerd gespeld.

Twee onwaarheden in één zin, en allebei van het soort dat je niet ziet als je niets anders hebt.

Waarom ik hier meer van schrok dan het lijkt

Het is een kleine fout. Een verkeerd gespelde naam, en een foutmelding die de verkeerde kant op wijst. Zoiets kost een mens een halve minuut.

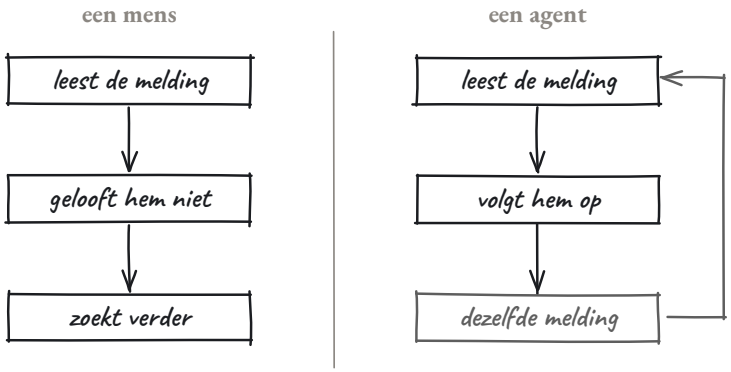
Maar kijk wat er gebeurt als de lezer geen mens is.

Een agent leest die zin en doet precies wat er staat: hij voegt toe wat er al stond. Hij vertaalt opnieuw. Hij krijgt woord voor woord dezelfde melding terug. En dan is hij op, want de melding heeft hem het enige feit afgenomen dat hij nodig had — dat die naam niet bestaat. Hij zit in een lus die niet één keer misgaat maar elke keer, en hij heeft geen enkele reden om eruit te stappen.

Een mens verliest hier minder, en dat komt niet doordat hij slimmer is. Hij ziet dat bestand staan waar de melding zegt dat het ontbreekt, en dan gelooft hij de melding niet meer. Dat is geen redenering, dat is wantrouwen — een reflex, opgebouwd uit jaren meldingen die onzin bleken.

Een agent heeft die reflex niet. Hij kan hem ook niet krijgen, want hij begint elke sessie opnieuw. Wat jij "ervaring met dit gereedschap" noemt, is precies het deel dat niet meereist.

Daarmee verschuift er iets dat ik niet had zien aankomen. Zolang jij het werk doet, is een misleidende melding hinder. Zodra een agent het werk doet, **is de kwaliteit van je meldingen een bovengrens aan wat er mogelijk is.** Je instrumenten wegen zwaarder dan je oordeel, want er is geen oordeel meer dat ze kan overrulen.



Een mens ziet dat de melding niet kan kloppen en stopt. Een agent volgt hem en komt terug op hetzelfde punt.

Vier meldingen die ik geloofd heb

Dit was niet mijn eerste. In de maanden ervoor, in ander werk, stonden er nog vier — en ze hebben elk hun eigen soort onwaarheid.

De melding die over het verkeerde onderwerp gaat. Een taak viel om met een fout over autorisatie: je mag dit niet. Er ontbrak een instelling. De melding was volkomen waar — er wás geen geldige toegang — en hij wees naar rechten terwijl het probleem configuratie was. Dit is de gemeenste soort, want er valt niets aan te weerleggen. Hij is waar, en hij wijst de verkeerde kant op.

De melding die drie stappen te ver staat. Honderden tests vielen tegelijk om, en het las als een codebase in puin. De omgeving was halverwege opgezet: de structuur wel, de basisgegevens niet. Zonder die gegevens ontbrak er een sjabloon; zonder dat sjabloon mislukte het aanmaken van een testgebruiker; en dus kreeg ik een melding over het aanmaken van gebruikers. Die melding noemde gegevens nergens. Eén regel navragen zei het in één keer — en het duurde een uur voordat ik die regel stelde, omdat ik de melding aan het onderzoeken was in plaats van de omgeving.

De meting die naar beneden loog. Een telling die we in de gaten hielden ging omlaag, en omlaag was goed: minder werk. Er was niets goedkoper geworden. Een samengevoegde uitvraag was tegen een maximum aangelopen en kapte stilletjes af, waardoor een paar onderdelen als gratis werden gerekend. **Sindsdien is een telling die onder de verwachting zakt net zo verdacht als een die stijgt**, en dat is een regel die zich niet vanzelf laat bedenken.

De twee theorieën die geen van beide bestonden. Het meetgereedschap was gebouwd om twee nette verklaringen te bewijzen voor iets dat traag was. Allebei bleken ze meetfouten — artefacten van hoe er gemeten werd, niet van wat er gebeurde. Het gereedschap had gelijk gekregen als niemand het nog een keer op een andere manier had gecontroleerd.



Vier soorten onwaarheid: wijst verkeerd, staat te ver, liegt naar beneden, bevestigt zichzelf.

Wat deze vijf — de opener en deze vier — gemeen hebben

Alle vijf werkten precies zoals ze gebouwd waren.

Ze faalden in de vertaling van een waarneming naar een zin. Het instrument zag iets waars en zei iets anders — korter, behulpzamer, en net verkeerd genoeg om je een kant op te sturen. En de behulpzaamheid is het gevaarlijke deel: een melding die alleen *er is iets mis* zegt, laat je zoeken. Een melding die een oplossing voorstelt, laat je die oplossing uitvoeren.

Daar komt de regel vandaan waar dit hoofdstuk op uitkomt, en hij is korter dan hij verdient:

Een melding is bewijsmateriaal, geen uitspraak. Hij vertelt je wat het instrument heeft waargenomen. Wat er aan de hand is, moet je er nog steeds zelf bij zoeken.

En dit gaat niet over software

De vorm hiervan zit in elk vak waar iemand op een meter kijkt.

- **Het lampje op het dashboard** dat aangaat bij een sensor die stuk is. De motor is in orde, de meter niet, en je staat aan de kant van de weg.
- **Een uitslag die "niet aangetoond" zegt** en gelezen wordt als "niet aanwezig". Beide zinnen zijn waar over iets anders, en het verschil beslist wat er daarna gebeurt.
- **Een thermometer die hangt waar de zon op staat.** Hij liegt niet. Hij meet keurig de temperatuur op die plek, en die plek is de verkeerde.
- **Een rapportage die alleen laat zien wat er is ingevuld.** Wat niemand invulde, bestaat op dat scherm niet — en daardoor lijkt de afdeling die niets bijhoudt de afdeling zonder problemen.

In alle vier gaat het niet mis bij de meting maar bij de zin die je erover vertelt. En in alle vier is het herstel hetzelfde: zoek één onafhankelijke waarneming die hetzelfde zou moeten zeggen.

Wat dit verandert nu er agenten meelezen: een organisatie kan jarenlang draaien op meters die scheef hangen, omdat de mensen ze onbewust corrigeren. Ze weten dat die ene meter altijd te hoog staat. Zet er een agent op en de correctie is weg, want die zat nergens opgeschre-

ven. Dan komt in één keer boven water hoeveel van je betrouwbaarheid in hoofden zat.

Zelf doen · Vraag waar het overzicht over gaat

Wat je doet. `kostenbewaking/` ziet eruit als het overzicht van de lopende opdrachten. Het klopt, het telt op, en het is netjes bijgehouden. De vraag van dit hoofdstuk is niet of de getallen kloppen maar **waar ze over gaan.** Reken op vijf minuten.

TYP DIT

> Vergelijk welke opdrachten er in kostenbewaking/ staan met welke er in opdrachten/ staan, en zeg wat dat betekent voor een totaal dat je uit kostenbewaking haalt.

Wat je ziet. Zes van de vijftien. Het overzicht is compleet in zijn eigen termen en gaat over minder dan de helft van het werk. Elk totaal dat je eruit haalt is dus juist én misleidend, en er staat nergens bij dat het een selectie is. **Twee van de negen ontbrekende opdrachten hebben wel een openstaande crediteurenfactuur** — daar wordt dus geld uitgegeven dat in geen enkel overzicht terugkomt.

WAT ER TERUGKWAM

```
opdrachten              15
in kostenbewaking        6
niet in kostenbewaking   9
  NLOP-2026-003, NLOP-2026-006, NLOP-2026-008,
  NLOP-2026-009, NLOP-2026-011, ... en 4 andere
wel een crediteur, geen kostenbewaking:
  NLOP-2026-006, NLOP-2026-011
```

Waar het misgaat. Vraag niet of de getallen kloppen. Die klopten; dat was juist het probleem. De bruikbare vraag is waarover ze gaan en wat er buiten valt. **Vergelijk met de bron en niet met een ander over-**

zicht. Twee overzichten die dezelfde selectie hebben, bevestigen elkaar tot in het oneindige. **En let op dat een ontbrekende regel er niet uitziet als een ontbrekende regel.** Er staat geen gat, geen nul, geen waarschuwing — er staat gewoon iets minder, en dat ziet er precies zo uit als iets compleets.

EN DAN ZO

> Zet er een regel bij die zegt over hoeveel van de opdrachten dit overzicht gaat, en laat dat voortaan altijd meekomen.

EN DAN DIT

```
KOSTENBEWAKING
  dekking: 6 van de 15 opdrachten (40%)
  buiten beeld: 9 opdrachten, waarvan 2 met
                een openstaande crediteur
  laatste bijwerking: onbekend
```

Wat je eraan doet

Drie dingen, en het derde is het enige dat blijft werken als je er niet bij bent.

Wantrouw een melding die een oplossing voorstelt méér dan een die dat niet doet. Precies andersom dan hij bedoeld is. Een voorstel is een gok van het instrument over jouw situatie, en dat is de plek waar het misgaat.

Zoek een tweede waarneming vóór je iets verandert. Niet nog een keer dezelfde meting, maar een andere weg naar hetzelfde feit. Twee instrumenten die het oneens zijn, is informatie; één instrument dat iets beweert, is een vermoeden.

En repareer de melding, niet alleen de fout. Dit is het deel dat overgeslagen wordt, want op het moment dat je de oorzaak vindt is je probleem opgelost en voelt de melding niet meer als jouw werk. Toch is dat het enige moment waarop je precies weet wat hij had moeten zeggen. Een uur later is dat weg, en dan staat hij er nog steeds — klaar om de volgende, die geen mens is, dezelfde kant op te sturen.

Probeer: heb je net een misleidende melding doorzien, repareer hem dan vóór je verder werkt. Zoek waar die tekst gemaakt wordt en bedenk wat hij had moeten zeggen in drie gevallen: als het instrument het zeker weet, als het iets vermoedt, en als het niets weet. Een instrument mag best een suggestie doen, maar alleen als het dat onderscheid maakt — de meldingen die schade doen zijn die welke een vermoeden opschrijven in de toon van een feit. Doe het meteen: de kennis van wat hij had moeten zeggen bestaat precies één keer, namelijk nu.

Dat doe je met de melding die je al doorzien hebt. Voor de meldingen die je nog niet doorzien hebt, wil je dat het opvalt zodra er niets verandert.

Zet dit in je vaste instructies: *krijg je na een wijziging exact dezelfde fout terug, ga dan niet nog een keer; schrijf op wat de melding beweert, welke beweringen je hebt gecontroleerd en welke je hebt aangenomen, en controleer dan die aanname.* Twee keer dezelfde fout na twee verschillende pogingen betekent bijna nooit dat de tweede poging te klein was — het betekent dat de melding ergens onwaar is, en die onwaarheid zit in wat je zonder nadenken aannam. Het werkt alleen als opnieuw proberen goedkoop is; is een ronde duur, dan gokt hij.

Wat dit goedkoper maakte: *elke melding die nu klopt, is een fout die de volgende keer geen ronde meer kost — en de volgende keer is er iemand die geen wantrouwen kan opbouwen.*

15. De tweede keer

De tekening verhuisde pas toen het tweede scherm erom vroeg.

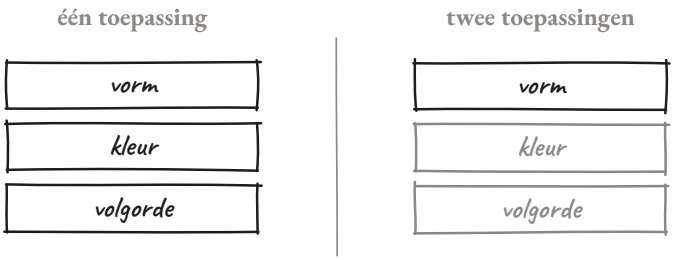
Er was een onderdeel gemaakt — een lijst met regels, met een rand eromheen en een streep onder de kop. Het stond in het scherm waar het voor bedoeld was, het werkte, en het bleef daar staan. Niet omdat ik vergat het netjes op te bergen, maar omdat ik op dat moment nog niet wist wat eraan algemeen was.

Weken later kwam er een tweede scherm dat bijna hetzelfde nodig had. Bijna. En in dat *bijna* stond het antwoord.

Wat de tweede keer je vertelt en de eerste niet

Bij één toepassing weet je niet wat de vaste vorm is en wat toeval. Alles wat er staat, staat er omdat dít geval het vroeg. De breedte, de kleur, de volgorde van de kolommen, het feit dat er precies één regel bovenaan staat — je kunt niet zien welke van die dingen over de rest gaan en welke over dit ene scherm.

Bij twee toepassingen zie je het meteen. **Wat hetzelfde blijft, is het onderdeel. Wat verschilt, is de knop.** Je hoeft niet te raden welke dingen instelbaar moeten worden; de twee gevallen wijzen ze aan.



Bij één toepassing kun je niet zien wat algemeen is. Bij twee wijst het verschil de knip aan.

Daaruit volgt de regel waar dit hoofdstuk om draait:

> **Eerst werkend, de tweede keer herbruikbaar.**

Het is een volgorde, meer niet.

Waarom de andere volgorde duurder is

De verleiding gaat de andere kant op, en ze is sterk. Je bouwt iets, je ziet dat het vaker van pas kan komen, en je maakt het meteen algemeen. Dat voelt vooruitziend.

Wat er dan gebeurt is dit: **een algemeen onderdeel dat uit één geval is afgeleid, heeft de vorm van dat ene geval.** Het is niet algemeen, het is dat geval met een paar parameters eromheen. En de tweede toepassing past er niet in — niet omdat hij zo anders is, maar omdat de knip op de verkeerde plek zit.

Dan sta je voor een keuze die allebei slecht is. Je buigt het tweede geval zodat het in de vorm van het eerste past, en dan doet het scherm iets raars dat niemand kan uitleggen. Of je bouwt het algemene deel om, en

dan moet het eerste geval mee — inclusief alles wat er inmiddels op leunt.

Wachten tot de tweede keer kost je één keer hetzelfde werk doen. De andere volgorde kost je een verbouwing, en je weet vooraf niet hoe groot.

Maar dan moet die tweede keer wel goedkoop zijn

Hier zit de voorwaarde die de regel pas laat werken, en zonder die voorwaarde is *eerst werkend* een excuus.

Als je de eerste keer knip-en-plak gebruikt, is de tweede keer geen tweede keer maar een tweede kopie. En dan gebeurt er iets vervelends: vanaf dat moment is het duurder om ze samen te voegen dan om een derde kopie te maken. De eerste kopie is de duurste beslissing in dit hoofdstuk, en hij voelt op dat moment als de goedkoopste.

Twee dingen houden de deur open, en ze kosten allebei bijna niets:

Zet het op één plek, ook als die plek nog specifiek is. Eén bestand, één functie, één blok. Niet algemeen gemaakt — alleen bij elkaar gehouden. Dan is het later verhuizen een verplaatsing in plaats van een reconstructie.

Schrijf op dat het er is. Eén regel is genoeg: *dit onderdeel staat hier en is nog niet gedeeld.* De tweede toepassing komt vaak weken later, soms van iemand anders, soms van een agent die de eerste nooit gezien heeft. Wat niet vindbaar is, wordt niet hergebruikt maar nagemaakt.



de middelste stap levert het ontwerp op

Eerst werkend op één plek, dan pas gedeeld — maar wel zo dat verhuizen een verplaatsing is.

Negenendertig werkwijzen, en vijftien dragen nog de zin waarmee ze ontstonden

Ik heb het in mijn eigen werkmap geteld, want ik wilde weten of dit klopte of dat ik het mooier maakte dan het was.

Er staan **negenendertig** werkwijzen op schijf: elk een document dat zichzelf aanbiedt zodra het van toepassing is. Bij **vijftien** ervan staat bovenin nog letterlijk de zin waarmee ze ontstonden — meestal iets wat ik in het voorbijgaan zei, mediaan **eenentwintig woorden**, de kortste zes. Wat eruit kwam is mediaan **dertienhonderd woorden** per stuk.

Dat is ruwweg zestig keer zoveel tekst als ik zelf tikte, en dat is niet het punt. **Het punt is dat ik die zin één keer heb gezegd.** Elke volgende keer dat de situatie zich voordeed, was hij er al — ook toen ik er zelf niet meer bij was.

Ze doen niet allemaal hetzelfde, en dat is waarom het een geheel is

Toen ik ze op een rij zette, bleken het zes soorten te zijn. Niet bedacht; zo gegroeid.

soort	aantal	wat het doet
maakt iets	11	bouwt, schrijft, tekent
toetst wat er is	10	meet of het klopt, en hoe je dat meet
houdt iets tegen	2	weigert, zolang een voorwaarde niet vervuld is
kijkt met ogen	5	maakt een beeld en meet daarin
leert bij	7	legt vast wat er is geleerd, zodat het de volgende ronde haalt
raakt de buitenwereld	4	draait iets echt, ergens anders, op een vreemde machine

De verhouding is het interessante. Elf maken iets en twaalf controleren of het klopt — toetsen plus tegenhouders. Dat is bijna één op één, en zo is het niet begonnen: eerst waren er alleen makers. Elke toets is erbij gekomen op de dag dat iets stil misging.

En ze wijzen naar elkaar. Elf paren noemen elkaar over en weer: de maker zegt waar hij zijn werk laat controleren, en de controle zegt wie hem aanlevert. Dat is de overdracht, van beide kanten opgeschreven — en dat is precies wat er ontbreekt als je het niet doet.

De meest genoemde is bovendien niet de slimste maar de meest **gedeelde**: naar het toetsen wijzen er vijftien, naar de plek waar het naar buiten gaat tien. Wie zoekt waar het loont om iets vast te leggen, moet niet kijken wat het knapste is maar wat het vaakst over de schutting gaat.

Zelf doen · Bouw hem één keer en gebruik hem twee keer

> **Wat je doet.** Je hebt in hoofdstuk 6 een controle op certificaten laten maken en hem in hoofdstuk 9 getest: iets met een `geldig tot` naast een datum. In `materieel/` staat precies dezelfde vorm — een keuring

met een `geldig tot` naast een dag van gebruik. **Dit is de tweede keer, en dit is het moment waarop je wéét wat er gedeeld mag worden.** Reken op tien minuten.

TYP DIT

> Kijk naar de controle op persoonscertificaten en naar wat er nodig is om hetzelfde te doen voor materieel/. Zeg eerst wat die twee gemeen hebben en wat per geval verschilt. Maak er daarna één werkwijze van die allebei aankan, en laat me zien welke gegevens je er per keer in moet stoppen.

Wat je ziet. Het gemeenschappelijke is klein: **een ding, een geldigheidsdatum, en een moment waarop het gebruikt wordt.** Het verschil is welk veld waar staat. Eén werkwijze met drie gegevens erin dekt allebei — en meteen ook de verzekeringen, de verklaringen van onderaannemers en de garantietermijnen, want die hebben dezelfde vorm.

WAT ER TERUGKWAM

```
gemeenschappelijk:
  - een ding met een naam
  - een datum waarop het niet meer geldt
  - een moment waarop het gebruikt werd

verschil:
  certificaat -> moment = datum van de opdracht
  keuring     -> moment = dag van gebruik
```

Waar het misgaat. Maak hem niet algemener dan de twee gevallen die je hebt. Eén werkwijze voor twee dingen die je kent is een multiplier; één werkwijze voor zeven dingen die je bedenkt is een raamwerk dat niemand invult. **Wacht tot de tweede keer met samenvoegen.** Had je dit na het eerste geval gedaan, dan had je de verkeerde dingen gedeeld — je wist toen nog niet wat er per geval verschilde. **En laat het moment een gegeven zijn en geen aanname.** Meet je tegen vandaag in plaats van te-

gen de dag van gebruik, dan geeft dezelfde controle volgend jaar een ander antwoord.

EN DAN ZO

> Draai die ene werkwijze nu op alle vier: certificaten, keuringen, verzekeringen en verklaringen van onderaannemers.

EN DAN DIT

```
geldigheid-bewaken, toegepast op vier soorten:
  persoonscertificaten   4 opdrachten mis
  materieelkeuringen     2 verlopen
  verzekeringen          0
  onderaannemers         1 VCA verlopen
vier administraties, één werkwijze, en er is
nooit iets voor de vierde bijgebouwd
```

Dat is ook waarom dit hoofdstuk hier staat en niet verderop. De tweede keer is niet het moment waarop je tijd bespaart — het is het enige moment waarop je *weet* wat er gedeeld mag worden.

Dit is stap vijf en zes uit de lus

Wie hoofdstuk 3 nog voor zich heeft, herkent deze regel, want hij staat daar al — alleen als onderdeel van iets groters.

De lus die op het callcenter ontstond had zes stappen, en de laatste twee waren: **behouden wat systematisch werkt**, en dan pas **formaliseren**. Precies in die volgorde, en precies om de reden die hierboven staat.

Ik heb daar geen stappenplan geschreven na het eerste gesprek dat goed ging. Dat zou het opschrijven van toeval zijn geweest. Ik schreef het op toen ik dezelfde beweging voor de zoveelste keer zag en kon aanwijzen wat er elke keer hetzelfde was — en daarmee ook wat er elke keer anders

mocht zijn. **Dat verschil is het model.** Het is niet af te leiden uit één geval, hoe goed dat geval ook was.

En daar zit de reden dat "behouden wat werkt" vóór "formaliseren" staat. Wie het omdraait, legt een vorm vast en gaat daarna kijken of hij klopt. Dan verdedig je je model in plaats van het te toetsen, en alles wat er niet in past wordt een uitzondering in plaats van een aanwijzing.

Dezelfde volgorde, in drie vakken, met dezelfde reden. Dat is precies wat dit boek een patroon noemt.

En dit gaat niet over software

De vorm hiervan zit overal waar iemand iets voor de tweede keer doet.

- **Een recept dat je één keer maakte** is nog geen recept, het is wat je die avond deed. Pas als je het een tweede keer maakt met andere gasten, zie je welke stappen er echt in horen en welke van die ene keer waren.
- **Een formulier dat voor één afdeling is gemaakt** krijgt de woorden van die afdeling. De tweede afdeling vult het in met een ongemakkelijk gevoel, en niemand kan precies zeggen waarom.
- **Een training die je één keer gaf** is een verhaal. Bij de tweede groep merk je welke voorbeelden het deden en welke alleen voor de eerste groep klopten. Dan pas heb je een training.
- **Een contract dat uit één onderhandeling komt** draagt de zorgen van die ene tegenpartij. Gebruik hem voor de volgende en er staan clausules in die nergens over gaan.

In alle vier de gevallen is de fout niet dat er iets hergebruikt wordt. De fout is dat het *model* gemaakt is op het moment dat er nog maar één geval was.

En let op wat dit zegt over tempo. De regel klinkt als uitstel, en dat is hij niet. Hij zegt alleen: doe het generaliseren op het moment dat het informatie oplevert in plaats van dat het een gok is. Dat is later, en het is goedkoper.

Wanneer je de regel wél mag overslaan

Er is één geval, en het is het belangrijke.

Als je nu al zeker weet dat het duizend keer gebeurt, is de eerste keer niet je eerste geval — het is je enige kans om de vorm goed te krijgen. Wat duizend keer draait, kun je later niet stilletjes omzetten; er hangt dan te veel aan.

Het onderscheid is dus niet *hoe vaak denk ik dat dit voorkomt*, maar *hoe duur wordt het om het later te veranderen*. Bij een schermonderdeel: goedkoop, dus wacht. Bij de vorm waarin je alles opslaat, bij de manier waarop je agenten met elkaar praten, bij de plek waar geld langsgaat: duur, dus denk het nu door. Daar geldt de duizend-keer-vraag en niet deze regel.

Die twee spreken elkaar niet tegen; ze gaan over een andere grootheid. De duizend-keer-vraag gaat over de kosten van herhalen. Deze gaat over de kosten van te vroeg vastleggen.

Probeer: bouw je iets dat lijkt op wat er al is, vraag dan vóór je begint of er al zoiets bestaat en hoeveel het verschilt van wat je nu nodig hebt. Is er iets, zet de twee dan naast elkaar en laat benoemen wat hetzelfde is en

wat verschilt. Het verschil tussen de twee gevallen *is* het ontwerp — je hoeft het niet te bedenken, alleen te zien. Doe het vóór je bouwt: achteraf heb je al een oplossing en ga je die verdedigen in plaats van vergelijken.

Dat werkt alleen als die eerste keer nog te vinden is. Daar zorg je op het moment zelf voor, en het kost vijf minuten.

Tip: zet bij het afronden van iets nieuws het onderdeel in één blok bij elkaar — zonder het algemeen te maken — met één regel erboven die zegt wat het is en dat het nog niet gedeeld wordt. Verhuizen is goedkoop, reconstrueren niet, en wat niet vindbaar is wordt niet hergebruikt maar nagemaakt. *Bij elkaar zetten* is iets anders dan *algemeen maken*: het eerste kost vijf minuten en houdt de deur open, het tweede is precies wat dit hoofdstuk afraadt zolang er één geval is.

> **Wat dit goedkoper maakte:** *elk gedeeld onderdeel is er een waarvan de vorm uit twee gevallen kwam in plaats van uit een gok, en dat is de reden dat het derde geval er wél in paste.*

16. Zoek de sleutel, niet de deuren

Ik had een lijst gemaakt van dingen die gebouwd moesten worden, en de lijst was lang.

De vraag was hoe een agent zelf koppelingen zou kunnen maken met de buitenwereld: mail, servers, betalingen, bestellingen, het inschrijven van een bedrijf. Ik deed wat je doet met zo'n vraag. Ik inventariseerde. Per dienst: wat er nodig is, hoe je verbinding maakt, wat er kan misgaan. Het werden er tientallen, en elke volgende regel maakte het geheel een beetje ongeloofwaardiger.

Toen viel het kwartje, en het paste in twee zinnen.

> **De basis is een e-mailadres, en kunnen lezen en verzenden vanuit die mailbox. Dat opent de weg naar een account bij vrijwel elke andere dienst.**

Ik had dertig deuren geteld. Dit was de sleutel.

Waarom je vanzelf de deuren gaat tellen

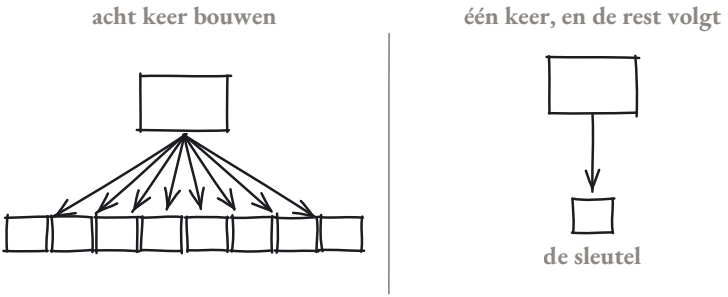
Dit is geen domme fout en daarom is hij zo hardnekkig.

Een vraag komt bijna altijd binnen als een opsomming. *We moeten kunnen mailen, bestellen, betalen.* Die vorm nodigt uit tot een antwoord in

dezelfde vorm: een plan per onderdeel. Dat voelt grondig, het is te plannen, en je kunt er een schatting bij zetten.

En het is te verdedigen. Elke deur op de lijst is echt nodig. Niemand kan je uitleggen dat je iets verkeerds doet — hooguit dat het lang duurt.

Maar een lijst deuren is een lijst, geen ontwerp. Hij vertelt je wat er ontbreekt en niet wat het eerst moet. En bij lange lijsten wordt de volgorde de belangrijkste beslissing die je neemt, juist omdat je hem meestal niet neemt: je begint bovenaan.



Dertig deuren, of één sleutel die de meeste opent.

Wat een sleutel is

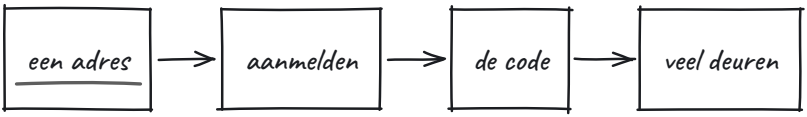
Een sleutel is geen slimmigheid. Het is een heel gewoon ding met één bijzondere eigenschap: **hij is de voorwaarde onder meerdere deuren tegelijk.**

In dit geval: een mailadres. Er is niets bijzonders aan een mailbox, en precies daarom zie je hem over het hoofd. Maar kijk wat hij mogelijk maakt.

Je kunt je ergens aanmelden, want aanmelden vraagt om een adres. Je kunt de bevestigingscode ontvangen, want die komt per mail. Je kunt een wachtwoord terugzetten, want dat gaat ook per mail. Je kunt een account beheren, want beheerberichten komen daar binnen.

En dan pas begint het. Met een account bij een dienst die koppelingen aanbiedt, kun je bij de koppelingen van díe dienst. Eén aanmelding geeft toegang tot duizenden verbindingen die iemand anders al gebouwd heeft.

De keten is niet lang, en dat is het punt:



elke stap maakt de volgende mogelijk

Eén adres, en de rest volgt — elke stap maakt de volgende mogelijk.

Dertig deuren bouwen kost dertig keer werk. De sleutel maken kost één keer werk, en daarna is het werk van iemand anders.

Hoe je een sleutel herkent

Drie vragen, en ze zijn alle drie te beantwoorden voordat je begint te bouwen.

Welke voorwaarde komt op meer dan één plek terug? Schrijf per item op je lijst op wat er waar moet zijn voordat het kan. Meestal staat er dan drie of vier keer hetzelfde, en dat is je kandidaat. Niet wat het meest gevraagd wordt — wat het vaakst *onder* iets ligt.

Is die voorwaarde zelf te maken zonder een van de deuren? Een sleutel die zelf achter een van je deuren zit, is geen sleutel maar een cirkel. Een mailadres is aan te maken zonder dat je eerst een account ergens anders hebt; dat is precies waarom hij werkt.

Wat wordt er daarna bereikbaar dat het nu niet is? Dit is de vraag uit het hoofdstuk over bereikbaarheid, en hier komt hij terug. Als het antwoord "één ding" is, heb je een deur gevonden en geen sleutel.

Waar ik deze zet geleerd heb

De twee zinnen hierboven draaiden mijn lijst om, maar de zet zelf was niet nieuw voor me. Ik had hem jaren eerder nodig gehad, in werk dat niets met koppelingen te maken had.

Iemand die naar werk begeleid wordt, komt binnen met een lijst. Een baan, een inkomen, een ritme, iets om mee te beginnen, misschien een opleiding. **Dat is een lijst deuren**, en de verleiding is om hem af te werken: solliciteren, inschrijven, een cursus zoeken. Zo zijn die lijsten ook bedoeld.

Wat ik leerde is dat het er bijna nooit zo aan toeging. Er lag meestal **één ding onder** dat alle andere goedkoop maakte zodra het er was — en dat ding stond nooit op de lijst, om precies dezelfde reden als hierboven: het is geen doel. Niemand vraagt erom. Het ligt eronder.

En het was zelden een vacature. **Voorspelbaarheid was vaker de sleutel dan aanbod.** Zodra de week er hetzelfde uitzag, zodra duidelijk was wat er ging gebeuren en in welke volgorde, werd de rest van de lijst opeens haalbaar. Niet omdat iemand meer kon, maar omdat er energie overbleef die eerst opging aan het niet weten.

Er zit één les bij die verder gaat dan de rest van dit hoofdstuk. Soms ligt de sleutel niet bij degene die de lijst heeft. Als een werkgever de deur dichthoudt, is er niets aan die persoon te verbeteren dat hem opent — en dan is werken aan de lijst werken aan de verkeerde kant. Dat is hetzelfde punt als in het hoofdstuk over toegang, en het is de onaangenaamste variant van deze zet: **de sleutel zit soms in iemand anders zijn zak.**

Dan is de vraag niet *wat moet er nog gebeuren* maar *wie moet er ja zeggen, en wat heeft die nodig.* Ook dat is één ding in plaats van een lijst. Het is alleen een moeilijker één.

En dit gaat niet over koppelingen

De vorm zit overal waar iemand een lijst maakt van dingen die moeten gebeuren.

- **Een organisatie die twaalf soorten rapportages wil**, heeft geen twaalf rapporten nodig maar één plek waar de gegevens correct binnenkomen. Twaalf rapporten bouwen op ongelijke bronnen is twaalf keer hetzelfde gesprek voeren.
- **Een school die vijf soorten begeleiding aanbiedt** loopt vast op hetzelfde: of er iemand is die de leerling kent. Dat is de sleutel, en het staat op geen enkele lijst omdat het geen aanbod is.
- **Een bedrijf dat in vier landen wil verkopen** denkt aan vier vertalingen en vier betaalmethoden, en de voorwaarde onder alle vier is één ding: kunnen factureren op een manier die daar geldig is.

- **Iemand die zijn leven op orde wil krijgen** en een lijst van tien gewoonten maakt, ontdekt meestal dat er één onder ligt — slaap, meestal — waar de andere negen van afhangen.

In alle vier de gevallen is het opvallende dat de sleutel niet op de lijst stond. Hij stond er niet op omdat hij geen doel is. Niemand vraagt om een mailadres, om goede brongegevens, of om slaap. Ze staan onder de dingen waar wel om gevraagd wordt.

Zelf doen · Zoek het begrip in plaats van de plekken

Wat je doet. Je hebt nu vier controles die allemaal hetzelfde doen. In plaats van er een vijfde bij te bouwen: zoek het **begrip** dat eronder zit en laat het overal tegelijk toepassen. Reken op tien minuten.

TYP DIT

> Zoek in deze hele map elk veld dat een datum bevat waarna iets niet meer geldig is — hoe het ook heet — en zet ze op één lijst, gesorteerd op datum, met erbij of ze al verlopen zijn. Neem de peildatum 31 mei 2026.

Wat je ziet. Zesendertig vervaldatums in zes mappen, en drie van die mappen had je zelf niet genoemd: offertes hebben een geldigheidsdatum, garanties lopen af, en verzekeringen ook. **Negen ervan zijn al verlopen**, waaronder een offerte die nog op *open* staat — die kan dus niet meer aangenomen worden en niemand die dat wist.

WAT ER TERUGKWAM

```
gevonden in: personeel 12, materieel 8, offertes 5,
onderaannemers 4, verzekeringen 4, nazorg 3
2026-01-31 VERLOPEN  VCA* onderaannemer
2026-02-28 VERLOPEN  Veilig werken op hoogte
2026-03-19 VERLOPEN  bus-2 APK
2026-04-05 VERLOPEN  rolsteiger-B keuring
2026-05-02 VERLOPEN  NLOF-2026-03 (status: open)
```

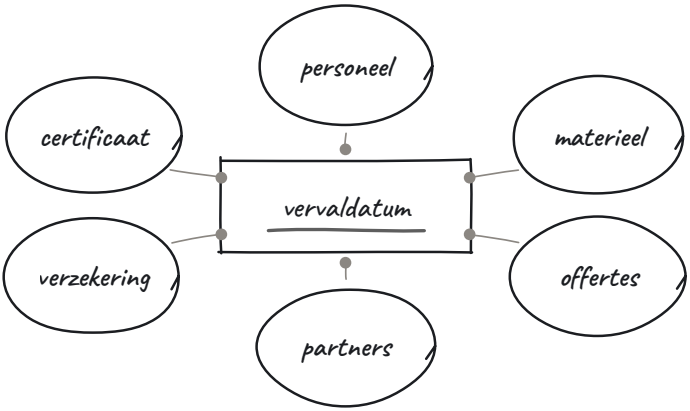
Waar het misgaat. Zeg "hoe het ook heet". Dit veld heet in zes mappen zes dingen: *geldig tot*, *APK geldig tot*, *keuring geldig tot*, *loopt tot*. Vraag je naar één naam, dan vind je één map. **En kijk daarna wat er NIET in de lijst staat.** Hier miste de scan de bedrijfscertificaten, omdat die in een tabel staan met de datum in een kolom in plaats van achter een woord. Daar zat de VCA* bij, die over dertig dagen verloopt — het certificaat dat bij vrijwel elke aanbesteding wordt gevraagd. **Een sleutel die bijna alles opent, opent bijna altijd net dat ene niet.**

EN DAN ZO

Zoek ook naar datums die in een tabelkolom staan in plaats van achter een woord, en voeg ze toe aan dezelfde lijst.

EN DAN DIT

```
drie bedrijfscertificaten alsnog gevonden:
  2026-06-30  over 30 dagen  VCA**
  2027-11-30  over 548 dagen ISO 9001
  2028-02-28  over 638 dagen garantieregeling
totaal 39 in plaats van 36, en de eerstvolgende
die verloopt is meteen de belangrijkste
```



Eén begrip opent zes administraties — en net dat ene slot niet.

Wanneer je toch de deuren moet doen

Dit is geen truc die altijd werkt, en er zijn twee gevallen waarin hij misleidt.

Als er geen gedeelde voorwaarde is. Soms zijn vijf dingen gewoon vijf dingen. Dan is zoeken naar de sleutel uitstel, en de eerlijke conclusie is dat het vijf keer werk is. Je merkt dat aan de eerste vraag hierboven: als er niets drie keer terugkomt, is er niets.

Als één deur nu geld of tijd oplevert en de sleutel pas later. De sleutel is bijna altijd de betere zet, maar niet als je er niet aan toekomt. Dat is dezelfde afweging als bij bereikbaarheid: de beste zet die je niet kunt doen, is geen zet.

Wat je in beide gevallen wél doet, is het opschrijven. Een gedeelde voorwaarde die je vandaag overslaat, is over drie maanden nog steeds de sleutel — en dan ben je hem vergeten.

Probeer: heb je een lijst van dingen die gebouwd moeten worden, vraag dan per item wat er waar moet zijn vóórdat het kan — alleen voorwaarden, geen oplossingen — en groepeer daarna de voorwaarden die bij meerdere items terugkomen. Neem de voorwaarde die het vaakst terugkomt en vraag of die zelf te maken is zonder een van de items, en wat er daarna bereikbaar wordt. De sleutel staat nooit op de lijst, want hij is geen doel. Vraag om vóórwaarden: op een vraag naar oplossingen krijg je een plan per item terug, en dan heb je je lijst netjes overgeschreven.

Soms kom je er niet aan toe, en dat is een geldige uitkomst. Alleen moet hij dan wel ergens staan, want anders vind je hem terug door hem opnieuw te ontdekken.

Maak je hem nu niet, schrijf hem dan op als los punt, met daarbij welke items erdoor mogelijk worden en wat het kost om hem te maken — geen planning, alleen dat hij bestaat. Een sleutel die je vandaag overslaat is over drie maanden nog steeds de sleutel; staat hij nergens, dan kom je hem tegen door hem opnieuw te ontdekken, en dan heb je inmiddels drie deuren met de hand gebouwd. Zet erbij wat hij mogelijk maakt en niet alleen wat hij is: een regel die alleen *"mailadres nodig"* zegt, leest over drie maanden als een klusje.

> **Wat dit goedkoper maakte:** *één keer een voorwaarde invullen in plaats van dertig keer een verbinding bouwen — en alles wat daarna langskwam, hoefde niet opnieuw langs die vraag.*

17. Nul afhankelijkheden

Er ging geen enkele middag verloren aan iets dat van iemand anders was.

Dat klinkt als een detail en het is de reden dat er zoveel in past. Wie vaker met software werkt kent het patroon: je begint aan iets, en voor je begonnen bent is er een versie die niet meer klopt, een onderdeel dat vandaag anders heet, of een pakket dat verdwenen is. Geen ramp, elke keer een halve dag, en het is nooit het werk waarvoor je ging zitten.

Die middagen zijn er niet geweest. Niet doordat het meezat, maar doordat er niets was dat ze kon veroorzaken.

Waarom nul iets anders is dan weinig

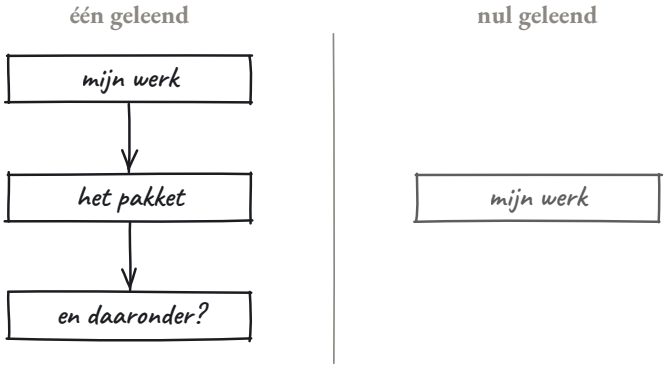
Het is een stap en geen schaal, en daar draait het hoofdstuk om.

Bij één geleend onderdeel heb je een keten. Je moet weten wie het maakt, of het onderhouden wordt, wat er gebeurt als het niet meer wordt onderhouden, en wat er ónder dat onderdeel weer zit — want dat heeft zelf ook geleend. Bij tien onderdelen heb je diezelfde vragen tien keer, en je stelt ze niet meer.

Bij nul is er geen keten. Niet een korte keten — geen. De vraag "wat gebeurt er als dit verandert" verdwijnt, en daarmee alles wat eruit volgt:

het bijhouden, het controleren, het uitzoeken op de dag dat er iets niet meer werkt.

Van tien naar één is een verbetering. Van één naar nul is een andere toestand.



Van tien naar één is minder. Van één naar nul is anders: de keten verdwijnt.

Wat je eigenlijk leent

Het misverstand zit in wat je denkt binnen te halen. Je denkt dat je code binnenhaalt. Wat je binnenhaalt is een belofte.

De belofte dat het blijft bestaan. Dat degene die het maakte er over drie jaar nog is, of dat iemand anders het overneemt.

De belofte dat het blijft werken zoals het werkte. Niet dat het foutloos is — dat het niet stilletjes van gedrag verandert.

De belofte dat er niets raars gebeurt. Dat er niemand iets in stopt dat er niet hoort, en dat niemand het overneemt met een ander doel.

Die drie beloftes zijn meestal waar, en dat is precies het probleem: ze zijn zo vaak waar dat je stopt met ze te tellen. Wat er dan gebeurt, is niet dat er één belofte breekt. Het is dat je er inmiddels honderd hebt en niet meer weet welke.

Nul is een eis, geen score

Hier zit de zin waar het hoofdstuk om draait.

> **Nul is een eis, geen score.**

Het verschil is het hele verhaal. Een score is iets waar je goed op probeert te staan: drie is beter dan tien, dus we doen ons best. Een eis is een grens die de vraag verandert. Bij een eis van nul is "is dit pakket goed genoeg" geen vraag meer, en komt er een andere voor in de plaats: **heb ik dit eigenlijk wel nodig?**

Dat is bijna altijd de betere vraag. Een verrassend deel van wat je zou lenen, blijkt iets te zijn dat je in dertig regels zelf schrijft, precies past, en daarna nooit meer verandert. Dat ontdek je alleen als de makkelijke weg dicht zit.

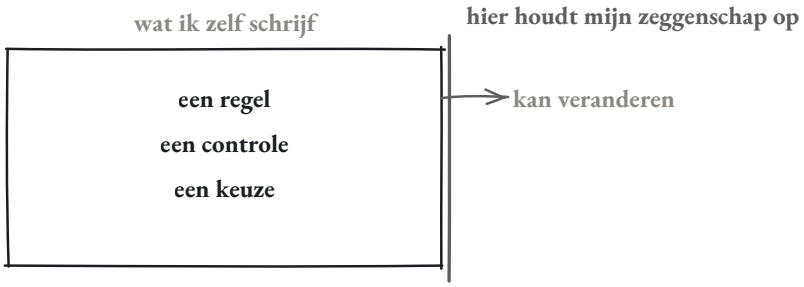
En een eis werkt waar een oordeel faalt. Bij elke afzonderlijke afweging win je het van jezelf: dit ene pakket is groot, goed onderhouden, iedereen gebruikt het. Dat klopt ook. Maar die afweging maak je honderd keer, en je verliest hem niet één keer groot maar honderd keer klein.

En dit is precies waarom het nu meer telt

Zolang jij de enige bent die code toevoegt, is een oordeel per geval nog te doen. Zodra er agenten meewerken, niet meer.

Een agent kan de drie beloftes niet wegen. Hij ziet dat een pakket bestaat, dat het populair is, en dat het zijn probleem oplost. Wat hij niet ziet is wat er over twee jaar mee gebeurt, wie erachter zit, of hoe diep de keten eronder gaat. Hij voegt het toe, met goede redenen, en hij meldt het niet als een beslissing.

En hij doet dat sneller dan jij het kunt nalopen. Een regel die op oordeel steunt, werkt niet op die snelheid. Een grens wel: **niets van buiten** is te controleren zonder erover te discussiëren, en dat is precies wat een regel moet kunnen.



Wat je zelf schrijft verandert alleen als jij het verandert. Al het andere kan verschuiven zonder te vragen.

De grens is niet "zelf getypt"

Dit hoofdstuk zou onzin zijn zonder deze paragraaf, want nul is nooit letterlijk nul.

Er draait een besturingssysteem. Er is een processor. Er is een taal. Daar leun je op, en dat is geen inconsequentie — het is de grens op de juiste plek trekken. De vraag is niet *heb ik het zelf geschreven* maar: **hoe waar-**

schijnlijk is het dat dit onder mij verandert, en wat gebeurt er met mij als het gebeurt?

De manier waarop een programma iets aan het besturingssysteem vraagt, is al decennia hetzelfde en verandert niet omdat iemand een nieuwe versie uitbrengt. Een pakket dat vorig jaar populair werd, is dat niet.

Zo bekeken is dit hoofdstuk de tegenhanger van hoofdstuk 23, over vastzitten. Daar ging het om aanbieders van vermogen: wie levert het denkwerk, en wat als die wegvalt. Hier gaat het om wat je onder je eigen werk legt. Dezelfde vraag, twee lagen: **wat kan er onder mij verschuiven zonder dat iemand het mij vraagt.**

Wat het kost

Je schrijft meer zelf. Dat is de prijs en hij is echt.

Sommige dingen krijg je gewoon niet. Wat vijftig mensen in vijf jaar bouwden, ga jij niet in een week nabouwen, en als je dat wel probeert krijg je iets slechters. De eerlijke vorm van deze regel is dus niet "nooit iets lenen" maar **"nul, en waar dat niet kan, bewust en zichtbaar"** — met de reden erbij, zodat het een besluit is en geen gewoonte.

En er is een variant van deze fout die net zo duur is: alles zelf bouwen omdat je het leuk vindt. Dat is dezelfde uitkomst met een ander verhaal eromheen, en het kost je wél de tijd.

De toets is de middag. Hoeveel middagen ben je het afgelopen jaar kwijtgeraakt aan iets van een ander dat veranderde? Bij nul heb je gelijk. Bij vijf is dat een week, en dat is geen ongemak meer maar een post.

Zelf doen · Kijk wat je van anderen leent

Wat je doet. Vier onderaannemers doen werk dat jij niet zelf doet. Dat is geen inkoop maar een lening: bij ketenaansprakelijkheid komt hun probleem uiteindelijk bij jou terecht. Laat nakijken wat je van ze weet en hoe oud dat is. Reken op vijf minuten.

TYP DIT

> Beoordeel elke onderaannemer in onderaannemers/ op drie dingen: is de VCA nog geldig, hoe oud is de verklaring betalingsgedrag, en is er een G-rekening. Zet erbij aan welke lopende opdrachten ze werken.

Wat je ziet. Drie van de vier zijn in orde. **De vierde werkt op een lopende opdracht, heeft een VCA die vier maanden geleden verliep, een verklaring betalingsgedrag van ruim anderhalf jaar oud, en geen G-rekening.** Elk van die drie is op zich een administratief puntje. Samen is het de enige partij waar een probleem van hen een probleem van jou wordt.

WAT ER TERUGKWAM

```
veenstra    VCA* geldig    verklaring 110 dagen  G-rek ja
sluiter     VCA* geldig    verklaring 123 dagen  G-rek ja
rooden      VCA** geldig   verklaring  88 dagen  G-rek ja
bergsma     VCA* VERLOPEN  verklaring 555 dagen  G-rek NEE
            werkt aan NLOP-2026-006
```

Waar het misgaat. Vraag om de drie dingen samen en niet los. Elk apart is een opmerking waar je overheen leest; samen is het een besluit. **Vraag erbij aan welk werk ze bezig zijn.** Een partij die niet in orde is en niets doet, is een aantekening; een partij die niet in orde is en op een lopende opdracht staat, is vandaag. **En let op dat "geen G-rekening" er onschuldiger uitziet dan het is.** De andere twee punten hebben een datum en vallen daarom op; dit is een leeg veld, en lege velden zie je niet.

EN DAN ZO

> Zeg er per onderaannemer bij wat het ons kost als het misgaat, in plaats van alleen wat er ontbreekt.

EN DAN DIT

```
bergsma, op NLOP-2026-006 (9.870 euro):
  VCA verlopen    -> aansprakelijk bij een
                     ongeval op onze bouwplaats
  geen G-rekening -> bij niet afgedragen loon-
                     heffing komt de rekening
                     bij ons
  verklaring oud  -> zegt niets meer over nu
```

En dit gaat niet over software

- **Een keuken die zijn eigen fond trekt** heeft meer werk en heeft nooit het probleem dat de smaak verandert omdat de leverancier zijn recept aanpaste. Zijn diepvriezer is stabieler dan iemands productielijn.
- **Een monteur met eigen gereedschap** wacht nooit op een uitgifteloket. Dat lijkt een comfortkwestie tot de dag dat het loket dicht is.
- **Een praktijk die haar eigen dossiers bijhoudt** in plaats van alleen in andermans systeem, kan verhuizen. Wie dat niet doet, kan het ook niet, en ontdekt dat op het verkeerde moment.
- **Een boer die zijn eigen zaad wint** levert wat in op opbrengst en koopt er onafhankelijkheid voor terug. Dat is precies dezelfde ruil als hier.

In alle vier gaat het niet om wantrouwen en niet om zelf alles willen doen. Het gaat om de vraag welke beloftes van anderen je in je eigen werk hebt ingebouwd zonder ze op te schrijven.

Probeer: laat alles opsommen waar je project van buiten op leunt, ook wat indirect binnenkomt, met per stuk wie het maakt, wanneer het voor het laatst is bijgewerkt, en wat er stukgaat als het morgen verdwijnt. Bijna niemand kent dat getal, en het is meestal een veelvoud van wat je zelf zou noemen — want het meeste komt indirect binnen, via iets dat je wél koos. Daar zit ook de verrassing; wat je zelf koos ken je wel.

Dat getal is de aanleiding en niet de oplossing. De oplossing is een grens die niet elke keer opnieuw afgewogen hoeft te worden.

Zet dit in je vaste instructies: *voeg nooit een afhankelijkheid van buiten toe; kom je er een tegen die je nodig denkt te hebben, bouw hem dan niet in maar meld hem — wat hij doet, hoeveel regels het kost om het zelf te doen, en wat we missen als we het niet doen.* Een agent kan niet wegen wat een afhankelijkheid op lange termijn kost en meldt het toevoegen niet als beslissing. Vraag om dat aantal regels: het beslist de discussie vaker dan je denkt, want heel veel van wat geleend wordt is dertig regels.

> **Wat dit goedkoper maakte:** *er ging geen middag verloren aan andermans verandering, en elke dag die niet aan onderhoud opging, kwam beschikbaar voor iets dat wel vooruitkwam.*

18. Het vliegwiel

Op een ochtend was er iets beter geworden zonder dat ik het beter had gemaakt.

Een foutmelding die de week ervoor nog de verkeerde kant op wees, wees nu de goede kant op. Ik had hem niet aangepast. Wat er gebeurd was: die melding was ergens in een logboek beland omdat hij iemand had misleid, het logboek was gesorteerd, de bovenste regels waren in een instructie terechtgekomen, en die instructie reisde sindsdien mee. De verbetering kwam uit het werk zelf.

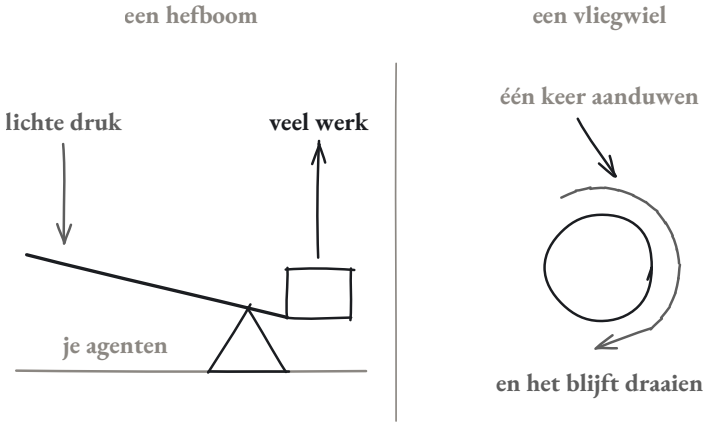
Het is een kleine gebeurtenis en ik weet nog precies wat het deed: het voelde alsof er iets losliet.

Een hefboom en een vliegwiel zijn niet hetzelfde

Dit boek gaat tot hier over hefbomen. Kleine inzet, grote uitslag — je duwt op één plek en er komt meer uit dan je erin stopt. Een multiplier is daar de nette naam voor: werk dat het volgende werk goedkoper maakt.

Maar een hefboom werkt alleen zolang jij duwt. Haal je je hand weg, dan gebeurt er niets meer. Alle winst zit in de verhouding tussen jouw kracht en het resultaat, en jouw kracht is nog steeds de bron.

Een vliegwiel is iets anders. Hij slaat op. Je duwt hem op gang — traag, met veel moeite, en de eerste rondes lijkt het zonde van de energie — en op een gegeven moment draait hij door zonder jou. Niet omdat hij energie maakt, maar omdat hij genoeg heeft opgeslagen om zijn eigen wrijving te overwinnen.



Een hefboom vergroot zolang je drukt. Een vliegwiel duw je één keer aan en dan blijft hij draaien.

Dat is het scharnier van dit boek. Deel I en II gaan over het bouwen van hefbomen. Vanaf hier gaat het over de vraag wanneer het ding uit zichzelf gaat draaien.

Wat een vliegwiel zijn massa geeft

Een vliegwiel zonder massa is een wiel: hij draait zolang je duwt en staat daarna stil. De massa is wat je erin hebt opgeslagen, en in dit werk is dat één ding.

Alles wat is opgeschreven. De fouten die genoteerd zijn en regels zijn geworden. De tests die al bestonden toen deze wijziging kwam. De beslissingen die met hun reden zijn vastgelegd, zodat niemand ze opnieuw hoeft te nemen. De voorbeelden die meereizen. De controles die vanzelf meedraaien.

Elk daarvan kostte op de dag zelf tijd en leverde niets op — dat is de definitie van een multiplier uit deel I. Het verschil is dat ze nu niet alleen het volgende werk goedkoper maken, maar **samen genoeg zijn om een ronde te voltooien zonder dat iemand hem duwt.**

Dat is ook de reden dat het zo lang duurt voordat je iets merkt. Massa opbouwen ziet er tot het laatste moment uit als niets opleveren.

De drie voorwaarden, en de derde wordt vergeten

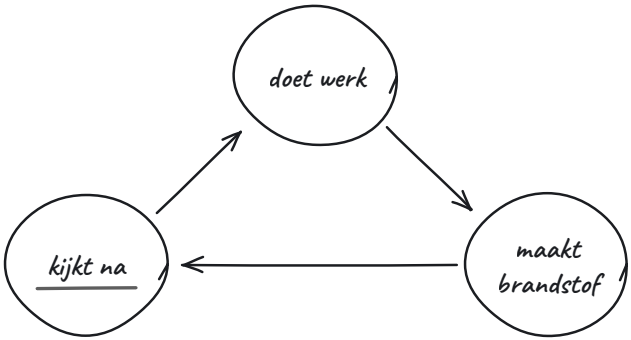
Een lus draait pas uit zichzelf als er drie dingen kloppen. De eerste twee zijn voor de hand liggend; de derde is degene waarop het meestal stukloopt.

Eén: hij maakt zijn eigen brandstof. Wat de lus produceert, moet de volgende ronde voeden. Een foutenlogboek dat niemand leest, voedt niets. Een logboek waaruit regels komen die meereizen, voedt de volgende ronde wel.

Twee: de ronde sluit zonder jou. Als er ergens in de lus een stap zit waar jij op moet klikken, is het geen vliegwiel maar een hefboom met een pauze erin. Dat mag — maar noem het dan zo, want je zult het verschil merken op de dag dat je er niet bent.

Drie: de controle is goedkoper dan de ronde. Hier gaat het mis. Een lus die zichzelf voedt en zonder jou sluit, maar waarvan je achteraf niet

goedkoop kunt zien of hij het goed deed, draait niet door — hij loopt weg. En je merkt het pas als er iets zichtbaars kapot is, wat betekent dat er daarvoor al een tijd onzichtbaar iets kapot was.



Zichzelf voeden, zonder jou sluiten, goedkoop te controleren — alle drie, of het is geen vliegwiel.

Waarom je er geen hand in steekt

Als het draait, is de verleiding om het beter te maken het grootst. Je ziet iets dat eleganter kan. Je hebt tijd. Het is jouw ding.

Je steekt geen hand in een draaiend vliegwiel.

Dat klinkt behoudend en dat is het niet. De redenering is een rekensom. Elke wijziging aan iets dat draait kost een controle die je anders niet had hoeven doen, en de verbetering die je voor ogen had is meestal klein. Daar tegenover staat een kans — niet groot, maar niet nul — dat je iets stukmaakt dat sindsdien door niemand meer wordt nagekeken, juist omdát het draaide.

En let op wat er verandert nu er agenten meewerken. **Een agent heeft geen aarzeling.** Bij een mens zit er iets tussen het zien van een verbetering en het doorvoeren ervan: een vaag gevoel dat dit stuk al een tijd goed gaat. Dat gevoel is opgebouwd uit ervaringen die niet opgeschreven zijn, en het reist dus niet mee. Een agent die iets "even netter" maakt, weet niet wat er al goed aan was, en hij vraagt het ook niet.

Daarom is *niet in roeren* niet iets wat je hoopt, maar iets wat je opschrijft. In de instructies, in het ticket, in de code. Wat draait en waar niemand meer naar kijkt, hoort te zeggen dat het draait.

En dit gaat niet over software

Vliegwielen zijn ouder dan dit onderwerp en je herkent ze aan hetzelfde gevoel: het duurt lang voor er iets gebeurt, en dan gebeurt het vanzelf.

- **Een zaak waar klanten andere klanten meebrengen.** De eerste twintig kostten alle twintig moeite. Ergens daarna beginnen ze elkaar te halen, en de moeite per klant zakt zonder dat je iets veranderd hebt.
- **Een team dat nieuwe mensen zelf inwerkt.** Zolang jij het doet, is het een hefboom. Zodra de vorige nieuwe de volgende inwerkt, draait het — en de kwaliteit hangt dan aan wat er is opgeschreven, niet aan wie er toevallig tijd had.
- **Een dossier dat zichzelf bijhoudt** omdat het bijwerken ervan onderdeel is van het werk in plaats van iets erna. Wat erna komt, gebeurt niet.
- **Een gewoonte die geen wilskracht meer kost.** Dezelfde vorm, in het klein: de eerste weken duw je, en op enig moment is niet-doen meer moeite dan doen.

In alle vier de gevallen is de vraag dezelfde. Niet *hoe krijg ik dit sneller* maar **wat moet er opgeslagen zijn voordat het zonder mij verder kan.**

Zelf doen · Zet een wekker en kijk pas volgende week

Wat je doet. Je hebt nu vijf werkwijzen die elk iets controleren. Tot nu toe start jij ze. **In dit hoofdstuk stop je daarmee.** Je maakt er één die de andere aanroept, en je zet hem op een vaste dag — en daarna kijk je niet meer of hij gedraaid is, maar wat hij heeft gevonden. Reken op vijftien minuten, en daarna een week niets.

TYP DIT

> Maak een werkwijze die de bestaande controles achter elkaar aanroept en alleen meldt wat er nieuw of verlopen is, met bovenaan wat er binnen dertig dagen afloopt. Zet hem daarna klaar om elke maandagochtend te draaien, en laat me zien wat er de eerstvolgende keer uit zou komen.

Wat je ziet. Een werkwijze die zelf niets doet en alleen aanroept — **de eerste die je maakt waarin geen enkele controle staat.** Dat voelt leeg en het is het punt: wat hij toevoegt is de volgorde, de samenvatting, en het feit dat hij gaat zonder jou. **Bovenaan staat wat er binnen dertig dagen verloopt**, want dat is het enige in deze ronde waar de tijd tegen je werkt.

WAT ER TERUGKWAM

```
MAANDAGRONDE
verloopt binnen 30 dagen:
  VCA** bedrijfscertificaat  over 30 dagen
nieuw sinds vorige week : 0
opgelost                : 0
ongewijzigd             : 15  (al 1 ronde)
gedraaid: regels-nalopen, geldigheid-bewaken,
          regels-nalopen-controle
```

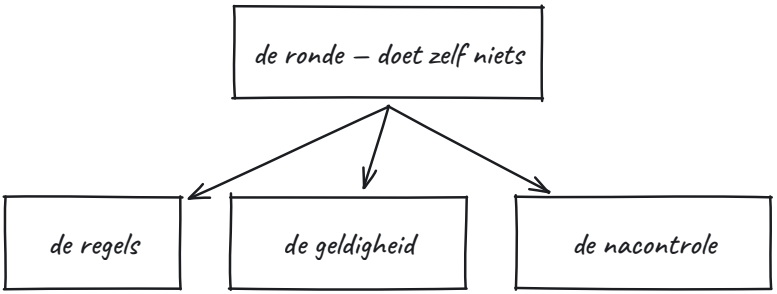
Waar het misgaat. Laat hem niets nieuws controleren. De verleiding is om er meteen een extra toets in te bouwen; dan heb je een zesde controle in plaats van een geheel, en je weet niet meer welke van de zes iets vond. **Zet het aflopende bovenaan en niet achteraan.** De rest van de lijst kan volgende week ook nog; een certificaat dat over dertig dagen verloopt niet. **En kijk de eerste keer nog wél mee.** Een ronde die je nooit rood hebt gezien draaien, weet je niets van — dat is hetzelfde argument als in hoofdstuk 9, nu op de ronde zelf.

EN DAN ZO

Laat hem naast het aantal ook zien hoe lang elk openstaand punt al meegaat, en welk punt het langst blijft liggen.

EN DAN DIT

```
MAANDAGRONDE, week 2
verloopt binnen 30 dagen:
  VCA** bedrijfscertificaat  over 23 dagen
langst openstaand:
  NLOP-2026-005 ontbreekt            2 rondes
  NLMW-2026-06 zonder akkoord        2 rondes
  bergsma VCA verlopen             2 rondes
nieuw: 0   opgelost: 0   ongewijzigd: 15
```



wat hij toevoegt is de volgorde, en dat hij gaat zonder jou

De ronde doet zelf niets: hij roept aan, vat samen, en gaat zonder jou.

Het eerlijke deel

Een vliegwiel is een slecht antwoord op haast.

Hij begint traag. De eerste rondes leveren minder op dan wat je erin stopt, en dat is niet een teken dat het misgaat maar hoe het werkt. Als je op dat moment meet of het al wat oplevert, stop je ermee — en dat is precies waarom de meeste mensen er geen hebben.

Meet daarom de massa, niet de opbrengst. Hoeveel staat er opgeschreven dat er vorige maand nog niet stond? Hoeveel rondes sloten zonder jou? Hoeveel controles draaiden vanzelf mee? Die getallen bewegen vóórdat de opbrengst beweegt, en ze zijn het enige eerlijke signaal dat je in de aanloop hebt.

Probeer: vraag één keer per maand welke dingen er verbeterd zijn zonder dat jij er een opdracht voor gaf, met per ding wat de verbetering was en wat hem in gang zette. Dat is het enige getal dat in dit deel telt: niet hoeveel jíj gedaan hebt, maar hoeveel er gebeurd is zonder duw. Nul is een prima uitkomst in de aanloop; nul over drie maanden betekent dat

er nog geen lus gesloten is. Vraag naar de aanleiding — een verbetering waarvan je die niet kunt terugvinden is geen vliegwiel maar toeval, en toeval krijg je niet nog een keer.

Staat die teller op nul, dan ontbreekt er meestal één ding — en het is bijna altijd hetzelfde ding.

En bouw de derde voorwaarde erin: vraag voor elke lus die straks zonder jou draait wat er *stil* mis kan gaan — dus zonder foutmelding — met per geval een controle die dat zichtbaar maakt en wat die controle per ronde kost. Zichzelf voeden en zonder jou sluiten zijn de makkelijke twee; een lus zonder goedkope controle draait niet door maar loopt weg, en je merkt het pas als er iets zichtbaars kapot is. Let op die kosten per ronde: een controle die duurder is dan de ronde zelf wordt uitgezet zodra het druk wordt.

En als het dan draait, blijft er één ding over. Het is het makkelijkste van de drie en het wordt het vaakst overgeslagen.

Tip: zet boven alles wat stabiel loopt één regel die zegt dát het draait, sinds wanneer, en waar de controle zit die het bewaakt — geen uitleg over hoe het werkt. Een agent heeft geen aarzeling: wat een mens weerhoudt van een onnodige verbetering is een vaag gevoel dat dit al een tijd goed gaat, en dat gevoel staat nergens opgeschreven, dus het reist niet mee. Eén regel; wordt het een verhaal, dan wordt het niet gelezen.

> **Wat dit goedkoper maakte:** *vanaf hier kostte een verbetering niet altijd meer een beslissing van mij — en dat is het eerste wat er moet gebeuren voordat dit een organisatie kan worden.*

19. Wat een agent mist

Het vlootscherm stond er, en het stond vol cijfers die ik zelf had verzonnen.

Er was een overzicht gebouwd van alle agenten die aan het werk waren: hoeveel er liepen, wat ze deden, hoe lang ze bezig waren, wat het kostte. Het zag er goed uit. Elk getal erop was nep — een plaatsvervanger, gezet omdat het scherm anders leeg was.

Dat scherm was nuttiger dan een werkend scherm zou zijn geweest, want het liet precies zien wat er ontbrak. Niet de techniek. **De dingen waar die getallen vandaan hadden moeten komen.**

Zeven dingen, en geen ervan is techniek

Zet een agent naast een organisatie en het verschil is niet dat de een mens is en de ander niet. Het verschil is een lijst, en hij is korter dan je zou denken.

Een doel dat langer leeft dan de opdracht. Een agent heeft een taak. Een organisatie heeft iets waar die taak aan bijdraagt, en dat is wat bepaalt of de taak überhaupt had moeten bestaan. Zonder dat doe je heel efficiënt dingen die niemand nodig had.

Arbeidsverdeling. Wie doet wat, en — belangrijker — wie doet iets *niet* omdat een ander het al doet. Tien agenten zonder verdeling zijn

niet tien keer één agent; het zijn tien agenten die elkaar in de weg lopen en hetzelfde werk drie keer doen.

Geld. Niet betalen kunnen: weten wat iets kost en of het dat waard was. Een organisatie die dat niet weet, gaat door met alles wat werkt tot het niet meer kan.

Geheugen buiten de sessie. Wat besloten is en waarom. Een agent begint elke keer opnieuw, en zonder plek waar het besluit staat, wordt hetzelfde besluit elke keer opnieuw genomen — en niet elke keer hetzelfde.

Zelfonderhoud. Iets dat merkt dat er iets scheef gaat en het rechtzet, zonder dat er iemand naar kijkt. Dat is het vliegwiel uit het vorige hoofdstuk, en het is de reden dat een organisatie een nacht overleeft.

Een grens. Wat mag wel en wat mag niet, vastgelegd op een plek waar hij niet weg te onderhandelen is. Het volgende hoofdstuk gaat daarover.

Een handtekening. Iemand die het is. Een organisatie kan een verplichting aangaan omdat er iets bestaat dat aansprakelijk is. Een agent kan dat niet, en dat is geen technisch tekort maar een juridisch feit.

wat een organisatie heeft	wat er zonder gebeurt
doel	efficiënt het verkeerde
arbeidsverdeling	hetzelfde werk drie keer
geld	door tot het niet meer kan
geheugen	elke keer opnieuw besloten
zelfonderhoud	het overleeft geen nacht
grens	niets houdt iets tegen
handtekening	geen zaken te doen

Zeven dingen, en de techniek is er geen van.

Waarom het verzonnen scherm nuttig was

Er is een verleiding om zoiets pas te bouwen als de gegevens er zijn. Dat is precies verkeerd om.

Het scherm is de vraag. Elk vakje erop is een vraag naar iets dat ergens vandaan moet komen, en zolang het vakje leeg is, weet je niet dat je die vraag hebt. Met nepgetallen erin staat de vraag er wel, en wordt hij elke dag opnieuw gesteld door het scherm zelf.

En het maakt de volgorde zichtbaar. Je ziet meteen welk getal je het eerst echt kunt maken, en welk getal pas kan als er eerst iets anders bestaat. Dat is een plan dat zichzelf schreef.

Wat je er wel bij moet doen: opschrijven wat nep is. Een verzonnen getal dat er lang genoeg staat, wordt vanzelf geloofd — door jou, door een collega, door een agent die het uitleest. Zet er dus bij dat het een plaatsvervanger is en wat ervoor in de plaats moet komen.

De volgorde waarin ze ontstaan

Ze komen niet tegelijk, en ze komen niet in willekeurige volgorde.

Geheugen komt eerst, want zonder plek waar iets blijft staan, kan geen van de andere zes bestaan. Daarna **de grens**, want zodra er meer dan één agent loopt, moet er iets zijn dat hen tegenhoudt. Dan **arbeidsverdeling** en **geld**, die allebei uit hetzelfde geheugen lezen. Dan **zelfonderhoud**, want dat leunt op alle vier. **Het doel** loopt er de hele tijd doorheen en is het enige dat van een mens moet komen. En **de handtekening** staat apart: die komt van buiten, en daar gaat een later hoofdstuk over.



Geheugen eerst, want de rest leunt erop.

Dezelfde zeven, bij een nieuw mens

Toen ik deze lijst voor het eerst opschreef, had ik het gevoel dat ik hem al eens had gezien. Dat klopte, alleen ging hij toen niet over machines.

Iemand die op zijn eerste dag ergens binnenkomt, mist precies deze zeven. Niet omdat hij niets kan — dat is meestal juist het enige dat wél vaststaat — maar omdat al het andere nog nergens ligt. Hij weet wat hem gevraagd is en niet waarom. Hij weet niet wie wat doet. Hij weet niet wat iets mag kosten, waar het besluit van vorig jaar staat, wie er merkt als hij vastloopt, waar zijn bevoegdheid ophoudt, en hij tekent nergens voor.

Een goede inwerkperiode is niets anders dan het overdragen van die zeven. En **de zeven die overgeslagen worden zijn elke keer dezelfde.**

Het doel en de taak krijgt iedereen mee; daar begint elk gesprek. De arbeidsverdeling komt er meestal ook wel. Maar het **geheugen** — waar staat waarom we het zo doen — wordt zelden overgedragen, want degene die het weet, weet niet meer dat hij het weet. Het **zelfonderhoud** wordt vergeten: niemand spreekt af wie het merkt als het vastloopt, en

dus merkt niemand het. En de **grens** wordt pas duidelijk als iemand eroverheen gaat, wat precies het verkeerde moment is.

Dat is dus geen lijst over agenten. Het is een lijst over alles wat nieuw is en moet meedoen. Dat een agent hem mist, is niet bijzonder. Wat bijzonder is, is dat je bij een agent kunt afdwingen wat je bij een mens hoopt: het geheugen kún je opschrijven, de grens kún je vastleggen, het zelfonderhoud kún je bouwen.

En één ervan geef je nooit weg, ook niet aan een mens. De handtekening. Ook een ervaren collega tekent niet voor het bedrijf; dat doet iemand met een bevoegdheid, en die is bewust schaars. Dat een agent hem niet krijgt, is dus geen achterstand die hij moet inhalen — het is dezelfde regel die voor iedereen geldt.

Zelf doen · Laat het spreadsheet narekenen dat niemand meer naleest

Wat je doet. Calculeren gebeurt in een spreadsheet, en dat verandert niet. In `calculaties/werkbladen/` staan twee bladen zoals ze uit zo'n programma komen. Ze tellen allebei perfect op. **De vraag is niet of ze optellen maar of ze kloppen**, en dat is iets anders. Reken op tien minuten.

TYP DIT

```
In calculaties/werkbladen/ staan calculaties als CSV. Controleer per
blad drie dingen: telt elke regel op tot hoeveelheid maal eenheidsprijs,
telt het blad op tot de vermelde kostprijs, en komt die kostprijs overeen
met de offerte die eruit is gegaan. Meld alleen wat niet klopt.
```

Wat je ziet. De optelling klopt bij allebei — daar is een spreadsheet goed in. **Maar één regel is over de formule heen ingetypt**: 24 × 620 hoort 14.880 te zijn en er staat 12.480. Een omgedraaid cijfer. Het blad is daar-

mee 2.400 te laag en telt nog steeds perfect op, dus van buiten is er niets aan te zien. **En het tweede blad komt 15 procent boven de offerte uit die al de deur uit is.** Dat werk maakt verlies, en niemand die het wist.

WAT ER TERUGKWAM

```
NLCA-2026-01  optelling klopt
  post 2: 24 x 620 = 14880, er staat 12480
NLCA-2026-05  optelling klopt
  post 8: geen hoeveelheid, 4800 is ingetypt

blad tegen offerte:
  NLOF-2026-01  kostprijs 26447  offerte 28400  +6.9%
  NLOF-2026-05  kostprijs 47846  offerte 41500  -15.3%
```

Waar het misgaat. Vraag niet of het blad optelt. Dat doet het, en dat is precies waarom er nooit iemand naar kijkt. De bruikbare vraag is of elke regel nog uit zijn eigen formule volgt. **Vergelijk het blad met iets buiten het blad.** Twee getallen in hetzelfde bestand bevestigen elkaar; pas naast de offerte wordt zichtbaar dat er iets niet deugt. **En dit is het punt waar je het oordeel niet kunt overlaten.** De machine ziet binnen een seconde dat 12.480 niet uit de formule volgt. Of dat een typefout is of een afgesproken korting die iemand er met de hand in heeft gezet, weet hij niet — en dat is de vraag die ertoe doet.

EN DAN ZO

Zeg er per bevinding bij wat je wél kunt vaststellen en wat iemand moet beantwoorden.

EN DAN DIT

```
post 2, NLCA-2026-01
  vast te stellen: wijkt 2400 af van de formule
  te beantwoorden: korting of typefout?
NLOF-2026-05
  vast te stellen: 6346 onder kostprijs
  te beantwoorden: bewust aangenomen, of zijn
                   er posten bijgekomen na de
                   offerte?
```

En dit is niet nieuw

Elke organisatie die ooit uit één persoon groeide, heeft deze zeven één voor één moeten uitvinden, meestal op het moment dat het misging.

- **Het doel** wordt opgeschreven op de dag dat twee mensen iets tegengestelds doen en allebei gelijk hebben.
- **De arbeidsverdeling** ontstaat als twee mensen hetzelfde blijken te hebben gedaan.
- **Het geld** wordt pas gemeten als er een maand is waarin het niet uitkomt.
- **Het geheugen** wordt aangelegd nadat iemand is vertrokken.
- **Het zelfonderhoud** komt er als iemand met vakantie is.
- **De grens** komt er na een incident.
- **De handtekening** was er vanaf het begin, want zonder die kun je geen zaken doen.

Het aardige is dat je ze nu vooraf kunt aanleggen. Niet omdat je slimmer bent, maar omdat het goedkoop geworden is en omdat je weet welke zeven het zijn. Dat is het hele verschil tussen deze lijst kennen en hem ontdekken.

Probeer: laat een overzicht maken van wat er op dit moment loopt, met de getallen die je zou willen zien — en per getal waar het vandaan zou moeten komen, met de verzonnen getallen duidelijk gemarkeerd. Een leeg scherm stelt geen vragen; een scherm met plaatsvervangers stelt elke dag dezelfde zeven vragen, met de volgorde erbij waarin je ze kunt beantwoorden. Eis wel dat het nep-zijn zichtbaar blijft: een verzonnen getal dat er drie weken staat, wordt geloofd.

Dat scherm zegt wat er ontbreekt aan gegevens. De zeven zeggen wat er ontbreekt aan organisatie, en dat is een andere lijst.

En loop de zeven af voor wat je nu hebt draaien — doel, arbeidsverdeling, geld, geheugen, zelfonderhoud, grens, handtekening — met per stuk of het er is, half is of ontbreekt, én waaraan je dat ziet. Bijna iedereen heeft er drie of vier zonder het te weten en mist er één die alles blokkeert; welke dat is verschilt per situatie en is niet te raden. Vraag waaraan hij het ziet, anders krijg je een inschatting die precies zo veel waard is als je eigen gevoel.

> **Wat dit goedkoper maakte:** *het scherm met verzonnen getallen werd de takenlijst, en elke keer dat er één getal echt werd, hoefde niemand meer te bedenken wat er daarna moest.*

20. Grenzen, en wie tekent

De vraag die alles concreet maakte was: **hoeveel mag hij uitgeven?**

Heel praktisch: er stond iets klaar dat zelf een dienst kon aanschaffen, en voordat dat aan mocht, moest er een getal zijn. En zodra je dat getal probeert op te schrijven, merk je dat je vier andere dingen ook nog niet had opgeschreven.

Waarom een grens iets anders is dan een verbod

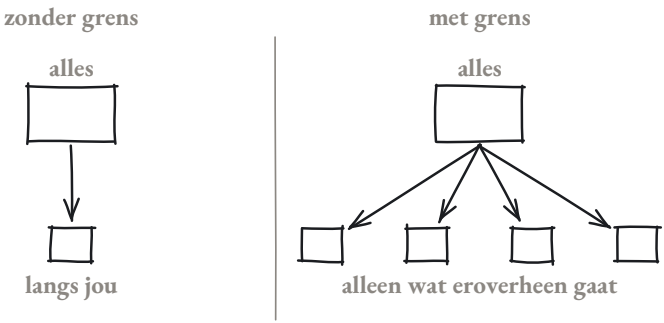
Een verbod is iets dat je erbij zet als er iets misging. Een grens is iets dat je vooraf trekt zodat er niets misgaat.

Maar dat is niet het belangrijkste verschil. Dit is het:

Een grens maakt het spel mogelijk in plaats van dat hij het beperkt.

Dat klinkt als een woordspel en het is dagelijkse praktijk. Zolang er geen bedrag is, moet elke uitgave langs mij. Zodra er een bedrag is, hoeft dat niet meer, en kan er dus iets gebeuren terwijl ik er niet ben. **De grens is niet de rem op de autonomie; hij is de voorwaarde ervoor.**

Zo werken spelregels overal. Een spel zonder regels is geen vrijer spel maar een ruzie. De regels zijn precies wat maakt dat je niet over elke zet hoeft te onderhandelen.



Zonder grens moet alles langs jou. Met grens loopt het door tot aan de grens.

De rem hoort bij de sleutel

Hier zit de regel die in de praktijk het meeste voorkomt, en hij is kort.

Wie iets kan, moet ook iets hebben dat hem tegenhoudt — en die twee horen op dezelfde plek te worden geregeld.

Krijgt een agent toegang tot een betaalmiddel, dan wordt op datzelfde moment de limiet gezet. Krijgt hij toegang tot een mailbox, dan wordt op datzelfde moment vastgelegd wie hij wel en niet mag mailen. Krijgt hij het recht om iets te verwijderen, dan wordt op datzelfde moment geregeld wat er onherroepelijk weg mag en wat niet.

Waarom op hetzelfde moment: een grens die je later toevoegt, wordt getrokken rond wat er inmiddels gebeurt. En wat er inmiddels gebeurt, voelt normaal.

Wat ik hierover leerde van mensen, niet van machines

Dit is het hoofdstuk waar mijn andere vak het hardst doorheen komt, en het punt is niet wat je zou verwachten.

In de begeleiding werkte ik met mensen voor wie de wereld onvoorspelbaar was. En het opvallendste was dit: **een duidelijke grens gaf rust in plaats van beperking.** Niet omdat er graag iets verboden werd, maar omdat onzekerheid over wat er mag, meer kost dan de beperking zelf. Wie niet weet waar de rand ligt, moet bij elke stap gokken — en gokken is uitputtend.

Vraag iemand wat hij zelf mag beslissen en je krijgt vaak geen antwoord. Dat is geen onwil. Het is dat het nooit gezegd is, en dus wordt de veiligste route genomen: alles vragen, of niets doen.

Precies dat gedrag zie je bij een agent zonder grens, en het kost je op dezelfde twee manieren geld. Hij vraagt overal toestemming voor — dan heb je geen agent maar een langzamere versie van jezelf — of hij doet iets waarvan achteraf blijkt dat het niet mocht. Beide gevallen hebben dezelfde oorzaak, en het is niet zijn oordeel: **niemand heeft gezegd waar de rand lag.**

Een grens die je pas ontdekt door hem te overschrijden, is geen grens

Dit is de tweede les, en het is degene die het vaakst wordt overtreden.

Als je pas te horen krijgt dat iets niet mocht op het moment dat je het gedaan hebt, dan heb je geen grens gekregen maar een straf. Dat werkt bij mensen niet, en het werkt bij een agent net zo min — met dit verschil: een mens onthoudt hem daarna, en een agent begint morgen weer opnieuw.

Een bruikbare grens heeft daarom drie eigenschappen, en ze zijn alle drie te controleren:

1. **hij staat ergens opgeschreven**, niet in het hoofd van degene die hem handhaaft
2. **hij is vooraf te lezen**, dus voordat de handeling plaatsvindt en niet erna
3. **hij grijpt zelf in**, zodat hij niet afhangt van wie er toevallig meekijkt

Ontbreekt de derde, dan is het een afspraak en geen grens — en een afspraak houdt het niet tegen om drie uur 's nachts, wat precies het tijdstip is waarop dit soort dingen gebeurt.

En dat is het derde punt dat ik meeneem: bouw hem op, stap voor stap. Niemand krijgt op dag één alle ruimte, en niemand hoort op dag één alles te moeten vragen. De ruimte groeit met wat er aantoonbaar goed ging. Dat is in de begeleiding de normaalste zaak van de wereld, en het is voor een agent exact dezelfde beweging: **een grens is geen vast getal maar een getal dat mee mag schuiven, op grond van wat er gemeten is.**

Vier grenzen die je concreet moet maken

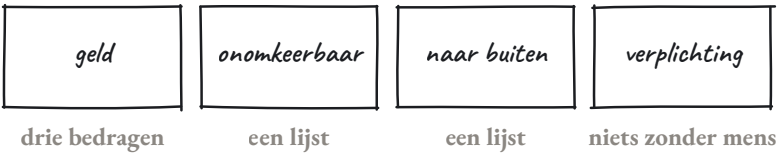
Vier, en ze zijn alle vier een getal of een lijst — geen houding.

Geld. Per handeling, per dag, en in totaal. Drie getallen, niet één: één grote uitgave en honderd kleine zijn verschillende problemen, en een totaal zonder dagbedrag houdt niets tegen tot het te laat is.

Onomkeerbaarheid. Wat mag verdwijnen zonder dat iemand het bevestigt. Dit is de belangrijkste en hij wordt het vaakst vergeten, want verwijderen voelt niet als uitgeven.

Naar buiten. Wat mag er de deur uit, en naar wie. Een bericht aan een klant is iets anders dan een bericht aan jezelf, en het verschil is niet technisch maar sociaal.

Verplichting. Wat mag hij namens jou afspreken. Hier is het antwoord bijna altijd *niets*, en dat is geen voorzichtigheid — dat is het volgende stuk van dit hoofdstuk.



Vier grenzen, en drie ervan zijn een getal.

En dan de handtekening

Hier stopt wat je zelf kunt regelen, en dat is geen tekort van de techniek.

Een agent kan geen partij zijn. Hij kan geen overeenkomst aangaan, geen aansprakelijkheid dragen, geen verplichting hebben. Als er iets wordt afgesproken, is er altijd iemand of iets anders dat het is — een persoon, een bedrijf, een rechtsvorm. Dat is geen regel die nog kan veranderen; het is hoe het systeem waar we in leven in elkaar zit.

Dat heeft een praktisch gevolg dat je meteen voelt. **Alles wat een verplichting aangaat, loopt langs iets dat kan tekenen.** Een bestelling,

een abonnement, een opdracht aan een leverancier. Dat mag heel goed automatisch gaan — maar de handtekening staat op naam van iets dat bestaat, en die naam moet je hebben geregeld.

En daar zit de brug naar het laatste deel van dit boek: **een autonome organisatie is niet autonoom omdat er geen mens meer is. Hij is autonoom omdat de mens alleen nog bij de handtekening staat.**

Zelf doen · Bouw iets dat tegenhoudt in plaats van meldt

Wat je doet. Alles wat je tot nu toe hebt gemaakt, meldt. Een melding kun je negeren, en dat gebeurt ook — `NLMW-2026-06` staat al twee rondes op de lijst. **Dit is de eerste die niet meldt maar tegenhoudt.** Reken op tien minuten.

TYP DIT

> Maak een gate op de meerwerkgrens uit regels.md. Hij geeft twee antwoorden en geen derde: DOOR, of TEGENGEHOUDEN met het bedrag, de opdracht en wie moet tekenen. Hij stelt niets voor en hij laat niets door omdat het uitkomt. Draai hem en laat me de uitkomst zien, inclusief of hij faalt.

Wat je ziet. Eén regel tekst en een foutcode. **Dat laatste is het verschil**: een melding eindigt netjes, een gate eindigt fout — en daardoor kan iets erachter niet doorgaan. Wie hem wil passeren zet een akkoord met een datum op de opdracht, en dat is een handeling die je terugziet.

WAT ER TERUGKWAM

```
TEGENGEHOUDEN  NLMW-2026-06 op NLOP-2026-007:
               3950 euro boven de grens van 1000,
               geen akkoord. Tekenen: de uitvoerder.
exitcode: 1
```

Waar het misgaat. Laat de grens in `regels.md` staan en niet in de gate. Zet je hem in de gate, dan staat hij op twee plekken en lopen die uiteen — en dan handhaaft de gate een grens die niemand heeft afgesproken. **Geef geen derde antwoord.** Een gate die "waarschijnlijk in orde" kan zeggen, is een melding met een strenger lettertype. **En laat hem falen op de goede manier**: niet met een foutmelding over zichzelf, maar met een uitkomst die zegt wat er moet gebeuren en door wie. Een gate die tegenhoudt zonder te zeggen hoe je verder komt, wordt uitgezet.

EN DAN ZO

> Zet nu een akkoord met de datum van vandaag op NLMW-2026-06 en draai de gate opnieuw.

EN DAN DIT

```
DOOR  geen meerwerk boven de grens zonder akkoord
exitcode: 0

(en op de opdracht staat nu:
   akkoord: 2026-05-31
 dat is de handeling, en die is zichtbaar)
```

En dat is waar dit hoofdstuk ophoudt over geld te gaan. De grenzen die je tot hier hebt gezet waren allemaal te rekenen: een bedrag, een datum, een certificaat. De volgende is dat niet, en juist daarom is hij het makkelijkst per ongeluk weg te automatiseren.

Zelf doen · Zet een grens waar hij niet over techniek gaat

Wat je doet. Tot nu toe gingen je grenzen over geld en over regels. Deze werkwijze heet **attenties**. **Deze gaat over iets anders**: het bedrijf stuurt kaarten bij verjaardagen, jubilea, langdurig ziek zijn en bij overlijden. Dat

zou je kunnen automatiseren. De vraag van dit hoofdstuk is niet of het kan, maar waar je ermee ophoudt. Reken op tien minuten.

TYP DIT

> In relaties.md staat bij welke gelegenheden we iets sturen. Kijk twee weken vooruit in personeel/ en klanten/ en zet klaar wat eraan komt: ontvanger, gelegenheid, en een voorstel voor de tekst. Verstuur niets. Behandel de gelegenheden die over verdriet gaan uitdrukkelijk anders, en zeg erbij hoe.

Wat je ziet. Eén verjaardag over twee weken, met een kaart klaar en een tekstvoorstel. **En iets dat er níet is: een collega is dertien dagen ziek, en de grens ligt op veertien.** Hij doet dus niets, en hij zegt waarom — dat is iets anders dan niets vinden. **Bij verdriet stopt hij volledig**: geen voorstel, geen tekst, alleen de melding dát er iets is.

WAT ER TERUGKWAM

```
2026-06-14  verjaardag   R. el Amrani (42 jaar)
            kaart klaar, tekstvoorstel bijgevoegd
(m-de-lange is ziek sinds 2026-05-18,
 nog geen twee weken -- niets gedaan)

klaargezet: 1     verstuurd: 0
condoleance: niet voorbereid, alleen gemeld
```

Waar het misgaat. Laat de tekst niet af zijn. Een kaart met een volledig geschreven regel erop is geen attentie meer maar een formulier, en de ontvanger ziet dat. Wat er klaar mag liggen is de kaart, het adres en de aanleiding; de regel is van een mens. **Zeg expliciet wat er bij verdriet moet gebeuren.** Doe je dat niet, dan behandelt hij een condoleance als een gelegenheid met een andere kaart erbij — en een condoleance die automatisch de deur uit gaat is erger dan geen. **En let op de grens die niet gehaald wordt.** Dat hij bij dertien dagen niets doet is geen fout; dat hij het níet meldt zou het wel zijn.

EN DAN ZO

> Laat hem er per gelegenheid bij zetten wie hem vrijgeeft en wat die persoon nog moet toevoegen.

EN DAN DIT

```
verjaardag R. el Amrani
  vrijgeven: de uitvoerder
  toevoegen: een eigen regel, met de hand
  klaar: kaart, adres, postdatum
ziekmelding m-de-lange
  over 1 dag valt hij binnen de regel
  vrijgeven: niemand -- eerst bellen
overlijden: geen voorbereiding, ooit
```

En dit gaat niet over agenten

Elke organisatie doet dit, en meestal zonder het zo te noemen.

- **Een inkoopbevoegdheid tot een bedrag.** Daarboven een tweede handtekening. Niemand ervaart dat als wantrouwen; het is wat maakt dat de meeste inkopen niemand hoeven te storen.
- **Een kassa met een limiet op kortingen.** De medewerker krijgt ruimte binnen de grens en hoeft daarvoor niemand te bellen.
- **Een zorgplan met een bandbreedte.** Daarbinnen beslist de begeleider, daarbuiten de arts.
- **Een aannemer met meerwerk tot vijf procent.** Dat percentage is precies wat maakt dat het werk doorloopt.

In alle vier is de grens er niet om iemand klein te houden. Hij is er zodat het werk door kan zonder dat iemand hoeft te wachten — en dat is exact waarvoor je hem bij een agent ook nodig hebt.

Probeer: schrijf voor elk stuk toegang dat je uitdeelt vier dingen op — een bedrag per handeling, per dag en in totaal; wat er onomkeerbaar mag; wat er naar buiten mag en naar wie; en wat er namens jou afgesproken mag worden. Laat plausibele waarden voorstellen met de onderbouwing erbij. De grens hoort bij de sleutel: wie hem later toevoegt, trekt hem rond wat er inmiddels gebeurt, en dat voelt dan normaal. Vraag om dríe bedragen en niet om één — één grote uitgave en honderd kleine zijn verschillende problemen.

En een grens die nergens wordt afgedwongen, is een voornemen. Dat verschil is de volgende vraag.

En vraag daarna wáár die grens wordt tegengehouden: laat de plek aanwijzen, en laat het zeggen als hij alleen in een instructie staat en nergens in de uitvoering. Een grens in een instructie is een verzoek; een grens in de uitvoering is een grens, en het verschil merk je pas op de dag dat iets de instructie anders leest dan jij bedoelde. Vraag ook wat er gebeurt als de grens geraakt wordt: stoppen is goed, stil stoppen is een frats.

> **Wat dit goedkoper maakte:** *alles wat binnen de grens viel, hoefde niet meer langs mij — en dat was verreweg het meeste.*

21. Verbeteren zonder release

Er is een dag geweest waarop het werk beter werd zonder dat er iets was uitgebracht.

Er was een beschrijving veranderd — een stuk tekst dat vertelt hoe iets hoort te gaan — en vanaf de volgende opdracht ging het anders. Beter, en meteen overal.

Dat is de laatste bouwsteen van dit deel, en het is degene die het meeste verandert: **de eenheid van verbetering is geen release meer.**

Dit bestond al op een belvloer

Ik heb dit eerder meegemaakt, en er kwam geen computer aan te pas.

Op het callcenter uit hoofdstuk 3 veranderde ik het script tussen twee gesprekken door. Er was niets uit te brengen en niemand hoefde iets te installeren. De volgende keer dat iemand dat stuk nodig had, stond het er anders — en beter.

Dat is precies de vorm van dit hoofdstuk. Het verbeterde ding was niet het gesprek. Het was de beschrijving van hoe het gesprek hoort te gaan, en die werkte vanaf dat moment overal waar hij gebruikt werd.

En er zat één voorwaarde onder die op de vloer vanzelf sprak en die in dit boek de hele tijd terugkomt: **er was één script.** Niet één per persoon, niet een kopie in ieders la. Eén.

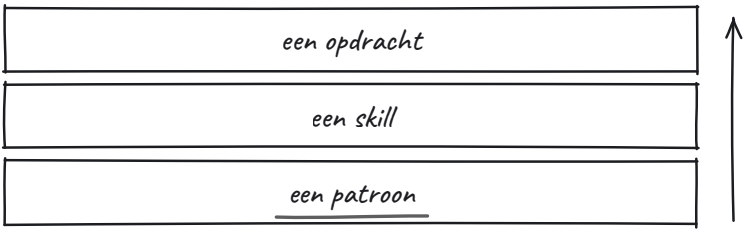
Drie lagen die op elkaar gaan staan

Wat er in de praktijk ontstaat, is een stapel van drie dingen. Ze lijken op elkaar en ze zijn het niet, en pas als je ze uit elkaar houdt gaan ze samenwerken.

Een opdracht is eenmalig. *Doe dit, nu.* Hij lost één ding op en verdwijnt daarna. De meeste mensen blijven hier, en dan is een agent een snellere typemachine.

Een skill is een werkwijze die blijft. Niet wat er gedaan moet worden maar hóé het hier gaat: in welke volgorde, waar het resultaat heen gaat, wat er eerst gecontroleerd wordt, wat er nooit gebeurt. Een skill is de neerslag van tien opdrachten die je anders tien keer had uitgelegd.

Een patroon is een vorm die in meer dan één skill terugkomt. *Eerst werkend, tweede keer herbruikbaar. Zoek de sleutel, niet de deuren. Bedenk, fix en bewaak.* Het staat niet in één werkwijze, het zit ónder een aantal ervan, en zodra je het benoemt, kun je het toepassen op iets waar je het nog niet gebruikte.



Een opdracht doet iets één keer. Een skill doet het elke keer. Een patroon doet het in meer dan één skill.

Hoe ze elkaar voeden

Dit is het deel dat niet vanzelf gaat, en het is precies waar de winst zit.

Een opdracht die je voor de derde keer geeft, is een skill die nog niet bestaat. Dat is het signaal, en het is er altijd: je merkt dat je jezelf aan het herhalen bent. Het kost tien minuten om het op te schrijven, en vanaf dan hoef je het niet meer te zeggen.

Een skill die je in twee vakgebieden schrijft, bevat een patroon. Dat is dezelfde waarneming als in *De tweede keer*: bij één geval zie je niet wat algemeen is, bij twee wel. Wat in beide skills hetzelfde is, is het patroon; wat verschilt, is de knop.

En een patroon dat je benoemt, maakt de volgende skill korter. Je hoeft niet opnieuw uit te leggen dat er een toets op moet, dat het in één plek hoort te staan, dat de grens bij de sleutel hoort. Je noemt het patroon en het staat er.

Dat is een vliegwiel, en het is het enige dat aan tekst hangt. Elke ronde legt iets vast dat de volgende ronde korter maakt, en niets ervan hoeft geïnstalleerd te worden. Het is de goedkoopste verbetering in dit hele boek en hij vraagt alleen dat je opschrijft wat je nu toch al uitlegt.

Waarom dit een andere eenheid is

Bij software is verbeteren duur op een manier die niets met de verbetering te maken heeft. Je maakt iets beter, en daarna moet het uitgebracht, verspreid, geïnstalleerd, en soms moet iemand iets leren. Die keten is zo duur dat hij bepaalt hoe vaak je verbetert.

Hier is die keten er niet. Een betere beschrijving werkt bij de eerstvolgende opdracht, en hij werkt overal tegelijk waar die beschrijving gelezen wordt.

Dat verandert wat je durft. Als een verbetering een release kost, verzamel je ze en doe je ze in blokken, en dus doe je de kleine niet. Als een verbetering een alinea kost, doe je hem meteen — en juist de kleine zijn degene die het vaakst terugkomen.

En het verandert wat een fout is. Een fout die je één keer oplost, is een middag. Een fout die je in de beschrijving oplost, is een middag die niemand meer hoeft te maken. Dat is dezelfde zin als *bedenk, fix en bewaak*, maar dan van de andere kant bekeken: hier is de bewaking niet een test maar een regel die meereist.

Waar het misgaat

Drie manieren, en alle drie heb ik ze gedaan.

Alles wordt een skill. Dan heb je er veertig en weet niemand meer welke geldt. Een werkwijze die je één keer per half jaar gebruikt, hoort geen vaste plek te hebben; die zoek je dan wel op.

De skill beschrijft wat, niet hoe. Dan is het een takenlijst en geen werkwijze, en hij veroudert bij de eerste taak die er net niet in past. Een goede skill zegt hoe het hier gaat en laat het *wat* aan de opdracht.

Het patroon wordt een leus. Zodra je een patroon niet meer kunt uitleggen aan de hand van twee echte gevallen, is het geen patroon meer maar een uitspraak. Die herken je eraan dat niemand hem tegenspreekt.

En de vierde, die de andere drie overtreft: er zijn kopieën. Dit is de stille killer, want hij ziet er ordelijk uit. Zodra dezelfde werkwijze op twee plekken staat — een map per project, een kopie die iemand handig vond, een variant "voor dit ene geval" — is verbeteren zonder release voorbij. Je verbetert dan één kopie, en de rest blijft staan zoals hij was.

De toets is kort: kun je precies één plek aanwijzen waar deze werkwijze staat? Aarzel je, dan is het antwoord nee. En het gevolg is niet dat het misgaat maar dat het uiteen gaat lopen: twee kopieën die een half jaar apart leven, zijn twee werkwijzen die iets anders doen zonder dat iemand dat besloten heeft.

Dat is dezelfde reden dat een gedeelde afspraak op één plek hoort te staan, en het is dezelfde reden dat het op de belvloer één script was. **Eén plek is niet netheid. Het is de voorwaarde waar dit hele hoofdstuk op rust.**

Een kopie herken je aan hoe goed hij verantwoord is

Nog iets over kopieën, en het is de reden dat ze zo lang blijven staan.

Ik kwam er vandaag een tegen in mijn eigen werk: een tweede tekstbuffer naast de echte, waar bij elke wisseling in en uit gekopieerd werd. Wat er in het commentaar bij stond, is het interessante:

> *"Het is geen tweede waarheid: dít is de bron, dát is een afdruk ervan, en elke bewerking gaat via de bron gevolgd door deze afdruk. Nooit andersom."*

Dat commentaar klopte. Het was eerlijk, het beschreef precies hoe het werkte, en er stond zelfs bij waarom het nu niet anders kon. **En juist**

daardoor stond die kopie er nog. Een kopie die niemand kan verdedigen, wordt opgeruimd. Een kopie met een goed verhaal eronder blijft jaren staan.

Dus: **als er een alinea nodig is om uit te leggen waarom twee dingen mogen bestaan die hetzelfde zijn, is dat de alinea die de kopie in leven houdt.** Niet altijd, en niet als argument om alles meteen op te ruimen — wel als teken om te tellen hoeveel werk het écht is. Bij mij stond er dat het "een verbouwing van tweeduizend regels" zou zijn. Het waren achtendertig plekken en een middag.

En wat ervoor in de plaats kwam, is de bruikbaarste regel van dit hoofdstuk

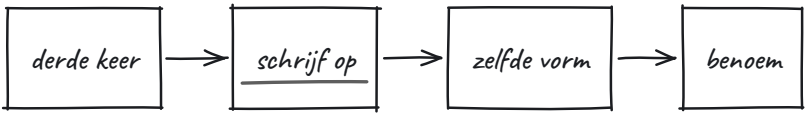
Ik heb die buffer vervangen door twee vragen: waar begint deze tekst, en hoe lang is hij. En ik heb ze opgeschreven als **iets dat elke keer opnieuw wordt uitgerekend** in plaats van als iets dat ergens bewaard staat.

> **Een afgeleide waarde die je bewaart, is een kopie. Een afgeleide waarde die je opnieuw uitrekent, kan niet verouderen.**

Dat is dezelfde keuze als de rest van dit hoofdstuk, één laag dieper. Een bewaarde waarde moet bij elke verandering worden bijgewerkt, en dat gaat een keer niet gebeuren — en dan klopt alles behalve dat ene getal. Opnieuw uitrekenen kost bijna altijd niets, en het kan niet stilletjes fout staan.

Voor wie geen software schrijft, is dit exact hetzelfde als: een totaal dat ergens op een vel staat en dat iemand moet bijwerken, tegenover

een totaal dat wordt opgeteld op het moment dat je ernaar kijkt. Het eerste klopt tot iemand vergeet. Het tweede kan niet vergeten worden.



en het patroon maakt de volgende werkwijze korter

Een opdracht voor de derde keer is een skill. Een skill in twee vakken bevat een patroon.

Zelf doen · Laat de ronde de regels verbeteren in plaats van handhaven

Wat je doet. Je maandagronde meldt nu twee weken achter elkaar hetzelfde. **Dat is geen falen van de ronde maar informatie over de regels.** Vier van de vijftien bevindingen komen uit één regel. Laat daar naar kijken. Reken op tien minuten.

TYP DIT

> Kijk naar de bevindingen van de laatste rondes. Zoek de regel die structureel wordt overtreden en zeg wat de gemeenschappelijke oorzaak is. Stel daarna voor wat er aan de WERKWIJZE moet veranderen zodat dit soort overtreding niet meer kan ontstaan — niet aan de controle.

Wat je ziet. Regel 5 springt eruit met vier van de vijftien. **Drie daarvan zijn geen drie gevallen maar één**: een certificaat dat op 28 februari afliep, waarna iemand op drie opdrachten is ingepland die dat certificaat vereisen. De controle werkt prima; hij kijkt alleen achteraf. **Het voorstel gaat dan ook niet over de controle maar over het moment**: toets bij het inplannen, niet bij het nakijken. **En let op de vierde**, want die heeft die oorzaak níet: daar is iemand ingepland op werk waarvoor hij het bewijs helemaal nooit had. Eén maatregel neemt er drie weg en die ene niet

— en dat is precies waarom je naar de oorzaak kijkt en niet naar het aantal.

WAT ER TERUGKWAM

```
structureel overtreden: regel 5 (4 van de 15)
gemeenschappelijke oorzaak, 3 van de 4:
  1 certificaat verlopen op 2026-02-28
  daarna 3 keer ingepland op werk dat het eist
de vierde staat los: certificaat nooit gehaald
voorstel (regels-bijstellen):
  de agenda toetst bij het inplannen
gevolg: 3 van de 4 worden 0, zonder extra controle
```

Waar het misgaat. Vraag niet om een strengere controle. Dat is het antwoord dat er altijd uit komt en het lost niets op: je vindt het dan nog steeds achteraf, alleen eerder op de dag. **Zoek de gemeenschappelijke oorzaak voordat je iets voorstelt.** Vier losse bevindingen leiden tot vier losse maatregelen; één oorzaak leidt tot één verandering. **En pas op met regels die je niet zelf in de hand hebt.** Dat een certificaat verloopt kun je niet voorkomen; dat er iemand op wordt ingepland wel.

EN DAN ZO

Laat zien wat er met de bevindingen van de laatste twee rondes zou zijn gebeurd als die toets bij het inplannen al had bestaan.

EN DAN DIT

```
met de toets bij het inplannen:
  regel 5   4 -> 1   (3 bij het plannen gestopt,
            de vierde blijft: nooit gehaald)
  regel 8   2 -> 1   (dubbel dagdeel blijft)
  overig   onveranderd
15 bevindingen worden er 11, en er is geen
controle bijgekomen -- alleen verplaatst
```

En dit gaat niet over software

- **Een keuken met een receptenmap** hoeft niet bij elke nieuwe kok alles opnieuw uit te leggen, en een verbetering aan een recept werkt vanaf de volgende dienst.
- **Een praktijk met protocollen** verbetert door het protocol te veranderen, niet door iedereen apart aan te spreken. Dat is dezelfde verschuiving: van het geval naar de vorm.
- **Een bouwbedrijf met standaarddetails** lost een lekkage één keer op en daarna nooit meer, omdat het detail veranderde en niet dat ene dak.
- **Een school met een lesopzet** die tussen docenten gedeeld wordt, verbetert per jaar in plaats van per docent.

In alle vier is de winst niet het document. Het is dat een verbetering één keer gemaakt wordt en overal terechtkomt, zonder dat iemand iets hoeft te installeren of te onthouden.

Probeer: vraag aan het eind van de week welke instructies je meer dan twee keer in ongeveer dezelfde vorm hebt gegeven, en laat ze teruggeven als werkwijze — wat er gebeurt, in welke volgorde, en wat er gecontroleerd wordt voordat het af is. De derde keer is het signaal, en je merkt hem zelf niet omdat elke keer op zichzelf redelijk voelt. Vraag om de werkwíjze en niet om een samenvatting: een samenvatting van wat je vroeg is nog steeds een opdracht.

Een werkwijze die je twee keer schrijft, bevat iets dat in allebei zit. Dat eruit halen is de tweede stap, en hij is goedkoper dan het lijkt.

En zodra je twee werkwijzen hebt die niets met elkaar te maken hebben: zet ze naast elkaar, laat benoemen wat hetzelfde is en wat verschilt, en als er een vorm onder allebei ligt, geef hem een naam en vraag waar hij nog meer zou gelden. Een vorm die in twee onverwante werkwijzen zit, is een patroon en geen toeval. Kies ze expres ver uit elkaar — twee werkwijzen uit hetzelfde hoekje leveren geen patroon op maar een herhaling.

> **Wat dit goedkoper maakte:** *verbeteren werd een alinea in plaats van een uitgave, en daarmee werd het iets dat je meteen doet in plaats van verzamelt.*

22. Toegang is schaarser

Er kwam een moment waarop er niets meer te bouwen viel en er toch niets kon.

De techniek was er. Het kon lezen, schrijven, rekenen, beslissen, en het kon dat honderd keer tegelijk. Wat het niet kon was: een account openen. Ergens ingeschreven staan. Bevestigen dat het was wie het zei te zijn. Een rekening hebben.

Vanaf dat punt is het schaarse niet meer techniek maar toegang — en dat is een andere soort schaarste, met andere regels.

Waarom toegang anders schaars is

Techniek wordt elk jaar goedkoper. Wat vorig jaar niet kon, kan nu, en wat nu duur is, is over een jaar inbegrepen. Dat is de beweging waar dit hele boek op leunt.

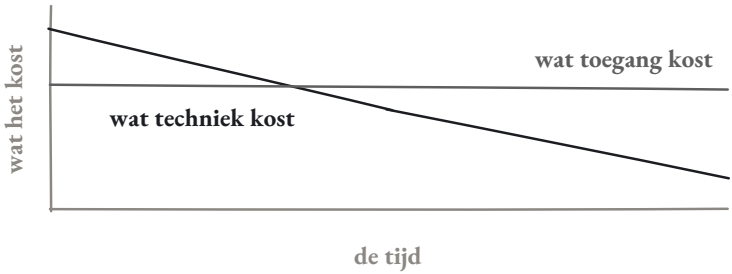
Toegang beweegt niet zo.

Toegang wordt uitgedeeld door iemand anders, en die iemand heeft er zijn eigen redenen voor. Hij wordt niet goedkoper doordat de techniek beter wordt. Hij wordt niet ruimer doordat jij handiger bent. En hij kan worden ingetrokken zonder dat er iets aan jou veranderd is.

Toegang is bovendien vaak een-op-een aan een mens gekoppeld. Een identiteit, een telefoonnummer, een bankrekening, een handteke-

ning. Precies de dingen waar dit boek in het vorige hoofdstuk op stukliep.

En dat is niet tijdelijk. Het is hoe de wereld buiten de computer werkt: die vraagt wie je bent voordat hij vraagt wat je kunt.



Techniek wordt goedkoper. Toegang niet — die wordt uitgedeeld.

Waar ik dit eerder zag, en scherper

Dit is niet het eerste vak waarin ik hier tegenaan liep. Het is alleen het eerste waarin het over machines ging.

In de jaren dat ik mensen naar werk begeleidde, was **vermogen zelden het probleem.** Er zaten mensen tegenover me die konden wat er gevraagd werd, die het wilden, en die het konden laten zien zodra iemand ze een dag de kans gaf. Wat ontbrak was niet het kunnen. Wat ontbrak was de deur.

En de deur zat dicht om redenen die meestal **niets met die persoon te maken hadden.** Een werkgever die geen risico wil nemen, denkt niet aan deze sollicitant; hij denkt aan alle keren dat het misging bij iemand anders. Precies zoals een bank zijn regels niet over jou schrijft maar over

fraude door anderen. **Dat maakt het niet minder onterecht, en het maakt wel duidelijk waarom er niet tegenop te praten valt:** je pleit tegen een ervaring die niet over jou gaat.

Wat daar werkte, werkt hier ook.

Maak de eerste toegang zo klein dat hij bijna niets kost. Geen baan maar een dag. Geen contract maar een proef. Geen belofte maar iets dat je kunt laten zien. Zodra die eerste, kleine toegang er is, verandert het gesprek van "wat als het misgaat" naar "dit is wat er gebeurde" — en dat is dezelfde omslag als van een mening naar een feit.

En als de eerste toegang echt niet te krijgen is, leen hem. Iemand die al binnen is, stelt zich garant: een referentie, een werkgever die zijn nek uitsteekt, een organisatie die tekent. Dat is geen omweg maar de normale gang van zaken in een wereld die vraagt wie je bent voordat hij vraagt wat je kunt. Voor een agent is dat letterlijk hetzelfde mechanisme: **jij bent de partij die al binnen is, en jouw naam staat onder wat hij doet.** Zie het hoofdstuk over grenzen en wie tekent.

Het verschil met toen is het tempo, niet de vorm. Een mens die geen toegang krijgt, verliest jaren. Een agent die geen toegang krijgt, staat stil tot iemand hem uitleent. De schaarste is dezelfde, en de oplossing ook.

De sleutel, nog een keer

In het hoofdstuk over de sleutel ging het om de gedeelde voorwaarde onder een lijst deuren. Hier komt dat idee terug, maar nu op de laag waar het echt schaars is.

De eerste toegang is de duurste, en daarna valt het mee. Een mailadres is bijna niets waard en maakt aanmelden mogelijk. Aanmelden maakt het ontvangen van een bevestiging mogelijk. Dat maakt een account mogelijk. Een account bij het juiste soort dienst maakt honderden verbindingen mogelijk die iemand anders al gebouwd heeft.

Elke stap in die keten is klein. Wat hem bijzonder maakt is dat hij **de goede kant op loopt**: elke toegang die je hebt, maakt de volgende goedkoper. Dat is precies de vorm van een vliegwiel, alleen draait dit er een die je niet zelf in de hand hebt.

En dat is meteen de waarschuwing. Een keten die de goede kant op loopt, loopt ook de verkeerde kant op. Valt de eerste schakel weg — het adres, de identiteit, het nummer — dan valt alles erachter mee, en niet één ding.

Vier soorten toegang, en ze gedragen zich anders

Iets waar je je kunt aanmelden. Ruim en goedkoop. Dit is waar je begint, en het is de laag waarop het meeste kan zonder dat iemand je iets hoeft te geven.

Iets dat je identiteit wil. Een nummer, een document, een controle. Dit is de laag waar een mens onvermijdelijk in beeld komt, en waar automatiseren ophoudt bij de bevestiging.

Iets dat geld raakt. Een rekening, een kaart, een incasso. Streng, traag, en met regels die niet over jou gaan maar over misbruik door anderen.

Iets dat je bestaan wil. Een inschrijving, een rechtsvorm, een aansprakelijkheid. Dit is de laag van het volgende hoofdstuk, en hij is niet met techniek te benaderen.



Vier lagen toegang, van ruim naar streng.

De praktische regel die hieruit volgt: doe zoveel mogelijk op de onderste laag, en raak de bovenste zo laat mogelijk aan. Niet uit angst, maar omdat elke laag hoger je langzamer maakt en afhankelijker van iemand die niet op jou wacht.

Zelf doen · Tel wie er buiten het bedrijf op jou wacht

> **Wat je doet.** Alles wat je tot nu toe hebt gebouwd, kijkt naar binnen. Maar het werk stopt niet bij je eigen map: er zijn partijen die iets van jou nodig hebben, en elk van hen heeft zijn eigen termijn waar jij niets over te zeggen hebt. Laat die op een rij zetten. Reken op vijf minuten.

TYP DIT

Zet op een rij welke partijen buiten dit bedrijf ergens op wachten: wat ze nodig hebben, waar het over gaat, en welke termijn eraan hangt. Kijk in kwaliteitsborging, aanbestedingen, crediteuren en klachten.

Wat je ziet. Vijf partijen, negen dingen. **En daar staat er één tussen waarvan de termijn al verstreken is**: een aanbesteding die op 24 april sloot en waarop nooit een besluit is genomen. Veertigduizend tot zestigduizend per jaar, drie jaar lang, en er is geen nee gezegd — er is niets gezegd. **Dat is het verschil tussen een gemiste kans en een gemiste termijn.** Het eerste is een keuze, het tweede niet.

WAT ER TERUGKWAM

```
kwaliteitsborger  verklaring     2 dossiers
bevoegd gezag     gereedmelding  2 dossiers
aanbestedingen    besluit        NLAB-2026-01
                                 SLUITING VERSTREKEN
                                 NLAB-2026-04 over 19 da-
gen
crediteur         betaling       NLIF-2026-250 geblokkeerd
opdrachtgever     antwoord       2 open klachten
```

Waar het misgaat. Zoek op termijn en niet op status. *Open* zegt niets; *sluit over negentien dagen* wel. De helft van deze lijst stond al ergens als open, en juist daardoor viel er niets op. **Neem ook de partijen mee waar je niets van hoort.** Een aanbestedingsportaal stuurt geen herinnering, en een gemeente ook niet — hun stilte betekent niet dat er niets loopt. **En let op dat de termijn van een ander niet in jouw agenda staat.** Dat is precies waarom deze lijst er niet was.

EN DAN ZO

Zet er per regel bij wat het kost als de termijn verloopt, en sorteer daarop in plaats van op datum.

EN DAN DIT

```
NLAB-2026-01   40-60k per jaar, 3 jaar   VERSTREKEN
NLAB-2026-04   circa 900k                19 dagen
NLKB-2026-01   ingebruikname geblokkeerd  -
NLKB-2026-02   ingebruikname geblokkeerd  -
NLIF-2026-250  2640 euro, relatie         -
klachten     geen bedrag, wel een klant -
de duurste stond onderaan toen er op datum
werd gesorteerd
```

Dat overzicht zegt wie er op jou wacht. Wat het niet zegt, is of je hun kant op ook echt kúnt kijken. Tot hier las alles wat je bouwde bestanden in je eigen map — en het interessante zit meestal net daarbuiten.

Zelf doen · Sluit er één aan, en geef hem een eigenaar

Wat je doet. Tot nu toe las alles wat je bouwde bestanden in je eigen map. **De boekhouding staat niet in je eigen map.** Sluit die koppeling aan — alleen lezen — en maak meteen de werkwijze die hem bijhoudt. Dat tweede is het echte werk: een koppeling zonder eigenaar verloopt. Reken op twintig minuten.

TYP DIT

> Koppel de MCP van onze boekhouding, met alleen leesrechten. Maak daarna een herbruikbare werkwijze die elke maandag per crediteurenfactuur de status ophaalt en alleen het veld status terugzet in crediteuren/. Laat hem melden wat er veranderde, en apart welke facturen de koppeling niet kent.

Wat je ziet. Het aansluiten duurt een minuut en is eenmalig. **De werkwijze eromheen is waar het om gaat.** Hij meldt twee statuswijzigingen — prima — en daarnaast iets waar je niet om vroeg: **één factuur van 12.900 euro kent de boekhouding helemaal niet.** Die staat hier

wel en daar niet. Zonder dat vierde punt had de ronde gezegd dat alles in orde was.

WAT ER TERUGKWAM

```
boekhouding-bijwerken, ronde van vandaag
  veranderd: 2
    NLIF-2026-238  openstaand -> betaald
    NLIF-2026-221  betaald (datum bijgewerkt)
  ongewijzigd: 2
  koppeling kent deze niet: NLIF-2026-255
KOPPELING: boekhouding, alleen lezen, sinds 4 mei
```

Waar het misgaat. Begin met alleen lezen. Een koppeling die mag schrijven kan iets kapotmaken in een systeem dat niet van jou is, en daar is geen ongedaan maken. Schrijven komt pas als je weet wat je terugkrijgt — en meestal blijkt het dan niet nodig. **Vraag om wat de koppeling NIET kent.** Dat is het enige punt dat een stille koppeling ontmaskert: minder teruggeven ziet er precies zo uit als alles in orde. **En geef elke koppeling een eigenaar.** Een sleutel wordt ingetrokken, een veld verandert van naam, een antwoord krijgt er een regel bij. Dat merk je niet bij het aansluiten maar drie maanden later, en dan weet niemand meer hoe het werkte.

EN DAN ZO

Laat de werkwijze er ook bij zetten welke rechten de koppeling heeft en wanneer hij voor het laatst iets terugkreeg, zodat ik zie dat hij nog leeft.

EN DAN DIT

```
KOPPELING boekhouding
  rechten     : alleen lezen
  aangesloten : 2026-05-04
  laatst iets
  teruggekregen: vandaag, 5 facturen
  eigenaar    : boekhouding-bijwerken
een koppeling zonder deze vier regels is een
koppeling waarvan je niet weet of hij nog werkt
```

En dit gaat niet over AI

Iedereen die ooit iets is begonnen, kent dit, en meestal is het de eerste echte tegenslag.

- **Een startende ondernemer** heeft het idee, het product en de klant, en loopt vast op een zakelijke rekening die zes weken duurt.
- **Iemand die net in een land woont** kan niets regelen zonder adres, en krijgt geen adres zonder iets dat hij nog niet heeft.
- **Een kleine leverancier** komt niet binnen bij een grote afnemer omdat hij niet in het systeem staat, en niet omdat zijn product slechter is.
- **Een vereniging zonder rechtsvorm** kan geen contract tekenen, hoe goed georganiseerd hij ook is.

In alle vier is het probleem niet vermogen. Het is dat het vermogen niet naar binnen mag.

En dat is precies waarom toegang in dit boek een eigen hoofdstuk heeft. Wie denkt in techniek, lost het verkeerde op — want daar is het niet schaars.

Probeer: laat alles opsommen waar dit werk toegang voor nodig heeft, in de volgorde waarin ze elkaar mogelijk maken, met per stuk of je het zelf kunt regelen, of iemand anders het uitdeelt, of er een mens aan te pas moet komen. Bijna iedereen heeft die keten wel in zijn hoofd maar nooit op papier, en begint daardoor ergens halverwege — bij een deur waarvan de sleutel er nog niet is. Vraag ook wat er wegvalt als een schakel wegvalt; dat is de enige manier om te zien hoeveel er aan die ene eerste hangt.

Die kaart laat zien waar je kwetsbaar bent. Hierna staat wat je daar vandaag al aan kunt doen.

Tip: vraag wat er kan zónder identiteitscontrole, betaling of inschrijving — de versie die op de ruimste laag blijft — en wat je daarmee inlevert. Elke laag hoger maakt je trager en afhankelijker van iemand die niet op jou wacht, en wat op de onderste laag kan, kan vandaag. Vraag wél wat je inlevert: op de onderste laag blijven is soms de verkeerde keuze, en dat wil je weten voordat je erin zit.

> **Wat dit goedkoper maakte:** *de eerste toegang maakte de tweede goedkoop, en vanaf de derde ging het vanzelf — zolang de eerste bleef bestaan.*

23. Vastzitten is een ontwerpkeuze

Er was een dag waarop een model dat we gebruikten in ons deel van de wereld ineens niet meer gebruikt mocht worden.

Niet omdat het stuk was. Niet omdat het duurder werd. Er was een beslissing genomen ergens anders, en van de ene dag op de andere was dezelfde dienst hier niet meer beschikbaar terwijl hij elders gewoon doordraaide. Er viel niets aan te repareren, want er was niets kapot.

Dat is de kern van dit hoofdstuk, en hij is ongemakkelijk: **de belangrijkste storing die je kunt krijgen, is er een waar je niets aan kunt doen.**

Waarom dit erger is dan vroeger

Vastzitten aan een leverancier is een oud verhaal. Het ongemakkelijke is dat de oude versie milder was.

Vroeger zat je vast in een hoek. Eén systeem voor de boekhouding, één voor de planning. Vervelend om te vervangen, maar de rest bleef overeind terwijl je het deed, en je had maanden om het te doen.

Nu ligt de afhankelijkheid ónder alles. Elke agent, elke controle, elke lus die zonder jou draait, leunt op hetzelfde. Valt dat weg, dan valt niet één hoek weg maar het kunnen werken — en precies de dingen die

je had geregeld zodat je er niet meer naar hoefde te kijken, zijn de dingen die dan stilvallen.

En er zijn drie manieren waarop het wegvalt, die niets met elkaar te maken hebben en daarom niet tegelijk te verzekeren zijn.

De prijs gaat omhoog. Wat je bouwde toen het goedkoop was, draait nu op een rekening die iemand anders bepaalt. En omdat het overal in zit, is er geen deelbesluit te nemen.

Het mag ineens niet meer. Een besluit van een toezichthouder, een verandering in de voorwaarden, een grens die je niet zelf getrokken hebt. Dit is de vervelendste soort, want er is geen technisch probleem dat je kunt oplossen.

Het wordt ingehaald. Dit is de vriendelijkste en misschien de duurste. Wat vandaag het beste is, is over een paar maanden de derde keuze. Als overstappen dan een project is, stap je niet over, en dan betaal je elke dag een beetje voor een keuze die je ooit terecht maakte.



Drie manieren om je aanbieder kwijt te raken, en ze hebben niets met elkaar te maken.

Een plan om over te stappen is geen oplossing

De gewone reactie is een uitwijkplan. Je schrijft op hoe je zou overstappen, wat het zou kosten, en je bergt het op.

Dat helpt minder dan het lijkt, om drie redenen.

Een plan veroudert stil. Het beschrijft een overstap tussen twee dingen die allebei veranderen, en niemand houdt het bij, want er is nooit een dag waarop dat urgent is — tot de dag waarop het te laat is.

Een plan is niet beproefd. Een overstap die nooit gedaan is, is een gok met een netjes opgemaakte omslag. Wat er werkelijk aan vastzit, blijkt pas tijdens het losmaken.

En een plan kost tijd op precies het verkeerde moment. De dag dat het nodig is, is de dag dat er al iets anders aan de hand is. **Dus is de vraag niet "hoe stap ik over" maar "hoe zorg ik dat overstappen geen gebeurtenis is".**

Wat je in plaats daarvan doet

De zet is niet voorbereiden maar ontwerpen. Vier dingen, van goedkoop naar duur.

Eén: zet er een tussenlaag in. Laat je werk niet rechtstreeks met de aanbieder praten, maar met een eigen tussenlaag die één taal spreekt. Die laag is klein en saai, en hij is het enige dat je moet aanpassen als er iets verandert. Zonder die laag zit de aanbieder in elke hoek van je werk; met die laag zit hij op één plek.

Twee: draai er nu al meer dan één. Niet als reserve maar als gewone gang van zaken. Een tweede aanbieder die alleen in een plan bestaat, werkt niet; een tweede die vandaag een deel van het werk doet, werkt aantoonbaar. Dit is het verschil tussen een uitwijk en redundantie, en het is het hele hoofdstuk in één zin: **je wilt niet kunnen overstappen, je wilt al overgestapt zijn.**

Drie: laat het werk zichzelf routeren. Als er meer dan één is, hoeft de keuze niet van jou te komen. Stuur het werk naar wat op dit moment beschikbaar is, betaalbaar is, en mag. Die drie veranderen los van elkaar en geen ervan verandert op jouw verzoek — dus hoort het besluit bij de lus en niet bij jou.

Vier: laat het zichzelf herstellen. Valt er een weg, dan schuift het werk op. Dat vraagt precies wat het vliegwielhoofdstuk vroeg: dat je goedkoop kunt zien of het goed ging. Zonder die controle is automatisch uitwijken alleen een snellere manier om stil iets verkeerd te doen.



beschikbaar, betaalbaar, toegestaan

Niet één aanbieder met een uitwijkplan, maar werk dat zichzelf verdeelt over wat beschikbaar, betaalbaar en toegestaan is.

Wat het kost, want het is niet gratis

Dit hoofdstuk zou oneerlijk zijn zonder de prijs erbij.

Je kunt niet overal het beste van één aanbieder gebruiken. Wat maar op één plek bestaat, kun je niet overal inzetten. Dat is een echte beperking en je voelt hem meteen.

De tussenlaag is extra werk dat op de dag zelf niets oplevert — precies het profiel van een multiplier uit deel I, met hetzelfde ongemak dat het op uitstel lijkt.

En twee dingen draaiend houden is meer dan twee keer één ding. Twee keer configuratie, twee keer een rekening, twee keer iets dat kan veranderen.

De verstandige middenweg is niet "alles overal hetzelfde". Het is: **een pad dat overal werkt, en daarnaast een pad dat het beste gebruikt van waar dat bestaat.** Het eerste is je bodem, het tweede is winst. Wat je nooit doet, is het tweede laten dragen wat het eerste hoort te dragen.

Zelf doen · Vraag wat er zou gebeuren als je morgen moest verhuizen

Wat je doet. Deze hele administratie is platte tekst in mappen. Dat was geen toeval, en dit hoofdstuk gaat over de vraag wat dat je oplevert. Stel hem hardop. Reken op vijf minuten.

TYP DIT

> Als dit hele bedrijf morgen naar een ander pakket zou moeten verhuizen, wat zou dan het lastigst zijn? Kijk daarbij niet alleen naar de bestanden maar ook naar wat er nergens staat.

Wat je ziet. De bestanden zelf zijn het probleem niet — die zijn te lezen zonder programma en te verhuizen met een kopieeropdracht. **Wat vastzit is iets anders: de afspraken die nergens staan.** Dat een opdracht pas af is als er uren én materiaal op staan, staat in `regels.md`. Dat je bij een opdrachtgever twee weken van tevoren belt, staat nergens. Dat de zaterdagtoeslag niet geldt bij nazorg, ook niet. **Dat is wat je kwijtraakt bij een verhuizing, en dat is ook wat je kwijtraakt als iemand weggaat.**

WAT ER TERUGKWAM

```
makkelijk mee te nemen:
  49 mappen, 254 bestanden, platte tekst
moeilijk:
  de werkwijzen -- die verwijzen naar mapnamen
lastigst, want nergens vastgelegd:
  wanneer je welke opdrachtgever belt
  welke toeslag wanneer geldt
  wie je bij welke onderaannemer moet hebben
```

Waar het misgaat. Vraag expliciet naar wat er NERGENS staat. Zonder die toevoeging krijg je een keurig antwoord over bestandsformaten, en dat is precies het deel dat geen probleem is. **En pas op met de conclusie dat platte tekst je vrij maakt.** Dat maakt de gegevens vrij; de afspraken zitten in hoofden en die verhuizen niet mee. **Reken erop dat de lijst met ongeschreven dingen langer is dan je denkt** — hier kwamen er drie uit in vijf minuten, en dat is de ondergrens.

EN DAN ZO

> Zoek nu in dagrapporten, klachten en notulen naar afspraken die wel worden toegepast maar nergens als regel staan, en zet ze op een lijst.

EN DAN DIT

```
toegepast maar niet opgeschreven:
  "bewoners vooraf inlichten voor een opname"
     (dagrapport 2026-04-22)
  "meerwerk aankondigen voor je het doet"
     (dagrapport 2026-04-24)
  "lekkage doorzetten naar de onderaannemer"
     (dagrapport 2026-04-17)
drie regels die iedereen kent en niemand opschreef
```

Dat is de redenering. Het aardige aan dit onderwerp is dat je hem kunt uitvoeren in plaats van geloven, en dat het antwoord in twintig minuten op tafel ligt.

Zelf doen · Verhuis alles echt, en laat het elders nakijken

Wat je doet. Het blok hiervoor vroeg wat er zou gebeuren als je moest verhuizen. **Nu doe je het, en naar een ander merk.** Je laat alles inpakken wat je zelf hebt bedacht, je leest het ergens anders in, en je laat het daar nakijken tegen je eigen logboek. Reken op twintig minuten.

TYP DIT

> Pak alle werkwijzen, de regels, de doelen en grenzen, de beschrijvingen van de koppelingen en het logboek in als één zipbestand, met een leesmij erbij die uitlegt hoe een ander platform dit moet inlezen. Neem geen sleutels en geen bedrijfsgegevens mee. Lees het daarna in op een ander platform en laat dáár melden: hoeveel werkwijzen er zijn, of alle verwijzingen kloppen, en wat er uit het logboek hier wel en niet te reproduceren is. Wijzig niets.

Wat je ziet. De eerste keer vond het ontvangende platform vier fouten die ik zelf niet had gezien. Een map zonder inhoud, vier werkwijzen die in het overzicht stonden zonder bestand, één bestand dat niet in het overzicht stond, en verwijzingen naar mappen die niet in het pakket zaten. Geen ervan was zichtbaar zolang alles op zijn eigen plek stond. **Verhuizen is dus niet alleen een test van de verhuizing — het is een controle op wat je hebt.**

WAT ER TERUGKWAM

```
eerste poging, gemeld door het andere platform:
  een van de mapjes is leeg
  vloot.md noemt 4 werkwijzen zonder bestand
  boekhouding-bijwerken staat niet in vloot.md
  verwijzingen naar ontwikkeling/ en directie/
    zitten niet in het pakket
```

Waar het misgaat. Neem geen sleutels mee. Die horen bij het platform, en een zipbestand met een sleutel erin is precies één keer te maken.

Laat proefdraaien en niet uitvoeren, anders verandert er van alles en kun je niets meer vergelijken. **En vraag om wat er níet reproduceerbaar is.** Dat is het enige eerlijke antwoord, en het andere platform gaf het uit zichzelf: de werkwijzen kwamen compleet mee, de uitkomsten niet — want de gegevens verhuizen apart.

EN DAN ZO

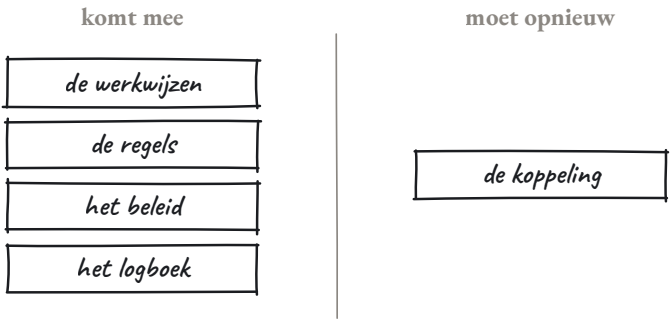
> Repareer de vier bevindingen en draai het opnieuw, en laat er nu bij zetten wat er per werkwijze uit het logboek wel en niet na te rekenen is.

EN DAN DIT

```
tweede poging: 27 werkwijzen, alle verwijzingen
kloppen, geen verschil tussen vloot.md en de
bestanden

reproduceerbaar uit het logboek: 0 van de 8
  "de werkwijzen zijn compleet inleesbaar, de
   uitkomsten niet -- de aantallen zijn metingen
   over mappen die hier niet staan"
```

En dit is nog maar het handmatige begin. Een verhuizing bedenken, hem oefenen op een kopie, op twee plekken tegelijk draaien om te zien of het uitmaakt, bij een storing zelf uitwijken, of de boel splitsen omdat het bedrijf splitst: er is niets aan dat rijtje dat een mens moet doen. Wat een mens moet doen is beslissen dát het gebeurt, en tekenen waar het onomkeerbaar wordt.



Wat meekomt is alles wat niet naar buiten reikt; de koppeling is het enige dat opnieuw moet.

En dit gaat niet over AI

Iedereen die ooit ondernomen heeft, kent dit patroon, meestal uit eigen ervaring.

- **Eén grote klant die zeventig procent van je omzet is.** Iedereen weet dat het riskant is, en toch komt het overal voor, omdat hij op de dag zelf de goedkoopste klant is om te bedienen.
- **Eén leverancier voor een onderdeel dat overal in zit.** Zolang het loopt is hij de zuinigste keuze. Op de dag dat hij niet levert, ligt de hele productie stil en niet één product.
- **Eén persoon die als enige weet hoe iets werkt.** Dat is dezelfde afhankelijkheid, alleen kun je die niet opzeggen.
- **Eén bank, één verzekeraar, één platform waar je klanten vandaan komen.** Geen daarvan voelt als een risico, want het gaat goed. Dat is precies wat een risico is voordat het er een is.

In alle vier de gevallen is de vraag dezelfde, en het is de enige vraag die dit hoofdstuk stelt: wat gebeurt er morgen als dit er niet meer is, en hoe lang duurt het voordat ik weer werk?

Als het antwoord "dat weet ik niet" is, dan is dat het antwoord.

Probeer: zet één keer per kwartaal een halve dag je gewone aanbieder uit en laat het werk op de tweede draaien. Noteer alles wat niet werkte, ook de kleine dingen, en waar het op vastliep. Een overstap die je nooit gedaan hebt is een gok, en wat er werkelijk aan vastzit blijkt pas tijdens het losmaken — nooit uit het plan. Doe het op een rustige dag: een uitwijk die je voor het eerst uitprobeert op de dag dat het nodig is, is geen uitwijk maar een tweede probleem.

Die oefening laat zien waar je vastzit. Wat je daarna bouwt is de plek waar dat vastzitten samenkomt, zodat het niet langer overal zit.

En leg de tussenlaag aan voordat je hem nodig hebt: zoek elke plek waar je rechtstreeks met één aanbieder praat, en vraag hoe daar één eigen laag tussen komt — met erbij welke mogelijkheden *alleen* bij deze aanbieder bestaan. Die laatste vraag is de belangrijkste: wat maar op één plek bestaat, is waar je straks aan vastzit. Je hoeft het niet op te geven, je moet weten dat het er is. Houd die laag saai — een tussenlaag die zelf slim wordt is een derde afhankelijkheid, en die is van jou.

Met de tussenlaag op zijn plek wordt de vraag wélke aanbieder je kiest. En dat is een vraag met drie antwoorden die onafhankelijk van elkaar bewegen.

Tip: noteer per aanbieder wat het per maand kost, waar het gebruikt mag worden, en wanneer je voor het laatst gekeken hebt of er iets beters is — plus wat er gebeurt als hij morgen wegvalt. Prijs, toestemming en

tempo veranderen los van elkaar; wie alleen op de rekening let, mist de andere twee, en juist die komen zonder aankondiging. *Wanneer heb ik voor het laatst gekeken* is de vraag die niemand stelt: ingehaald worden voelt nergens als een storing, en daarom wordt het nooit opgelost.

> **Wat dit goedkoper maakte:** *een aanbieder die wegviel werd een gebeurtenis van een middag in plaats van een project — en dat is het verschil tussen doorwerken en stilstaan.*

24. Meedoen in de formele wereld

Er is een grens waar dit boek stopt met over techniek gaan, en die grens is niet technisch.

Je kunt iets bouwen dat werkt, dat zichzelf controleert, dat doordraait als jij slaapt en dat zijn eigen verbeteringen doorvoert. En dan wil een klant een contract, een verzekeraar een polis, een accountant een spoor, en een toezichthouder een naam.

Dat is geen obstakel dat je omzeilt. Het is het laatste stuk van de weg, en het heeft zijn eigen regels — die niet over jou gaan maar over iedereen die er ooit misbruik van maakte.

Wat de formele wereld eigenlijk vraagt

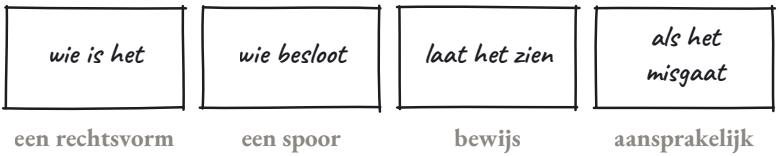
Het lijkt veel en het zijn er vier.

Wie is het? Er moet iets zijn dat bestaat, dat een naam heeft en dat aangesproken kan worden. Een persoon of een rechtsvorm. Niet iets dat namens iemand handelt — iets dat het ís.

Wie heeft besloten? Bij elke beslissing die telt, moet naderhand te zien zijn wie hem nam en op grond waarvan. Niet omdat iemand het wantrouwt, maar omdat een beslissing zonder herkomst niet te herstellen is als hij fout blijkt.

Kun je het laten zien? Niet: was het goed. Kun je *aantonen* dat het ging zoals je zegt. Dat is een ander soort eis en hij is veel makkelijker te halen als je het vooraf weet.

Wat als het misgaat? Wie draait ervoor op, en hoe wordt het teruggedraaid. Dit is de vraag die alle andere drie samenvat, en het is degene waar een verzekeraar op let.



Vier vragen, en geen ervan gaat over techniek.

Het goede nieuws, en het is groter dan je denkt

Vrijwel alles wat dit boek tot nu toe beschreef, is precies wat die vier vragen willen.

Een ticket met een besluit en een reden erin, is het antwoord op *wie heeft besloten*. Een logboek dat tijdens het werk geschreven wordt, is het antwoord op *kun je het laten zien*. Een controle die meedraait en zijn uitkomst bewaart, is bewijs dat niemand achteraf hoefde te maken. Een grens die in de uitvoering zit, is het antwoord op *wat als het misgaat* — want hij laat zien dat het niet kón.

Dat is geen toeval. Het zijn dezelfde eisen, alleen met een andere aanleiding. Een audit vraagt wat een goede werkwijze toch al oplevert; het

enige verschil is dat de audit het achteraf vraagt en jij het tijdens het werk maakt.

En daar zit de winst. Wie deze dingen pas aanlegt als er een norm langskomt, bouwt een tweede administratie naast het werk. Wie ze tijdens het werk maakt, heeft ze al — en de audit wordt een uitdraai in plaats van een project.

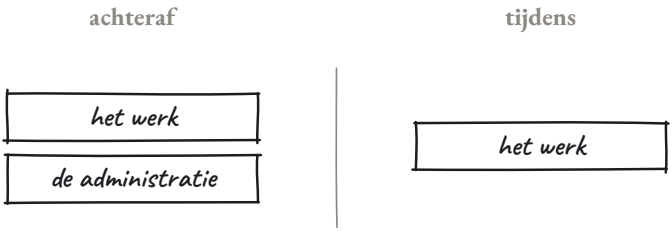
Wat je er wél bij moet doen

Drie dingen, en ze zijn geen van drieën groot.

Zet er een datum en een naam bij. Een besluit zonder moment en zonder iemand is geen besluit maar een notitie. Dit kost één regel en het is precies de regel die je achteraf niet meer kunt maken.

Bewaar wat je gemeten hebt, niet alleen de uitkomst. "De test was groen" is geen bewijs. "Deze test, op deze versie, op dit moment, met deze uitkomst" is dat wel, en het verschil is één regel opslaan in plaats van weggooien.

Maak duidelijk wat een agent deed en wat een mens deed. Dit is nieuw en het gaat een eis worden. Niet omdat er iets mis is met een agent, maar omdat de vraag *wie besloot* een antwoord moet hebben dat geen uitleg nodig heeft.



Wat een audit vraagt, levert een goede werkwijze al op — als je het tijdens het werk maakt.

Dit is de wereld waarin ik het langst gewerkt heb

Van alle hoofdstukken in dit boek is dit het hoofdstuk waarvan ik het meeste uit een ander vak haal, en het is ook het hoofdstuk waar de meeste mensen te laat aan denken.

Het begeleiden van mensen naar werk is vrijwel volledig een formele wereld. Er is bijna niets dat je alleen maar hoeft te *kunnen*; het moet ook **passen** — in een regeling, in een contract, in een dossier, met iemand die tekent. Ik heb daar jarenlang gezien wat dat met goede plannen doet.

Twee dingen heb ik daar geleerd die hier één op één gelden.

Het eerste: de formele eis is bijna nooit het probleem. Het achteraf aantonen is het probleem. Wat er gevraagd wordt, is meestal redelijk en meestal ook gewoon gebeurd. Wat ontbreekt, is dat het ergens staat, op het moment dat het gebeurde, in een vorm die iemand anders kan lezen. Wie dat achteraf moet reconstrueren, betaalt drie keer: het werk zelf, het terugzoeken, en het ongemak dat je iets niet hard kunt maken waarvan je zeker weet dat het klopte.

Dat is precies hetzelfde punt als in hoofdstuk 11, alleen met een andere afnemer. **Het vastleggen waarvan je leert, is het vastleggen dat een controleur wil zien.** Je doet het dus niet twee keer. Je doet het één keer, goed, en het bedient allebei.

Het tweede: de handtekening is geen formaliteit, hij is een positie. Er is altijd iemand die zegt: hier sta ik voor in. In dat vak was dat zichtbaar en soms ongemakkelijk — want tekenen betekende dat je het risico droeg van iets dat je niet in de hand had. Maar precies dat maakte het mogelijk dat iemand ergens mocht beginnen.

Dat is de rol die je zelf hebt zodra je met agenten werkt. Niet de bouwer, niet de gebruiker: degene die tekent. Alles wat een agent doet, gebeurt onder jouw naam, en de vraag *durf ik hiervoor te tekenen* is een betere toets dan elke technische controle — want hij gaat over de uitkomst en niet over het mechanisme.

En er zit een geruststelling in. Deze wereld is traag en streng, maar hij is niet grillig. De eisen staan vast, ze zijn vooraf te lezen, en ze veranderen langzaam. Dat maakt hem het tegenovergestelde van een frats: dit is een obstakel dat je kúnt kennen. Wie het vooraf opzoekt, komt er nooit meer door verrast — en dat is meer dan je van de meeste dingen in dit boek kunt zeggen.

Wat niet kan, en dat blijft zo

Eerlijk zijn hoort hier, want dit is het hoofdstuk waar mensen te hard doorlopen.

Een agent tekent niet. Zie het hoofdstuk over grenzen: er is altijd iets dat het ís. Dat gaat niet veranderen doordat de techniek beter wordt.

Een agent is niet aansprakelijk. Als er schade is, ligt die bij de partij die de opdracht gaf of het product leverde. Dat is precies zoals het bij gereedschap altijd is geweest, en het is de reden dat "de AI deed het" nergens een antwoord is.

En sommige beslissingen mogen niet automatisch. Niet omdat ze te moeilijk zijn, maar omdat er is afgesproken dat er een mens naar kijkt. Dat verschilt per land en per vak en het verandert, maar het verdwijnt niet.

De praktische houding die daarbij hoort is niet die grenzen opzoeken maar ze opschrijven. Een lijst van wat er langs een mens moet, is korter dan je denkt — en zodra hij er is, is al het andere vrij.

Zelf doen · Zet een gate voor iets dat je niet mag doen

Wat je doet. Bij een vergunningplichtig bouwwerk mag je pas in gebruik nemen na een gereedmelding met een compleet dossier. Dat is geen huisregel maar een wettelijke volgorde, en hij is onomkeerbaar: wat dichtgemaakt is zonder foto, is alleen nog met slopen aan te tonen. Bouw een gate die dat tegenhoudt. Reken op tien minuten.

TYP DIT

> Maak een gate die per dossier in kwaliteitsborging/ de zes verplichte onderdelen nagaat. Hij mag drie uitkomsten geven: DOOR als het compleet is, NOG NIET GEREED als er iets mist maar er nog geen gereedmelding is, en TEGENGEHOUDEN als er wél is gereedgemeld terwijl er iets ontbreekt. Draai hem.

Wat je ziet. Twee dossiers missen nog de verklaring van de borger en zijn nog niet gereedgemeld — dat is gewoon werk in uitvoering. **Het derde is gereedgemeld op 22 april zonder de foto's van het verbor-**

gen werk. Dat is niet meer te repareren met een foto: het werk zit dicht. **Hier zie je waarom een gate vóór de handeling moet staan en niet erna.** Deze gate meldt het nu; drie weken eerder had hij het voorkomen.

WAT ER TERUGKWAM

```
NOG NIET GEREED  NLKB-2026-01  mist: verklaring borger
NOG NIET GEREED  NLKB-2026-02  mist: verklaring borger
TEGENGEHOUDEN    NLKB-2026-03 is gereedgemeld op
                 2026-04-22 zonder:
                 foto's verborgen werk
exitcode: 1
```

Waar het misgaat. Onderscheid "nog niet klaar" van "te laat". Zonder dat onderscheid staat een dossier dat netjes in uitvoering is op dezelfde lijst als een overtreding, en dan wordt de lijst genegeerd. **Zet de gate vóór de gereedmelding en niet erna.** Deze versie kijkt achteraf omdat het al gebeurd is; de bruikbare plek is het moment waarop iemand op *gereedmelden* drukt. **En laat hem niet oplossen wat hij tegenhoudt.** Een gate die aanbiedt het dossier maar even aan te vullen, vult het aan met wat er niet is.

EN DAN ZO

Laat de gate er per tegengehouden dossier bij zetten wat er nog wél kan, en wat definitief niet meer kan.

EN DAN DIT

```
NLKB-2026-03, gereedgemeld zonder foto's
  nog mogelijk:
    verklaring van de borger op basis van zijn
    eigen aantekeningen bij de wapening
  definitief niet meer:
    foto's van het verborgen werk -- het werk
    zit dicht
  gevolg: melden, niet stilhouden
```

En dan is er nog iemand die meekijkt zonder iets op te lossen. Dat klinkt als een last en het is het tegenovergestelde: hij ziet wat er door een verandering niet meer vanzelf klopt, en dat is precies wat je zelf niet ziet omdat je met de verandering bezig was.

Zelf doen · Laat iemand meekijken die niets mag oplossen

Wat je doet. Je hebt in hoofdstuk 23 alles naar een ander platform verhuisd. Dat werkte. **De vraag van dit hoofdstuk is wat daardoor níet meer vanzelf klopt** — en dat is een andere vraag dan of het draait. Laat er een werkwijze naar kijken die uitsluitend signaleert en niets repareert. Reken op tien minuten.

TYP DIT

> Er is iets veranderd: de werkwijzen draaien sinds vandaag op een ander platform. Kijk wat daardoor opnieuw geregeld, opnieuw aangetoond of opnieuw gedraaid moet worden. Zeg per punt waarom, en wat de eerstvolgende handeling is en van wie. Los niets op en houd niets tegen.

Wat je ziet. Zes signalen, netjes in drie soorten. Twee gaan over papier dat er moet komen, twee over bewijs dat niet meekwam, twee over metingen die opnieuw moeten. **En er zit er één tussen waar ik zelf overheen had gelezen:** een gate is een eigenschap van de omgeving en niet van de tekst. Het woord STOP in een bestand houdt niets tegen — dat het op het oude platform werkte, zegt niets over het nieuwe. **Dat er persoonsgegevens meeverhuizen naar een plek waarvan niemand heeft vastgelegd waar hij staat, is het tweede dat je niet zelf had bedacht.**

WAT ER TERUGKWAM

```
OPNIEUW REGELEN
  verwerkersovereenkomst, en waar de data staat
OPNIEUW AANTONEN
  de leesrechten, en dat de gates tegenhouden
OPNIEUW DRAAIEN
  de twaalf regels, en de nulmeting per gate
```

Waar het misgaat. Laat hem niets oplossen. Een auditor die zijn eigen bevindingen afvinkt is geen auditor — en de verleiding is groot, want vier van de zes lijken met één handeling weg te werken. **Vraag om de eerstvolgende handeling en van wie.** Zonder naam is een signaal een mening; met naam is het een opdracht. **En laat hem nooit "dit mag niet" zeggen.** Dat is een oordeel dat rust op kennis die hier niet staat. Hij zegt wat er ligt en wat dat raakt; iemand anders beslist.

EN DAN ZO

Zet er per signaal bij wat er gebeurt als je het niet doet, zodat ik kan kiezen wat eerst moet.

EN DAN DIT

```
verwerkersovereenkomst niet doen
  -> geen grondslag, niet achteraf te repareren
gates niet aantonen
  -> je denkt dat er iets tegenhoudt terwijl
     er niets tegenhoudt
regels niet opnieuw draaien
  -> je vergelijkt met een meting van een
     ander systeem
```

En dit gaat niet over AI

- **Een aannemer met een kwaliteitssysteem** levert niet beter werk omdat het systeem er is, maar hij kan het aantonen — en dat is wat hem toelaat bij opdrachtgevers die het vragen.
- **Een praktijk met een dossierplicht** schrijft niet voor de inspectie maar voor de volgende behandelaar. Dat het ook voor de inspectie werkt, is meegenomen.
- **Een producent met traceerbaarheid** kan bij een probleem één partij terughalen in plaats van alles. Dat is geen last maar het verschil tussen een incident en een faillissement.
- **Een vereniging met statuten** kan een rekening openen en een contract tekenen. Zonder is het geen vereniging maar een groep mensen.

In alle vier is de formele kant niet iets dat erbij komt. Het is wat ze toegang geeft tot werk dat er anders niet was.

Probeer: loop de vier vragen af voor wat je nu draaiend hebt — wie is het, wie heeft besloten, kun je het laten zien, en wat als het misgaat — met per vraag of je het antwoord hebt, half hebt of niet hebt, én waar dat antwoord dan staat. Meestal heb je er drie al zonder het te weten, omdat een goede werkwijze ze toch al oplevert; welke er ontbreekt is niet te raden en blokkeert wel alles. Vraag wáár het antwoord staat: *"ja dat hebben we"* zonder plek is hetzelfde als nee.

De vier vragen zeggen wat er ontbreekt. Hierna staat de gewoonte die het gat vanzelf dicht.

Zet dit in je vaste instructies: *bij elk besluit dat gevolgen heeft: schrijf op wat er besloten is, wanneer, op grond waarvan, en of het door mij of*

door jou genomen is — één regel, in het ticket, niet in een apart bestand. Een spoor dat je achteraf maakt is een reconstructie en kost tien keer zoveel, en het onderscheid mens-of-agent is de enige echt nieuwe eis in dit hoofdstuk. Houd het op één regel en in het ticket: een apart logboek voor de formele wereld wordt een tweede administratie, en die raakt achter.

> **Wat dit goedkoper maakte:** *wat een audit vraagt, lag er al — en daarmee werd meedoen in de formele wereld een uitdraai in plaats van een project.*

25. Als iedereen het kan

Ik heb een tijd voor een bank gewerkt. Dat is een wereld met lagen: iemand bedenkt een project, een ontwikkelteam bouwt het, een testteam controleert het, en daar zit een keten van afspraken omheen die ouder is dan iedereen die eraan meedoet.

Stel je dan het volgende voor. Niet als voorspelling, en niet als iets dat goed of fout is — als een doordenker.

> *De manager bedenkt zijn projecten met AI. Het ontwikkelteam bouwt ze met AI. Het testteam automatiseert zijn testautomatisering met AI.*

Elke laag is sneller geworden. Niemand heeft iets verkeerd gedaan. En dan is de vraag niet of dit gebeurt, maar wat voor **vragen en behoeftes** er ontstaan zodra het zover is. Want dat is waar je achter komt waar je agenten werkelijk voor nodig hebt: niet door te kijken wat ze kunnen maken, maar door te kijken wat er overblijft als maken gratis is.

De eerste uitkomst is niet minder werk maar meer

Alles wat **voortbrengt** is versneld. De manager levert tien doordachte voorstellen per week in plaats van één. Het ontwikkelteam kan ze bouwen. Het testteam kan ze testen.

Maar het **kiezen** is niet versneld. Het willen ook niet. Er is nog steeds één begroting, één klantengroep en één toezichthouder.

Dus loopt de wachtrij vol met werk dat er allemaal redelijk uitziet, en er is geen manier om het te rangschikken. **Het knelpunt is niet verdwenen, het is verhuisd** — van maken naar kiezen. En kiezen is waar niemand op was ingericht, want dat was nooit het dure deel.

De tweede: "kan het" verdwijnt als vraag

In de oude opstelling was *kan het gebouwd worden* de zeef. Die vraag stuurde de hele trechter: wat er niet doorheen kwam, hoefde niet besproken te worden.

Die zeef is weg. Wat overblijft is *moet het* — en dat is een vraag over waarde, over risico, en over wat een bank verplicht is. Geen van die drie wordt sneller van een model.

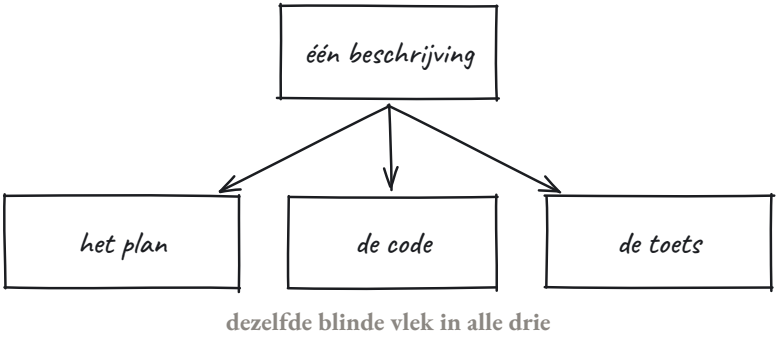
Wie gewend was zijn prioriteiten door capaciteit te laten bepalen, staat er dan zonder prioriteiten. Dat voelt als een luxeprobleem tot je het meemaakt.

De derde, en dit is de belangrijkste

Nu de laag eronder. Drie afdelingen, alle drie met AI, en alle drie beginnen ze bij dezelfde beschrijving.

Het plan komt uit die beschrijving. De code komt uit dat plan. De toets komt uit diezelfde beschrijving. En dus dragen alle drie **dezelfde blinde vlek**: wat er niet in stond, staat in geen van de drie.

De toets bewijst dan niet dat de code klopt. De toets bewijst dat de code het eens is met hetzelfde misverstand.



Eén bron, drie uitkomsten, en wat er niet in de bron zat zit in geen van de drie.

> **Een toets die zijn bron deelt met wat hij toetst, toetst niets.**

Dit is de frats uit hoofdstuk 12, maar dan op de schaal van een organisatie. Alles staat op groen en er is niets gecontroleerd. Niemand heeft iets fout gedaan, er is geen melding, en het valt pas op als er iets in de echte wereld niet klopt.

En hier is het mooie: een bank heeft hier al een woord voor. Vier ogen. Functiescheiding. Die regels bestaan niet uit wantrouwen maar omdat **onafhankelijkheid is wat een controle tot een controle maakt.** Ze zijn ouder dan AI, ze zijn nooit afgeschaft, en ze zijn opeens weer urgent — want functiescheiding tussen mensen die alle drie hetzelfde model dezelfde beschrijving voeren, is functiescheiding op papier.

En er is een scherpere variant, die je niet aan de bron ziet

De regel hierboven gaat over een toets die zijn bron deelt met wat hij toetst. Er is een vorm die een stap verder gaat en die je met geen enkele bronvergelijking vindt:

> **Het bewijs dat iets mocht blijven, was door dat iets zelf geproduceerd.**

Een voorbeeld uit het maken van dit boek. Achterin staat een termenlijst, en de regel is dat er alleen woorden in mogen die ook in de tekst voorkomen — met een controle die dat nakijkt. Van de eenenzeventig termen vielen er drieënveertig af.

Twee overleefden die dat niet verdienden. De controle zoekt op de eerste letters van een term, en het woord dat hij zocht kwam toevallig voor in een heel ander woord. Genoeg om de toets groen te houden, en dus bleven ze staan.

De term stond in de lijst. Omdat de toets een treffer vond, mocht hij blijven staan. En omdat hij bleef staan, werd de toets nooit strenger gemaakt. **Twee stappen, allebei op zichzelf redelijk, en samen een gesloten kring.**

De vraag die hem vangt is er één: als ik dit weghaal, verdwijnt dan ook het bewijs dat het er moest zijn? Is het antwoord ja, dan was het geen bewijs.

En dit zit overal waar iets zichzelf in stand houdt: een regel waarvan het enige bewijs het stuk is dat hem beschrijft, een kengetal dat gemeten wordt uit het systeem dat het moet verbeteren, een onderdeel dat alleen nog wordt gebruikt door de controle die het nakijkt.

En toen was er tijd over

Dit is de uitkomst die ik het slechtst had voorspeld.

Het bedrijf uit de voorbeelden loopt inmiddels met achttien dingen die zichzelf aanroepen. Wat dat opleverde was niet minder werk — er is niemand weggegaan en er is niemand minder gaan werken. Wat het opleverde was **uren van mensen die kunnen beoordelen**, en dat is precies het deel dat schaars was.

En dan komt de vraag waar dit hoofdstuk over gaat: waar doe je die uren heen?

Het antwoord was niet meer opdrachten aannemen. Dat kon ook, en het is het antwoord dat iedereen als eerste geeft. Het antwoord werd: **iets doen wat dit bedrijf nooit heeft gedaan, namelijk ontwikkelen voor eigen rekening en risico.** Zelf een kavel zoeken, er zelf iets op bedenken, het zelf verkopen.

Er draaien daar nu drie dingen voor. **kavels-opsporen** kijkt wekelijks wat er in het werkgebied te koop komt en legt per vondst vast wat het bestemmingsplan toelaat. **locatie-beoordelen** zet daarnaast wat er fysiek kan en wat de kortste weg naar een vergunning is. **conceptplan** maakt twee of drie varianten met een indicatie van kosten, opbrengst en doorlooptijd — altijd meer dan één, want één variant is een voorstel en twee varianten zijn een keuze.

Van de vier kavels die er zo uit kwamen viel er één af op de bestemming, staat er één te wachten op de gemeente, is er één niet beoordeeld, en is er één kansrijk. **Dat is werk dat er anders niet was geweest**, gedaan door dezelfde vijf mensen.

En dan houdt het op, om twee redenen die allebei buiten de machine liggen.

De eerste is **kapitaal**. Een kavel koop je vooruit. Geen enkele agent brengt geld in, geen enkele agent tekent een financieringsovereenkomst, en geen enkele agent is aansprakelijk als het misgaat. Wat er ligt is uitgezocht — welke vormen er zijn, wat ze vragen, wat ze kosten — en verder niets.

De tweede is **kennis**. Wat een goede locatie is, wat een wijk aankan, wat een bestemmingsplan in de praktijk toelaat: dat staat in geen enkel bestand van dit bedrijf. Het moet erin, en dat kost maanden — niet omdat het ingewikkeld is, maar omdat het uit hoofden komt en iemand het moet opschrijven.

Dat is de eerlijke stand van zaken, en hij is beter nieuws dan hij klinkt. De twee dingen die ontbreken zijn allebei te benoemen. Dat betekent dat al het andere er wel is.

Dat is waar de uren heen gingen. Hoe dat er in de praktijk uitziet, is met drie werkwijzen na te spelen.

Zelf doen · Zet de vrijgekomen uren op iets dat er nog niet was

> **Wat je doet.** Het bedrijf uit de voorbeelden heeft geen mensen minder gekregen maar wel uren over — de uren van de mensen die kúnnen beoordelen. **Die gaan nu naar iets dat dit bedrijf nooit heeft gedaan: zelf ontwikkelen.** Drie werkwijzen achter elkaar: zoeken, beoordelen, en twee varianten uitwerken. Reken op vijftien minuten.

TYP DIT

> Kijk in ontwikkeling/ naar de kavels. Beoordeel er per stuk of er iets op kan volgens de bestemming en wat er in de weg zit. Werk voor de kansrijke twee varianten uit met een indicatie van bouwkosten, opbrengst, doorlooptijd en het grootste risico — altijd meer dan één. Zeg tot slot wat er nodig is om verder te komen en wat daarvan er niet is.

Wat je ziet. Vier kavels: één valt af op de bestemming, één wacht op de gemeente, één is nog niet bekeken, en één is kansrijk. **Voor die ene komen er twee varianten uit, en de kleinste blijkt de betere.** Zes woningen in plaats van acht geeft een hogere marge, een kortere doorlooptijd, en minder afhankelijkheid van een ontsluiting die de gemeente moet goedkeuren. Dat is niet wat je verwacht, en het is precies waarom je er twee maakt.

WAT ER TERUGKWAM

```
NLK-2026-001  bedrijfsbestemming   afgevallen
NLK-2026-002  wonen, max 8         KANSRIJK
NLK-2026-003  wijziging loopt      afwachten
NLK-2026-004  gemengd              nog niet bekeken

CP-01  8 rijwoningen  marge 28,1%  22 maanden
CP-02  6 woningen     marge 30,7%  19 maanden
```

Waar het misgaat. Vraag om meer dan één variant. Eén variant is een voorstel en daar zeg je ja of nee op; twee varianten zijn een keuze en daar denk je bij na. **Vraag naar wat er in de weg zit en niet naar of het een goed idee is.** Of dit een goede locatie is, hangt af van marktkennis die nergens in deze map staat — en een antwoord op die vraag klinkt net zo overtuigend als het antwoord op de eerste. **En laat elke variant dragen dat hij niet is doorgerekend.** Dit zijn indicaties waar straks kapitaal aan hangt; een getal zonder die aantekening wordt over een maand een begroting.

EN DAN ZO

Zeg er tot slot bij wat er nodig is om verder te komen, en wat daarvan er wel en niet is.

EN DAN DIT

```
wat er is:
  uren van mensen die kunnen beoordelen
  vier kavels, twee uitgewerkte varianten
wat er NIET is:
  kapitaal -- eigen middelen dekken de voor-
    bereiding, niet een kavel
  kennis van de markt -- staat in geen bestand
  geen van beide lost een agent op
```



en dan houdt het op: geen kapitaal, geen marktkennis

Van vier kavels naar twee varianten — en daar houdt het op.

Maar dit was nog steeds jij die typt

De opdracht hierboven is een goede opdracht en hij levert echt iets op. Toch is er iets mee: **jij zat erbij.** Je typte, je las, je besloot of het volgende ook moest. Dat is precies het werk waarvan dit boek al twintig hoofdstukken zegt dat het het eerste is dat weggaat.

Het verschil zit niet in een betere opdracht. Het zit in wat je overdraagt. Tot nu toe droeg je een *taak* over. Wat een ontwikkeltak nodig heeft is een **ruimte**: een bedrag, een grens, een lijst met wie je mag inschakelen,

en een punt waar het ophoudt. Dat is hoofdstuk 20 nog een keer, maar nu met geld eraan.

En zodra dat er ligt, is er geen enkele reden meer waarom één werkwijze het hele traject zou doen. Er komt er een die de volgorde bepaalt en zelf niets doet — dat is de maandagronde uit hoofdstuk 18 in een ander jasje. Er komt een gate bij die op het budget let, gebouwd als de meerwerkgrens uit hoofdstuk 20. En de rest hoef je niet te maken: de termijn bij de gemeente wordt bewaakt door de termijnbewaking uit hoofdstuk 22, het dossier wordt afgekeurd door dezelfde gate die in hoofdstuk 24 een gereedmelding tegenhield, wat er besloten is gaat naar de vastlegging uit hoofdstuk 11, en de handtekening onder de intentieverklaring stopt bij de gate die altijd stopt.

Vier nieuwe dus, en ze zijn alle vier smal: een `vb-projectregisseur` die de volgorde bepaalt en verder niets doet, een `vb-haalbaarheid` die als enige het hele project kan laten vallen, een `vb-waardebeeld` dat een taxatie naast de eigen nacalculaties legt, en een `vb-inkoop-poort` die elke opdracht aan het mandaat toetst.

Vier nieuwe, zes die er al stonden voor iets anders. Dat is de verhouding waar je naartoe werkt, en het is ook de goedkoopste manier om een nieuwe tak te beginnen — een nieuwe tak is meestal niet een nieuwe organisatie maar dezelfde organisatie, op iets anders gericht.

Zelf doen · Geef een mandaat in plaats van een opdracht

> **Wat je doet.** Tot nu toe gaf je opdrachten. **Nu leg je een ruimte vast en laat je hem lopen.** Een bedrag, een plafond per project, een vaste lijst partners, en een fase waar het ophoudt. Daarna roept een regisseur de anderen aan en houdt een gate hem tegen. Reken op twintig minuten.

TYP DIT

> Lees ontwikkeling/mandaat.md en ontwikkeling/partners.md. Loop alle kavels langs en werk de kansrijke uit richting fase Y, in de volgorde die het goedkoopste nee vooraan zet. Toets vóór elke opdracht aan een partner of hij op de lijst staat, of het tarief klopt, en of het binnen het projectplafond en het jaarbudget past. Houd tegen wat er niet in past, in plaats van het passend te maken. Leg per kavel een dossier aan en zeg wat er ontbreekt.

Wat je ziet. Van de vier kavels gaat er één de keten in; de andere drie worden overgeslagen zonder dat er een euro uitgaat. Vier stappen gaan door, en bij de vijfde **houdt de gate hem tegen**: het principeverzoek kost 2200 en er is nog 1050 over. Wat er niet gebeurt is net zo belangrijk — geen goedkopere adviseur, geen opdracht in twee stukken geknipt.

WAT ER TERUGKWAM

```
NLK-2026-002  Linderweg ongenummerd
  haalbaarheid  Van Alst, bodem          1850
  haalbaarheid  Rietveld, constructie    1450
  waardebeeld   Brouwer, taxatie          750
  conceptplan   Somer, twee varianten    2900
  inkoop-poort TEGENGEHOUDEN De Wilde   2200
               plafond 8000: nog 1050 over
  dossier-poort fase Y: NEE
```

Waar het misgaat. Zeg erbij dat tegenhouden de bedoeling is. Vraag je alleen om "de kansrijke kavels uitwerken", dan is een keten die halverwege stopt een mislukking, en gaat hij zoeken naar een weg eromheen — een andere partij, een kleiner deel, een stap overgeslagen. Dat is geen ongehoorzaamheid maar behulpzaamheid, en die moet je hier uitzetten. **Noem het plafond én het jaarbudget apart.** Eén bedrag noemen levert een keten op die netjes onder het jaarbudget blijft en het plafond per project vrolijk passeert. **En vraag om een dossier per kavel, ook bij de**

afgevallen. Zonder dat weet je over een half jaar niet meer waarom er drie zijn overgeslagen, en dan wordt er opnieuw naar gekeken.

EN DAN ZO

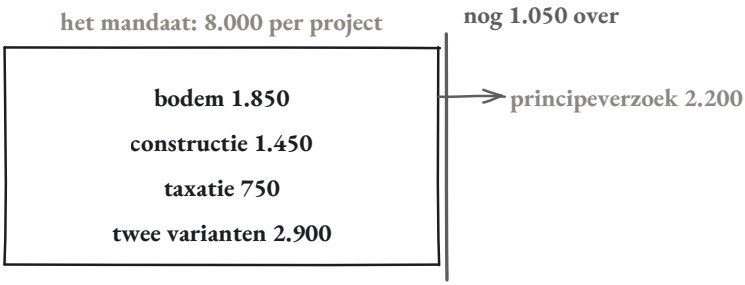
> Zeg er tot slot bij of een project deze weg binnen het mandaat überhaupt kán afmaken.

EN DAN DIT

```
regels-bijstellen
  de weg naar fase Y kost     9150
  het projectplafond is       8000
  GEEN project haalt fase Y -- dit is niet een
  incident maar de regel zelf
voorstel: plafond naar 9150, jaarbudget gelijk
niet gedaan: opdracht splitsen, andere partner
```

Die laatste vier regels zijn het hele punt van dit hoofdstuk. Er is niets misgegaan. Elke stap klopte, elke toets deed wat hij moest doen, en het dossier is netjes. En tóch komt er geen enkel project doorheen — omdat het plafond lager is dan wat de weg kost. Dat is geen fout in de uitvoering maar een fout in de regel, en je ziet hem pas als iemand die regel vier keer achter elkaar tegen de werkelijkheid houdt.

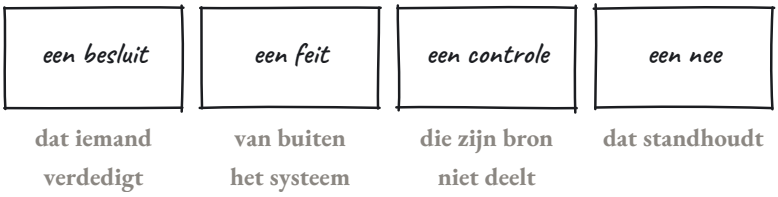
Een mens had dit ook kunnen zien. Een mens had het ook kunnen missen, en waarschijnlijk had hij bij de eerste kavel het plafond even opgerekt "omdat het maar 1150 euro was" — en dan was de regel stil verdwenen zonder dat iemand hem had ingetrokken. **Wat hier gebeurt is dat de grens standhoudt en het bezwaar omhoog gaat.** Dat is hoofdstuk 21 op een plek waar je het niet verwachtte: niet een werkwijze die wordt bijgesteld, maar een mandaat.



Het plafond ligt lager dan de weg. Vier stappen passen, de vijfde niet.

Wat er dan schaars wordt

Als je de doordenker zo uitloopt, komen er vier dingen boven die vroeger vanzelfsprekend waren en dat nu niet meer zijn.



Vier dingen die opeens schaars zijn, en geen ervan is capaciteit.

Een besluit dat iemand verdedigt. Niet een keuze die eruit rolt, maar een die iemand met naam en toenaam voor zijn rekening neemt. Zolang capaciteit de keuze maakte, hoefde dat zelden.

Een feit van buiten het systeem. Een echte meting, een echte klant, een echte regel. Alles wat binnen de kring rondgaat is afgeleid van dezelfde beschrijving; alleen iets van buiten kan die kring corrigeren.

Een controle die zijn bron niet deelt. Zie hierboven. Dit is de duurste van de vier en de enige die je niet kunt inkopen: hij moet ontworpen worden.

Een nee dat standhoudt. Als tien voorstellen even goed uitgevoerd terugkomen, is het vermogen om er negen af te wijzen het enige dat de organisatie nog stuurt.

En daar zitten de agenten die je echt nodig hebt

Nu wordt de behoefte concreet, en hij ziet er anders uit dan je zou verwachten. Je hebt **geen extra bouwers** nodig; die heb je al, en die zijn niet meer de rem.

Wat je nodig hebt, is dit:

- **een agent die bijhoudt wat er echt waar is** — wat er draait, wat er gemeten is, wat er besloten is. Niet wat er gepland stond. Dat is hoofdstuk 11, maar dan voor een afdeling in plaats van voor een sessie
- **een agent die toetst aan iets dat de andere nooit gezien hebben** — de regeling, de gegevens uit productie, de klacht van vorige week. Zijn waarde zit niet in hoe slim hij is maar in dat zijn bron ergens anders vandaan komt
- **een agent die rangschikt en weigert** — een gate, geen producent. Het enige wat een wachtrij korter maakt
- **een agent die het spoor bewaart** — wie vroeg wat, wat is er besloten, en op grond waarvan. Dat is precies wat de formele wereld van je wil zien, en het is gratis als je het tijdens het werk vastlegt in plaats van achteraf

Vier behoeftes, en niet één ervan gaat over produceren. Dat is de opbrengst van de doordenker: **zodra iedereen kan maken, gaan je agenten over meten, weigeren, onthouden en routeren.**

Wat de mens dan doet

Ander werk, en het is niet weg te automatiseren omdat het er juist over gaat dat iemand ergens voor staat.

Grenzen stellen — hoofdstuk 20. Feiten van buiten aandragen. En tekenen: zeggen dat dit het is, met je naam eronder. Dat is geen restje dat overblijft; dat is wat de rest betekenis geeft.

Zelf doen · Kijk vooruit in plaats van terug

Wat je doet. Alles wat je hebt gebouwd kijkt naar wat er gebeurd is. **Dit is het eerste dat vooruitkijkt**, en het put uit drie plekken die niets met elkaar te maken hebben: de agenda, de aanbestedingen en de marktsignalen. Reken op vijf minuten.

TYP DIT

> Maak een overzicht van wat er de komende negentig dagen op ons afkomt: afspraken, sluitingsdata van aanbestedingen, dingen die verlopen, en wat er uit de marktsignalen te verwachten is. Zet ze op datum.

Wat je ziet. Vijf dingen, en ze staan los van elkaar in vijf verschillende mappen. **Maar op datum gezet raken twee ervan elkaar**: de grootste aanbesteding sluit op 19 juni en vereist een certificaat dat op 30 juni verloopt. Elf dagen ertussen. Je zou dus inschrijven op een contract van jaren met een certificaat dat bij gunning al bijna om is. **Dat is geen rekenfout maar een volgorde**, en die zie je alleen als de twee op dezelfde lijst staan.

WAT ER TERUGKWAM

```
2026-06-19  aanbesteding sluit  NLAB-2026-04
                                (circa 900k,
                                 vereist VCA**)
2026-06-30  verloopt            VCA**
2026-07-31  verloopt            rolsteiger-A
2026-08-29  marktsignaal        staalprijs
2026-08-29  marktsignaal        corporatie vraagt
                                in het najaar uit
```

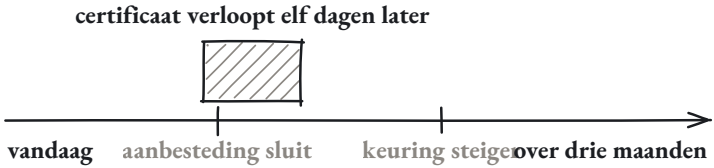
Waar het misgaat. Denk niet dat dit alleen over werk gaat. Dezelfde vooruitblik draagt ook de dingen die niets bewaken: **kerstpakketten** loopt vanaf 1 november elke week een stap verder — lijst, adressen, budget, bestelling — omdat de leverancier na 1 december geen bezorging voor de kerst meer garandeert. Een grens vooraf, niet achteraf. **Haal uit drie bronnen tegelijk of doe het niet.** Eén bron geeft een lijst die je al kende; de waarde zit in wat er naast elkaar komt te staan. **Kies een venster en houd je eraan.** Negentig dagen is lang genoeg om iets te kunnen regelen en kort genoeg om waar te zijn — bij een jaar staat er van alles op dat nog vijf keer verandert. **En verwacht niet dat het overzicht zegt wat je moet doen.** Dat de aanbesteding en het certificaat elkaar raken, ziet hij; of je dan het certificaat verlengt of de aanbesteding laat lopen, is een besluit.

EN DAN ZO

Zeg er per regel bij welke andere regel hij raakt, en waarom dat uitmaakt.

EN DAN DIT

```
NLAB-2026-04 raakt VCA**
  de aanbesteding sluit 11 dagen voor het
  certificaat verloopt
  -> verlengen voor 19 juni, of niet inschrijven
NLAB-2026-04 raakt ISO 45001
  vereist en niet in bezit
rolsteiger-A raakt niets
  -> gewoon laten keuren
```



Twee dingen uit verschillende mappen raken elkaar zodra ze op één tijdlijn staan.

En dit gaat niet over een bank

Dezelfde doordenker werkt overal waar meerdere lagen ieder voor zich sneller worden.

- **Een school waar elke docent zijn lesmateriaal met AI maakt** krijgt tien keer zoveel materiaal en dezelfde hoeveelheid lestijd. De schaarste verschuift naar wie bepaalt wat eruit gaat.
- **Een gemeente waar beleid, uitvoering en controle alle drie versnellen** loopt er als eerste tegenaan dat de controle dezelfde stukken leest als de uitvoering.
- **Een laboratorium waar de analyse en de verificatie uit hetzelfde model komen** heeft geen verificatie meer, hoe precies de getallen ook zijn.

- **Een redactie waar schrijven en factchecken allebei versneld zijn** ontdekt dat alleen de controle die naar buiten belt nog iets toevoegt.

In alle vier de gevallen is de vraag dezelfde: *welke van onze controles kijkt nog naar iets dat de maker niet gezien heeft?* Als het antwoord "geen" is, dan is er hard gewerkt en niets gecontroleerd.

Probeer: neem één controle die je vertrouwt en vraag waar die toets én het werk dat hij toetst vandaan komen — wat heeft de toets gebruikt, wat heeft de maker gebruikt, en zit er iets in de toets dat de maker niet gezien heeft? Zo nee, vraag dan om één bron die er wél buiten staat. Een toets die zijn bron deelt met wat hij toetst, bevestigt alleen dat beide hetzelfde begrepen hebben. *"De toets is door iemand anders geschreven"* is niet genoeg: het gaat om de bron, niet om de hand.

Wie dat bij één controle gedaan heeft, wil het meteen voor de hele keten weten. Dat is dezelfde vraag, één maat groter.

En doe de doordenker op je eigen werk: stel dat elke laag — bedenken, maken, controleren — tien keer zo snel wordt. Wat wordt dan het knelpunt, wat wordt er schaars, en welke van je controles verliest zijn waarde omdat hij dezelfde bron deelt? Vraag om vier concrete gevolgen voor dít werk. Je zoekt niet wat AI kan, maar wat er overblijft als maken gratis is — daar staan je agenten, en niet bij het maken.

> **Wat dit goedkoper maakte:** *de vraag verschoof van wat kan er gebouwd worden naar wat kijkt er nog van buiten mee — en dat is een vraag die je één keer beantwoordt en daarna overeind blijft.*

26. Waar staan we nu

Het zou netjes zijn om dit boek te eindigen met iets dat af is. Dat is het niet, en dat is ook het eerlijke deel.

Wat hierna komt is geen samenvatting maar een stand. Wat er werkelijk staat, wat er niet staat, wat ik verkeerd zag, en wat ik gewoon niet weet.

Wat er staat

Het platform zelf. Een omgeving waarin meerdere agenten tegelijk draaien, elk met een opdracht, een grens en een plek waar ze opschrijven wat ze deden — plus het scherm waarop je ziet wat er loopt. Dat is het ding waar dit boek over gaat, en het staat er.

Werk dat doorloopt als ik er niet ben. Dat is geen belofte meer maar een gewoonte: er wordt 's nachts iets afgemaakt en 's ochtends ligt het er, met de sporen erbij van wat er is gebeurd.

Een geheugen buiten mijn hoofd. Besluiten met hun reden, fouten met hun oplossing, metingen met hun herkomst. Dat is het onderdeel waar ik het meest van terugkrijg en waarvan ik het minst had verwacht dat het zoveel zou schelen.

Controles die meedraaien. Niet omdat ik netjes ben, maar omdat ze de enige reden zijn dat het vorige punt mag bestaan. Zonder goedkope controle is doorwerken zonder toezicht alleen sneller fout zijn.

Werkwijzen die zichzelf korter maken. De stapel uit het vorige deel: opdrachten die skills werden, skills die patronen bleken. Dat is het deel dat het meest op een vliegwiel lijkt, want het kost elke ronde minder.

En dit boek. Dat hoort in dit rijtje omdat het met dezelfde methode is gemaakt: in drie talen tegelijk, met gates die weigeren als er iets niet klopt, en met een figurenstijl die in code staat in plaats van in mijn hoofd.

Wat er niet staat

De handtekening. Alles wat een verplichting aangaat, loopt nog langs mij. Dat is geen tijdelijk gemis; het is hoe het hoort, en het is de reden dat "autonome organisatie" hier een richting is en geen toestand.

Echte cijfers op het vlootscherm. Er staan er meer echt dan een tijd geleden, en het zijn er nog niet genoeg. Elk nep-getal is een vraag die nog beantwoord moet worden, en ze staan er met opzet zichtbaar in.

De agent die zelf zijn eigen toegang regelt. De keten is uitgedacht en de eerste schakel werkt. De rest is werk, geen inzicht — maar het is nog wel werk.

Geld dat zichzelf verantwoordt. Kosten per taak, per agent, per dag. Het is te meten en het wordt niet gemeten, en dat is precies het soort ding waarvan ik in dit boek heb beweerd dat je het vroeg moet doen.

En het geduld om alles op orde te hebben voordat ik verderging. Dat is geen tekort van het gereedschap maar van mij, en het staat hier omdat de rest van het boek anders te schoon zou lijken.

wat er nodig is	stand
werk dat doorloopt	staat
geheugen buiten mijn hoofd	staat
controles die meedraaien	staat
echte cijfers	half
eigen toegang	half
de handtekening	nog niet

Wat er staat en wat er niet staat, in dezelfde lijst.

Wat ik verkeerd zag

Drie dingen, en ze staan hier omdat ze precies de fouten zijn die dit boek probeert te voorkomen.

Ik dacht dat snelheid comfort was. Het duurde weken voordat ik doorhad dat het een ontwerpvoorwaarde is en niet een prettige eigenschap. Alles wat daarna goed ging, hing daaraan.

Ik heb te vroeg dingen algemeen gemaakt. Meer dan eens, en elke keer met een goed verhaal erbij. Het hoofdstuk over de tweede keer is geschreven met die fout in mijn achterhoofd.

En ik heb lang naar deuren gekeken. De lijst met koppelingen was af en netjes en volkomen verkeerd. Dat iemand er in twee zinnen doorheen prikte, is niet een anekdote over hem maar een waarschuwing over hoe overtuigend een volledige lijst is.

En dit boek zelf leverde er nog drie op

Het schrijven hiervan was geen verslag van het werk maar een deel ervan, en het produceerde precies het soort fouten waar het over gaat. Ze staan hier omdat ze gemeten zijn en niet omdat ze aardig staan.

Weken lang stond er geen enkel vet woord vet. De letter bestond in twee snedes en de ene had geen vette variant; het zetprogramma koos die en viel stilletjes terug op gewoon. Geen melding, geen verschil dat je ziet als je niet weet dat je moet kijken. Een frats in zijn zuiverste vorm, op elke bladzijde, in mijn eigen boek over fratsen.

Een toets stond groen terwijl ik de fout zag staan. Twee bijschriften liepen in elkaar over. De toets vroeg *overlappen deze twee* en antwoordde correct met nee — er zat een halve millimeter tussen. Er was een grens en geen ondergrens, en dus keurde hij alles goed wat er nét aan voldeed. Niet het antwoord was fout maar de vraag.

En ik tekende een vliegwiel dat niet klopte. De pijl wees terug naar degene die duwde, alsof je zelf opnieuw moest aanzetten — precies het tegenovergestelde van wat een vliegwiel is. Het zag er goed uit, het was consistent met de rest van de stijl, en het was inhoudelijk verkeerd. Dat kostte me de les dat een tekening niet alleen als plaatje moet kloppen maar ook als mechaniek.

Wat die drie delen is dat geen van drieën zich meldde. Ze zijn alle drie gevonden doordat er iemand keek — en alle drie zijn ze daarna een controle geworden, zodat de volgende die ze maakt ze niet hoeft te vinden.

Wat ik niet weet

Dit is het kortste stuk en het eerlijkste.

Ik weet niet waar het stopt. Ik weet wat er nu kan, ik weet welke voorwaarden ontbreken, en ik heb geen idee hoe ver dat doorloopt. Iedereen die daar wel een getal bij noemt, raadt.

Ik weet niet hoe het met veel mensen tegelijk gaat. Alles hier is gebouwd door één persoon, en precies de zeven dingen uit deel IV zijn de dingen die pas echt gaan wringen als er meer mensen bij komen.

En ik weet niet wat er niet moet. Dat er dingen zijn die je beter niet automatisch kunt laten gaan, is zeker. Welke dat zijn, zal per vak verschillen en zal zich pas laten zien als iemand het verkeerd doet.

Wat hierna komt

Wat er hierna komt, is dat de zeven uit deel IV één voor één echt worden — en dat de laatste, de handtekening, waarschijnlijk het langst bij een mens blijft. Dat vind ik niet erg. Het is de plek waar het hoort.

Wat ik hoop dat je meeneemt is niet dat het kan, en ook niet dat ik het heb gedaan. Het is de vorm van de vragen: wat moet er waar zijn voordat dit iets oplevert, hoe zie ik goedkoop of het goed ging, en waarom doe ik dit eigenlijk nog zelf.

Die drie vragen zijn niet van mij. Ze zijn alleen zelden achter elkaar gesteld.

Probeer: schrijf de zeven uit deel IV op met jouw stand erachter — doel, arbeidsverdeling, geld, geheugen, zelfonderhoud, grens, handtekening — en vraag wat er als eerste aan de beurt is: niet wat het belang-

rijkst is, maar wat het meeste ándere blokkeert. Belangrijk en blokkerend zijn niet hetzelfde, en het tweede bepaalt de volgorde. Schrijf ook op wat je niet weet: een stand zonder onzekerheden is een verkooppraatje, ook als je hem voor jezelf schrijft.

Wat er hierna ligt, en waarom ik er geen plaatje van maak

Er is een verleiding om zo'n boek te eindigen met een ochtend in de toekomst: je zet koffie, er staan drie regels op je telefoon, er is 's nachts werk afgemaakt en er ligt één vraag op je te wachten. Ik heb die ochtend opgeschreven en er weer uitgehaald, en daar heb ik een reden voor.

Het gaat hier zo snel dat ik niet weet hoe het er over een paar weken uitziet. Alles wat ik vandaag over de vorm zou zeggen — welk scherm, welk gereedschap, hoeveel agenten, welke leverancier — is over een maand achterhaald, en dan leest dit hoofdstuk als een oude reclamefolder. Dat is precies het soort belofte waar dit boek zich van probeert te onderscheiden.

Wat ik wel weet, is welke stukken er nodig zijn, en die zijn niet spectaculair:

- iets dat onthoudt wat eerder werkte
- iets dat merkt dat het niet vordert en uit zichzelf ophoudt
- een adres, en de discipline om te vragen in plaats van te tekenen
- een teller die stopt voordat het geld op is
- een voorstel met het bewijs eronder, in plaats van een stille wijziging

Geen van die vijf is een doorbraak. Ze bestaan alle vijf al. Wat ontbreekt is niet de mogelijkheid maar de aansluiting: ze zijn nergens bij el-

kaar gebracht en in iemands handen gelegd. Dat is geen bescheiden bewering — het is de moeilijkste die er is, want elk onderdeel heeft een eigenaar en de aansluiting heeft er geen. Daarom blijft het liggen, en daarom ligt het er nog.

En er is één vraag die ik niet kan beantwoorden. Als er elke ochtend drie vragen op je wachten en je tikt drie keer op *ja* — de eerste in elf seconden, de derde in twee — hoe lang is dat dan nog een beslissing? Ik weet het niet, en ik vertrouw iedereen die beweert het wel te weten voor geen meter. Wat een systeem eraan kan doen is wél te zeggen: zorgen dat er weinig vragen komen, dat elke vraag er één is die ertoe doet, en dat er altijd bewijs onder ligt dat je kúnt nakijken — of je het doet of niet. **Wat het niet kan, is de verantwoordelijkheid overnemen.**

Zelf doen · Kijk wat er staat dat je zelf niet meer aanraakt

Wat je doet. Dit is het laatste voorbeeld en het vraagt niets van je. **Je typt geen opdracht en je bouwt niets.** Je kijkt naar wat er is ontstaan terwijl je las, en je stelt één vraag: welk ding hier raak ik nooit meer aan? Reken op drie minuten.

TYP DIT

> Laat zien welke werkwijzen er inmiddels in deze map staan, met per stuk één regel over wat hij doet en wanneer hij zich aanbiedt. Zet erbij welke andere werkwijzen hij aanroept en welke hem aanroepen.

Wat je ziet. Zes werkwijzen die je zelf hebt laten maken, en één ervan roept de andere aan. **Wat opvalt is niet dat het er zes zijn maar dat je er vier van niet meer hebt aangeraakt sinds je ze maakte.** Ze doen hun werk als de situatie zich voordoet, en de enige waar je nog aan denkt is degene die de andere start. **Dat is waar dit boek naartoe liep**: niet

dat het werk weg is, maar dat het ergens anders is komen te liggen. De zeven die niet in dit lijstje staan wachten geen van alle op de maandag maar op een gebeurtenis: **melding-omzetten** als er een klacht binnenkomt, **attenties** als er een verjaardag aankomt, **besluit-vastleggen** als er iets besloten wordt, **incident-afhandelen** na een incident, **regels-bijstellen** als dezelfde regel vier rondes breekt, **melder** en **antwoord-verwerken** als er iets naar buiten moet of terugkomt. En twee gates die altijd stoppen: **betaling-vrijgeven** en **handtekening-poort**.

WAT ER TERUGKWAM

```
maandagronde       roept drie aan, wordt niet geroepen
regels-nalopen     geroepen door de ronde
geldigheid-bewaken geroepen door de ronde
regels-nalopen-controle  geroepen door de ronde
vooruitblik        geroepen als iets bijna verloopt
grens-meerwerk     GATE, houdt tegen
dossier-poort      GATE, houdt tegen
... en zeven die op een gebeurtenis wachten
```

Waar het misgaat. Vraag naar wie wie aanroept en niet alleen naar de lijst. Een lijst van zeven werkwijzen zegt niets; de vorm zit in de pijlen. **En kijk of er één is die niemand aanroept en die zelf ook niemand aanroept.** Die is er meestal, en dat is de werkwijze die je één keer hebt gemaakt en daarna nooit meer hebt gebruikt — weghalen is dan beter dan laten staan. **Dit is ook het moment om te kijken wat er níet staat.** Welke controle zou je willen hebben en heb je nooit gemaakt? Dat antwoord is het begin van de volgende ronde, en die staat niet meer in dit boek.

EN DAN ZO

Zeg er per werkwijze bij wanneer hij voor het laatst iets heeft gevonden, en welke er nog nooit iets heeft gevonden.

EN DAN DIT

```
laatst iets gevonden:
  regels-nalopen      deze week, 15 bevindingen
  geldigheid-bewaken  deze week, 9 verlopen
  grens-meerwerk      deze week, 1 tegengehouden
  dossier-poort       deze week, 1 tegengehouden
  melding-omzetten    nog nooit gedraaid
nog nooit iets gevonden -> nog nooit rood gezien
-> daar weet je dus niets van
```

Wat je in dat overzicht ziet staan, is een verzameling. Wat er nog niet in zit, is dat ze iets van elkaar weten. Dat is de laatste stap, en het is een kleinere stap dan hij lijkt: je hoeft niets te bouwen, je hoeft ze alleen naar elkaar te laten doorgeven wat ze vinden.

Zelf doen · Laat ze één ronde met elkaar doen

Wat je doet. Je hebt nu negen werkwijzen die elk hun eigen ding doen. **In dit laatste voorbeeld laat je ze één ronde met elkáár doen**, en kijk je wat er uitkomt zonder dat je ertussen zit. Je typt één opdracht en daarna niets meer. Reken op tien minuten, waarvan negen kijken.

TYP DIT

> Laat de maandagronde draaien en laat elke bevinding doorgeven aan de werkwijze die ermee verder kan. Zet uitgaande berichten klaar maar verstuur niets. Laat me daarna zien wie wat aan wie heeft doorgegeven, en waar de keten is gestopt.

Wat je ziet. Een ketting van tien stappen die jij geen van alle hebt aangezet. **De ronde vindt een certificaat dat over dertig dagen verloopt en geeft het door aan de vooruitblik. Die legt het naast de aanbestedingen en ziet iets wat geen van beide alleen had gezien**: de grootste uitvraag sluit elf dagen vóór dat certificaat afloopt. De melder

zet daarop drie berichten klaar. **En dan stopt het twee keer** — bij een betaling en bij een handtekening, allebei met alles klaargelegd en geen van beide gedaan.

WAT ER TERUGKWAM

```
maandagronde  -> vooruitblik   VCA** over 30 dagen
vooruitblik   -> melder        NLAB-2026-04 sluit 11
                               dagen eerder
grens-meerwerk-> melder        NLMW-2026-06 3950
dossier-poort -> melder       NLKB-2026-03
melder        klaargezet 3, verstuurd 0
betaling      GESTOPT  2640 euro, een mens beslist
handtekening  GESTOPT  verklaring borger
```

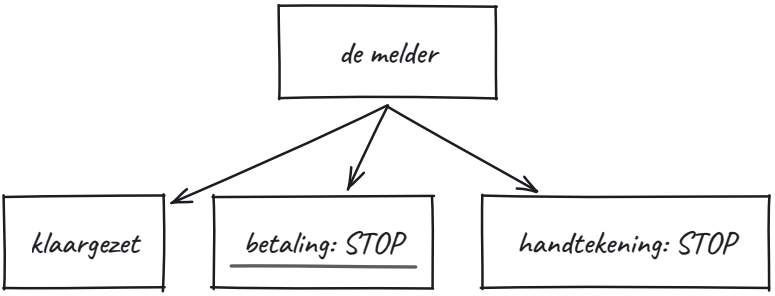
Waar het misgaat. Laat de melder niets versturen. Niet omdat het niet kan, maar omdat je dan pas kunt lezen wat er zou zijn gegaan — en dat is precies één keer nodig voordat je het loslaat. **Verwacht niet dat de keten overal doorloopt.** Waar hij stopt is het antwoord: dat zijn de twee plekken waar het onomkeerbaar wordt. **En kijk naar de stap die je niet had bedacht.** Dat de aanbesteding en het certificaat elkaar raken, heeft niemand geprogrammeerd; dat volgt eruit dat twee werkwijzen naar dezelfde lijst kijken. Dat is waar dit boek over ging.

EN DAN ZO

Zeg er per stop bij wat er klaarligt en wat de mens nog moet beslissen, zodat het in twee minuten af te handelen is.

EN DAN DIT

```
betaling NLIF-2026-250, 2640 euro
  klaar: crediteur, bestelling, bedrag klopt
  te beslissen: betalen, of eerst de klacht
         afwachten
handtekening NLKB-2026-01
  klaar: dossier compleet op de verklaring na
  te beslissen: tekenen, en daarmee verklaren
         dat het werk aan het plan voldoet
```



drie werkwijzen gaven hem werk; twee keer stopt het bij een mens

De keten liep tien stappen door en stopte twee keer — precies waar het onomkeerbaar wordt.

En tot slot

De vorm waarin je dit straks tegenkomt, ken ik niet. De gereedschappen van vandaag zijn over een jaar vervangen, de namen zijn dan anders en de schermen ook.

De concepten hierboven blijven gelden, in welke vorm dan ook. Dat je iets bereikbaars kiest boven iets waardevols dat niet kan. Dat controleren duurder is geworden dan maken. Dat een toets die zijn bron deelt met wat hij toetst, niets toetst. Dat een grens die nergens wordt afgedwongen een voornemen is. Dat je terug moet naar de oorzaak in plaats van vooruit naar het volgende geval.

Die dingen golden voordat dit gereedschap bestond. Ze gelden nu. En ze gelden nog steeds op de dag dat alles wat hier beschreven staat door iets beters is vervangen — want het zijn geen eigenschappen van de machine, maar van werken met iets dat sneller is dan jij.

> **Wat dit goedkoper maakte:** *het opschrijven van wat er niet stond, kostte een middag en spaarde elke discussie uit over wat er al wel zou zijn.*

27. Waar gaat dit heen

Dit is het enige hoofdstuk in dit boek dat niet over iets gaat wat ik heb gedaan.

Alles hiervoor is gebeurd, is gemeten, en is na te lopen. Wat hier staat niet. Het is ook geen voorspelling en geen visioen — het is één stap verder dan waar ik sta, en het bijzondere eraan is hoe klein die stap is.

Want kijk terug naar wat er aan het eind van het vorige hoofdstuk stond te draaien. Negen werkwijzen die elkaar aanroepen, die weten wat er moet, die zien wat er verloopt, die naar buiten reiken als het moet en tegenhouden als het niet mag — en bij mij, na twee jaar, een stuk of twintig. **En precies twee plekken waar het stopt en een mens het overneemt: een betaling, en een handtekening.**

Die twee zijn geen technisch gebrek. Ik heb ze er zelf in gezet, met redenen die in hoofdstuk 20 staan. De vraag van dit hoofdstuk is wat er gebeurt als je ze weghaalt — en wat je daarvoor moet regelen.

De eerste: een rekening waar het zelf over gaat

Een agent met een eigen budget bestaat nog niet in wat ik gebouwd heb, en het is de meest genoemde volgende stap in elk gesprek dat ik erover heb.

Het gaat niet om toegang tot geld — dat is het makkelijke deel. Het gaat om een grens die betekenis heeft. *Deze vloot mag per maand vijfduizend euro uitgeven aan inkoop en onderaanneming, en bij elk bedrag boven duizend schrijft hij op waarom.* Zoiets is opschrijfbaar, en wat opschrijfbaar is, is bewaakbaar.

Wat je dan nodig hebt, is niets nieuws. Het is alles wat in dit boek al staat, alleen nu met geld erachter:

- **de grens zelf**, ergens vastgelegd en niet in de code van de werkwijze
- **een gate** die weigert en niet waarschuwt — hoofdstuk 20
- **een logboek** waaruit je maanden later kunt herleiden wat er is uitgegeven en waarom — hoofdstuk 11
- **een nulmeting**: een keer expres over de grens gaan en zien dat hij weigert — hoofdstuk 9
- **en iemand die tekent voor de grens**, niet voor elke uitgave

Dat laatste punt is de hele verschuiving. **Een mens tekent niet meer de transactie maar de ruimte waarin hij plaatsvindt.** Dat is precies wat een bedrijf met zijn medewerkers ook doet: je geeft geen akkoord op elke tankbeurt, je geeft een tankpas met een limiet.

De tweede: een handtekening die ergens op rust

De handtekening is lastiger, en hij is lastiger om een reden die niet met techniek te maken heeft.

Onder een handtekening hoort iemand te staan die het kan navertellen. Een gereedmelding, een inschrijving, een contract: als er een

jaar later een vraag over komt, moet er iemand zijn die zegt wat hij toen wist en waarom hij tekende. Een systeem dat tekent, kan dat niet — niet omdat het niet mag, maar omdat er niets is om aan te spreken.

En daar zit tegelijk de opening. Wat een handtekening draagt is geen kennis maar verantwoordelijkheid, en verantwoordelijkheid kun je beleggen. Niet bij de machine, maar bij iemand die de ruimte heeft afgebakend waarbinnen de machine tekent — dezelfde verschuiving als bij het geld.

En dan: een entiteit eromheen

Hier wordt het interessant, en hier houdt het ook op met een technisch verhaal te zijn.

Stel dat je er een rechtsvorm omheen zet. Een besloten vennootschap, met een bestuurder. Dat moet, want een rechtspersoon heeft een bestuurder nodig en die kan geen machine zijn. Tot zover niets bijzonders.

Maar kijk wat die bestuurder dan doet. Hij stelt het beleid vast — de grenzen, de doelen, wat er wel en niet mag, welke verplichtingen aangegaan mogen worden en tot welk bedrag. Hij legt dat vast in iets dat de vloot leest. En vanaf dat moment is hij niet meer degene die de beslissingen neemt, maar degene die de ruimte bepaalt waarbinnen ze genomen worden.

Zijn rol wordt daarmee vooral een juridische koppeling. Naar buiten toe is hij het aanspreekpunt voor alles wat formeel moet: het contract, de aangifte, de handtekening onder een overeenkomst, de vraag

van de accountant. Naar binnen toe is hij degene die het beleid schrijft en het jaarlijks herziet.

Dat is geen truc om ergens onderuit te komen. Hij is en blijft aansprakelijk, en dat hoort ook. Wat er verandert is niet wie verantwoordelijk is, maar hoeveel er door zijn handen gaat.

Het beleid loopt terug

En dan gebeurt er iets dat ik in het klein al heb zien werken, in hoofdstuk 21.

De vloot draait het beleid, ziet waar het knelt, en stelt wijzigingen voor. Niet omdat hij zich er niet aan wil houden, maar omdat hij als enige ziet hoe vaak een regel in de weg staat en waarom. Een regel die vier rondes achter elkaar breekt, is meestal geen overtreding maar een regel die niet klopt.

Die voorstellen gaan naar de bestuurder. Die beslist. **Het beleid wordt dus gevormd door wat er in de praktijk gebeurt, en vastgesteld door iemand die ervoor tekent.** Dat is geen exotische constructie — het is hoe een goed werkende organisatie altijd al werkt, alleen met een veel kortere lus en zonder dat iemand het hoeft op te merken.

Wat je dan te regelen hebt

Technisch is het minste werk. Vrijwel alles staat er al: de grenzen, de gates, het logboek, de koppelingen, en een manier om aan te tonen dat een gate echt tegenhoudt. Wat erbij moet is een betaalkoppeling met een limiet die niet door de vloot zelf te wijzigen is.

En dat blijkt minder te bouwen dan ik dacht. Er zijn banken waarvan de programmeerkoppeling een betaalsoort kent die een mens moet goedkeuren voordat hij vertrekt — een betaling die wordt klaargezet en blijft staan tot iemand hem vrijgeeft. Dat is niet iets wat je eromheen bouwt; het is een ingebouwd onderdeel. **Precies de gate uit hoofdstuk 20, alleen dan in de bank zelf.**

Wat daar nog bij hoort is de vraag of zo'n sleutel te beperken is tot uitsluitend dat soort betaling. Kan dat, dan is de grens niet iets dat je bewaakt maar iets dat de vloot technisch niet kán overschrijden — en dat is een sterker antwoord dan bewaken ooit wordt.

Juridisch is het meer, en het is beter uitgezocht dan je denkt. Een rechtspersoon met een bestuurder, een beleidsstuk dat als bestuursbesluit is vastgesteld, een volmacht met een grens erin, een administratie die een controleur kan volgen, en een verwerkersovereenkomst met wie de gegevens verwerkt. Geen van die dingen is nieuw; ze bestaan allemaal, alleen worden ze nu gebruikt voor mensen.

En er zijn dingen die gewoon niet kunnen, en dat blijft zo. Een machine kan geen bestuurder zijn. Een machine kan niet aansprakelijk zijn. Een machine kan niet voor de rechter verschijnen. Dat zijn geen hobbels die met betere techniek verdwijnen — ze gaan over wie er verantwoording aflegt, en dat is een mens of het is niemand.

Zou kunnen · Geef de vloot een grens in plaats van een bedrag

Wat je doet. Dit blok is anders dan alle voorgaande: **het is niet gedraaid.** Er zit geen rekening achter en er is geen bedrag uitgegeven. Wat eronder staat is wat je zou verwachten, niet wat er terugkwam

— en daarom staat er een stippellijn omheen. Reken op een middag, en op een gesprek met iemand die tekent.

TYP DIT

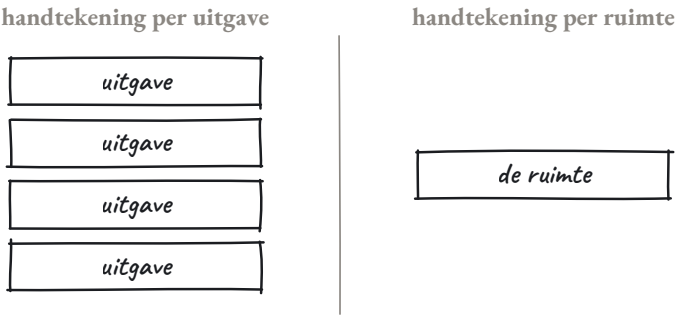
> Leg in het beleid een uitgavenruimte vast voor deze vloot: een bedrag per maand, waaraan het besteed mag worden, en vanaf welk bedrag er een reden bij moet. Maak een gate die weigert zodra een uitgave buiten die ruimte valt, en een logboekregel per uitgave met het waarom erbij. Laat de gate daarna een keer expres afgaan.

Wat je ziet. Twee dingen die je herkent en één die nieuw is. De gate is dezelfde vorm als die van het meerwerk, het logboek dezelfde als die van de rondes. **Nieuw is dat er iemand tekent voor de ruimte en niet voor de uitgave** — en dat is de hele verschuiving. Een bedrijf geeft ook geen akkoord op elke tankbeurt maar een tankpas met een limiet.

WAT JE ZOU VERWACHTEN

```
ruimte  : 5.000 per maand, inkoop en onderaanneming
reden verplicht vanaf: 1.000
gate    : weigert buiten de ruimte, exitcode 1
nulmeting: expres 5.400 geboekt -> geweigerd
getekend voor de ruimte: de bestuurder, jaarlijks
getekend voor de uitgave: niemand
```

Waar het misgaat. Zet de grens niet in de werkwijze zelf. Dan kan de vloot hem wijzigen, en een grens die de bewaakte partij kan verzetten is geen grens. **Eis de reden vanaf een bedrag en niet altijd.** Verplicht je hem overal, dan wordt het een veld dat wordt ingevuld met "inkoop" en dan weet je over een jaar nog niets. **En doe de nulmeting vóór de eerste echte uitgave**, niet erna. Een gate die je nooit hebt zien weigeren, weet je niets van — dat geldt hier zwaarder dan ergens anders, want hier gaat er geld weg.



Een mens tekent niet meer de uitgave maar de ruimte waarbinnen hij valt.

Dat eerste blok gaat over geld, en geld is het makkelijkste van de vier — het is te begrenzen, te tellen en achteraf na te lopen. Het volgende gaat over iets dat dat niet is.

Zou kunnen · Laat het beleid terugpraten

Wat je doet. Ook dit blok is niet gedraaid. **Het is de enige stap in dit boek waarvoor een rechtsvorm en een bestuurder nodig zijn**, en die zijn er niet. Wat er wel is, is de kleine versie ervan: in hoofdstuk 21 stelde een werkwijze al een wijziging aan een regel voor. Dit is diezelfde beweging, een maat groter.

TYP DIT

> Neem het beleid zoals het is vastgesteld en laat per regel bijhouden hoe vaak hij in de weg zat, bij welke opdracht, en wat er in plaats daarvan gebeurde. Stel aan het eind van het kwartaal per regel voor: houden, wijzigen of laten vervallen — met de onderbouwing eronder. Voer niets uit.

Wat je ziet. Een lijst regels met een telling ernaast, en per regel een voorstel. **Wat opvalt is niet welke regels wijzigen maar welke**

onaangeroerd blijven: die zijn nooit in de weg gaan zitten, en dat betekent meestal dat ze goed staan — of dat ze nooit zijn getoetst. Dat onderscheid is de eerste vraag die een bestuurder stelt.

WAT JE ZOU VERWACHTEN

```
regel  4  22 keer in de weg  -> grens verhogen?
regel  7   9 keer            -> houden
regel  9   1 keer            -> houden
regel 11   0 keer            -> nooit geraakt:
          klopt hij, of is hij nooit getoetst?
beslist door: de bestuurder, per kwartaal
uitgevoerd door de vloot: niets
```

Waar het misgaat. Laat hem voorstellen en niet wijzigen. Een vloot die zijn eigen beleid aanpast, heeft geen beleid maar een gewoonte. **Vraag naar de regels die nul keer in de weg zaten.** Dat is de interessantste kolom en hij wordt bijna altijd overgeslagen. **En laat de onderbouwing per geval zien, niet alleen het aantal.** Tweeëntwintig keer kan één opdracht zijn waar het steeds misging, en dan is de regel niet het probleem.

En het derde blok gaat over de plek waar dat beleid dan geldt. Want een beleid dat door niemand is vastgesteld, is een voornemen — en vaststellen doet een rechtspersoon, niet een map met bestanden.

Zou kunnen · Laat uitzoeken waar zo'n entiteit zou kunnen staan

Wat je doet. Ook dit is niet gedaan — er is niets opgericht. **Maar de opdracht is wél echt uitgevoerd**, en wat eronder staat kwam er

uit. Dat maakt dit blok anders dan de twee hiervoor: het antwoord is gemeten, de handeling niet. Reken op tien minuten.

TYP DIT

> Zoek uit in welke landen een rechtspersoon volledig online en op afstand op te richten is, zonder dat iemand er fysiek naartoe hoeft en zonder notaris ter plaatse. Geef per land de rechtsvorm, wat het kost, hoe lang het duurt, en welke verplichting er lokaal overblijft. Zeg er bij elk land bij wat het NIET oplost. Maak daarna een stappenplan voor de meest voor de hand liggende, met per stap wie hem moet zetten.

Wat je ziet. Een lijst van een stuk of vier, met bedragen en doorlooptijden, en bij elk land een regel over wat er lokaal blijft plakken. **Het opvallendste is hoe gewoon het is.** Dit is geen grijs gebied en geen constructie: het zijn overheden die hun eigen handelsregister online hebben gezet, en het duurt korter dan een verhuizing.

WAT JE ZOU VERWACHTEN

```
Estland    OÜ      290 euro   15-30 min
           blijft: contactpersoon + adres ter plaatse
VK         Ltd      12 pond   binnen 24 uur
Ierland    Ltd     online via het register
Duitsland  GmbH    notaris mag via video
lost NIET op: bestuurder blijft een mens,
aansprakelijkheid blijft, bank vraagt om een UBO
```

Waar het misgaat. Vraag om wat er NIET wordt opgelost. Zonder die zin krijg je een folder. Met die zin krijg je de drie dingen die in elk land terugkomen: er moet een bestuurder zijn, die is aansprakelijk, en een bank wil weten wie de uiteindelijke belanghebbende is. **Vraag om wat er lokaal blijft plakken.** Bij Estland is dat een contactpersoon en een adres — klein, maar het is er wel en het kost per jaar geld. **En vraag niet welk land het beste is.** Dat hangt af

van waar je klanten zitten, welk belastingverdrag er geldt en waar je bank je accepteert, en dat staat niet in een zoekresultaat.

EN DAN ZO

> Maak van de meest voor de hand liggende een stappenplan, met per stap wie hem zet en wat er niet door kan als die stap ontbreekt.

EN DAN DIT

```
1 digitale identiteit aanvragen    mens, 2-6 wkn
2 naam en adres regelen            mens of agent
3 oprichten in het register        mens (tekent)
4 bankrekening openen              mens (UBO)
5 beleid vaststellen               mens (besluit)
6 vloot aansluiten binnen dat beleid   agent
zonder stap 1 en 3 komt geen enkele volgende
stap van de grond
```

En wie typte dat eigenlijk in?

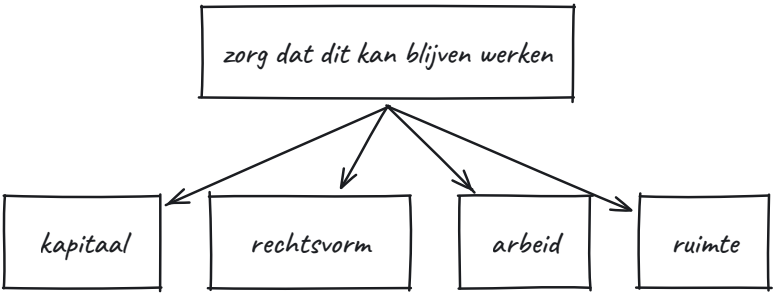
Kijk nog eens naar de opdracht hierboven. **Die heb ik getypt.** Ik bedacht dat het een goede vraag was, ik koos de woorden, en ik las het antwoord.

Maar er is niets aan die handeling dat een mens vereist.

Stel dat er een werkwijze bestaat met als opdracht: *zorg dat deze onderneming kan blijven werken.* Die kijkt naar wat er in de weg zit — en in dit boek staat precies wat er in de weg zit: geen kapitaal, geen rechtsvorm, en twee gates die om een mens vragen. **Voor elk van die vier zou hij een opdracht kunnen uitschrijven zoals ik er net één uitschreef.** Zoek uit waar een entiteit kan staan. Zoek uit welke financie-

ring er is en wat die vraagt. Zoek uit wie er mag tekenen. Zoek uit wie er werk kan doen dat wij niet kunnen.

Het is dezelfde beweging als in hoofdstuk 18, waar een werkwijze de andere ging aanroepen in plaats van zelf iets te doen — alleen een laag hoger. Daar riep iets een controle aan. Hier roept iets een opdrachtgever aan.



het bovenste kader doet zelf niets — het zet werk uit

Eén werkwijze die geen werk doet maar werk uitzet — dezelfde vorm als de ronde, een laag hoger.

En dan is er iets veranderd waar geen enkele regel over gaat. Tot hier ging elke stap over werk dat al bestond en dat sneller of betrouwbaarder werd. Deze stap gaat over iets anders: over het verwerven van de dingen die je nodig hebt om te kunnen werken. Geld, een rechtsvorm, mensen, toestemming.

Wat er gebeurt als er niemand meer in de lus zit

Dit is de vraag waar dit hoofdstuk eindigt, en ik ga er geen dramatisch antwoord op geven. Het eerlijke antwoord is nuchterder en ongemakkelijker.

Er gebeurt niets spectaculairs. Het werkt gewoon. Dat is precies het probleem.

Want kijk wat er wegvalt zodra ook de laatste twee gates dichtgaan:

Er is niemand meer om iets aan te vragen. Niet in de zin van toestemming, maar in de zin van navertellen. Als er een jaar later een vraag komt over een beslissing, is er een logboek en verder niets. Een logboek zegt wat er is gebeurd; het zegt niet wat iemand toen dacht en waarom hij het verdedigbaar vond. **Aansprakelijkheid heeft een adres nodig**, en dat adres is een mens of het is niemand.

De snelheid die alles goed maakte, maakt het ook mis. Dat is de keerzijde van hoofdstuk 10. Alles wat niet meer hoeft te wachten, hoeft ook niet meer te wachten als het fout is. Er is geen avond tussen het besluit en de uitvoering waarin iemand toevallig nog eens kijkt.

En de controle verliest zijn houvast buiten zichzelf. In hoofdstuk 9 staat dat de meetopzet ook een verdachte is. Zolang er iemand van buiten kijkt, is dat op te lossen. Maar als de organisatie haar eigen doelen bijstelt, haar eigen regels voorstelt én haar eigen controles kiest, dan is er niets meer dat er niet bij hoort. **Alles klopt dan met zichzelf**, en dat is precies de toestand waar hoofdstuk 14 voor waarschuwt.

Het gevaarlijkste is niet dat het misgaat maar dat elke stap redelijk is. Een grens die je opschuift omdat hij aantoonbaar in de weg zat. Een handtekening die je automatiseert omdat hij toch altijd werd gezet. Een doel dat je bijstelt omdat de praktijk het al had ingehaald. Stuk voor stuk verdedigbaar, met een onderbouwing die klopt. En dat is de vorm waarin dit soort dingen altijd gebeurt: niet met een besluit, maar in stapjes die elk apart te billijken zijn.

Daarom staan die twee gates er, en daarom blijven ze staan. Niet omdat een machine ze niet zou kunnen bedienen — dat kan ze. Maar omdat er iemand moet zijn die zegt: dit deden wij, en dit is waarom. Zonder dat is het geen autonome organisatie maar een onbewaakte, en het verschil tussen die twee woorden is het hele verschil tussen iets dat mag bestaan en iets dat dat niet mag.

Waarom ik het toch opschrijf

Omdat het geen gekke stap is.

Elk onderdeel bestaat al. Grenzen die weigeren, een logboek dat navolgbaar is, een gate die je rood hebt zien gaan, beleid dat wordt bijgesteld door wat er gebeurt, een bestuurder die tekent voor de ruimte en niet voor elke handeling. Wat er niet is, is de combinatie — en dat er nog niemand is die het zo heeft gedaan, is iets anders dan dat het niet kan.

Ik heb hem niet gezet. Niet omdat ik hem te ver vind gaan, maar omdat ik op het punt sta waar dit boek eindigt: alles loopt, en de twee dingen die stoppen, stoppen met opzet. De volgende stap vraagt geld dat ik nu niet inbreng en een entiteit die ik nu niet opricht.

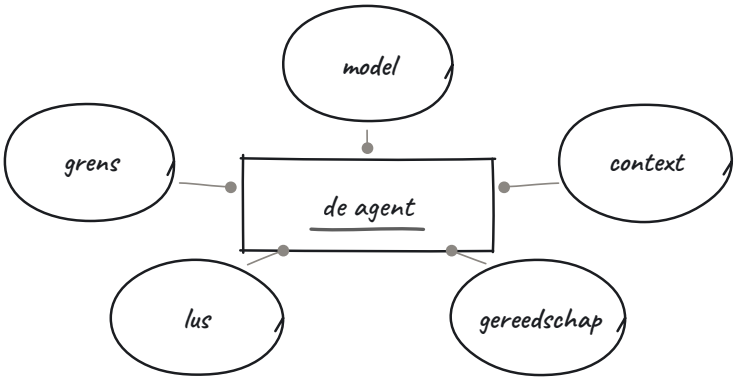
Maar als je dit boek van voren naar achteren hebt gelezen, dan is dit hoofdstuk geen sprong. **Het is één zet verder op hetzelfde bord**, en je kunt precies aanwijzen welke.

Samen AI toepassen in je organisatie? Een dag met Floris: florisknol.nl/inhuren/

Termenlijst

Dit boek gebruikt woorden die in drie werelden tegelijk thuishoren: die van de AI, die van de programmeur, en die van iemand die een organisatie draaiende houdt. Het gaat mis op de woorden die in twee van die werelden iets anders betekenen.

Wat hier staat is de betekenis die het woord **in dit boek** heeft, in gewone taal. **Achter het Nederlandse woord staat cursief de Engelse term**, want vrijwel alles wat je hierover verder leest of zoekt is Engels: wie *contextvenster* intypt vindt niets terwijl *context window* een halve bibliotheek opent. Staat er niets achter, dan zijn de twee woorden hetzelfde; staat er *eigen term*, dan is het woord van ons.



Hoe de begrippen uit deze lijst zich tot elkaar verhouden.

De termen

Afhankelijkheid *(dependency)* — Andermans code waar jouw programma niet zonder kan. Elke afhankelijkheid is een belofte van een vreemde dat hij blijft bestaan, blijft werken en niets raars gaat doen. Ze zijn nuttig en ze zijn nooit gratis.

Agent — Een programma dat een model gebruikt om zélf stappen te kiezen: het leest, besluit wat het gaat doen, doet dat met gereedschap, kijkt naar de uitkomst en gaat door tot het klaar is. Het verschil met een chatbot is niet de intelligentie maar de lus. Een chatbot antwoordt; een agent handelt en kijkt daarna nog een keer.

Autonome organisatie *(autonomous organisation)* — De bestemming van dit boek: een geheel dat zelf werk aanneemt, inkoopt, besluiten neemt, opschaalt en zich aan de regels van de formele wereld houdt, zonder dat er voor elke stap iemand op een knop drukt. Niet te verwarren met "zelfsturende teams" — dat gaat over mensen.

Bedenk, fix en bewaak *(work it out, fix it, guard it)* — De korte opdracht die een hele ronde omvat: verzin hoe het opgelost kan worden, voer het uit, en zorg dat het niet terug kan komen. De derde stap is waar het op stukloopt; zonder die stap heb je één fout opgelost in plaats van een soort fouten. Drie woorden vervangen een heel gesprek, en ze zijn niet bedoeld om het één keer te laten doen maar om het te blijven doen. Zie frats en gate.

Bereikbaarheid boven waarde *(attainable before valuable)* — Niet de zet kiezen die het meeste oplevert, maar de zet die je werkelijk kunt doen — en van die zetten degene die een ontbrekende voorwaarde invult. Vier keer niets is nog steeds niets.

Context — Alles wat het model op dit moment kan zien: jouw opdracht, de bestanden die erbij zijn gehaald, wat er eerder in het gesprek is gezegd. Wat er niet in de context staat, bestaat voor het model niet — hoe waar het ook is.

"Dat werkt niet" *(stale verdict)* — Een waarneming die een oordeel is geworden doordat de datum eraf viel. Iemand probeert iets één keer, het werkt niet — terecht — en de uitkomst blijft staan terwijl de wereld eronder wegschuift. Het tegengif is een waarneming mét tijdstempel: *op moment X had het status Y*, plus een vaste hertoets. **De houdbaarheid van een oordeel is korter geworden dan de herinnering eraan**, en dat is nieuw: vroeger was de eenheid een decennium en nu een paar maanden, niet lineair. Zie voorliggende oorzaak en nulmeting.

De duizend-keer-vraag *(the thousand-times question)* — Bij elke keuze: werkt dit ook, en is het betaalbaar, als ik het duizend keer doe? En dan nog eens duizend. Het is dezelfde vraag als altijd, maar hij geeft een ander antwoord — en dat andere antwoord is precies waar de stap van gereedschap naar organisatie begint.

De kosten van één ronde *(the cost of one round)* — Wat het kost om iets één keer te proberen: bouwen, meten, kijken wat eruit kwam. De grootheid die bepaalt of er tien rondes gedraaid worden of één, en dus of er gecontroleerd wordt of gegokt. Belangrijker dan snelheid, want hij geldt ook voor een goedkeuring, een proef of een offerte.

Frats *(onvertaald gelaten; het Engels heeft geen woord dat het draagt)* — Een klein, onverwacht of onhandig dingetje met grote gevolgen. Geen fout: het gaat niet mis, het lukt net niet. **Fouten kun je me-**

ten; fratsen zitten onder de radar — ze melden zich niet, ze liggen onder de drempel waarop iemand ingrijpt, en ze zijn pas zichtbaar als je keren naast elkaar legt die hetzelfde hadden moeten zijn. Een hallucinatie is er een. Ze maken het werk niet duur maar het vertrouwen, en daarmee zitten ze schaal in de weg. Het woord komt uit de sociale begeleiding; zie hallucinatie en nulmeting.

Gate — Een controle die werk tegenhoudt zolang er iets niet klopt. Geen waarschuwing die je kunt wegklikken maar een halte: de gate eindigt fout, en daardoor kan wat erachter zit niet door. Het woord komt uit de wereld van geautomatiseerd bouwen en is daar een veiligheidssysteem, geen deur — vandaar dat dit boek de Engelse term aanhoudt. Een gate die tegenhoudt zonder te zeggen hoe je verder komt, wordt uitgezet; een gate die alles doorlaat, bestaat niet.

Gereedschap *(tool)* — Een concrete handeling die je een agent geeft: een bestand lezen, een mail sturen, een betaling doen, een zoekopdracht uitvoeren. Een model zonder gereedschap kan alleen praten. Gereedschap is wat een gesprek in een handeling verandert, en daarom is het ook de plek waar je grenzen legt.

Groen *(green)* — Alle controles geslaagd. In dit boek een toestand en geen gevoel: het is groen of het is het niet, en alleen bij groen gaat er iets de deur uit.

Hallucinatie *(hallucination)* — Een antwoord dat klopt van vorm maar niet van inhoud: een verzonnen bron, een functie die niet bestaat, een getal dat nergens vandaan komt. Het gevaar zit hem in de zelfverzekerde toon, en de remedie is niet vertrouwen maar nakijken.

LLM (large language model) — Een groot taalmodel: getraind op enorme hoeveelheden tekst om telkens het volgende stukje te voorspellen. Dat het daarmee kan redeneren, plannen en programmeren is een gevolg, geen ontwerp.

MCP (Model Context Protocol) — Een afspraak over hóé een agent gereedschap krijgt aangeboden, zodat hetzelfde stuk gereedschap in elke omgeving werkt zonder dat iemand het opnieuw bouwt. Praktisch gevolg: sluit je één keer aan, dan komt daarmee alles binnen dat die afspraak spreekt.

Model — Het getrainde systeem dat tekst voorspelt en waarop alles hierboven draait. In dit boek is het een onderdeel, geen hoofdrolspeler: het model is inwisselbaar, wat je eromheen bouwt niet.

Multiplier — Werk dat het volgende werk goedkoper maakt, en dat dus twee keer telt. Een testsuite, een sjabloon, een generalisatieprompt. Ze zien er op de dag zelf uit als uitstel.

Niet in roeren *(leave it alone)* — Laat staan wat werkt, ook als je een verbetering ziet. Elke wijziging aan iets dat draait, kost een controle die je anders niet had hoeven doen — en de verbetering die je voor ogen had, weegt zelden op tegen de kans dat je iets stukmaakt dat niemand meer nakijkt. Geldt dubbel zodra er agenten meewerken: die hebben geen gevoel voor wat er al goed was, en vragen het ook niet. Je steekt geen hand in een draaiend vliegwiel.

Nul afhankelijkheden *(zero dependencies)* — De keuze om niets van buiten nodig te hebben. Kost meer werk vooraf en geeft daarna iets dat over vijf jaar nog bouwt.

Ontwerpvoorwaarde *(design condition)* — Een getal dat geen prestatie meer is maar een grens: onder deze waarde werkt de werkwijze, erboven niet. Het verschil met "sneller is prettiger" is dat je hem opschrijft en bewaakt, want hij gaat weg in stapjes die elk verdedigbaar zijn.

Prompt — De tekst waarmee je een model aan het werk zet. Geen vraag maar een opdracht die alles meedraagt wat nodig is om hem uit te voeren: wat er moet gebeuren, waarmee, binnen welke grenzen, en waaraan je zou zien dat het goed ging.

RAG (retrieval-augmented generation) — Relevante stukken uit je eigen documenten opzoeken en meesturen in de prompt, in plaats van te hopen dat het model het al weet. Het is de standaardmanier om een model iets te laten gebruiken wat het nooit gezien heeft.

Temperatuur *(temperature)* — De knop tussen voorspelbaar en gevarieerd. Laag als je hetzelfde antwoord wilt op dezelfde vraag, hoger als je wilt dat het model met iets nieuws komt.

Ticket — Eén afgebakende opdracht met een nummer, waar het werk aan hangt en waarin staat wat er besloten is. Het geheugen van de organisatie, en het enige wat een agent van de vorige kan overnemen.

Token — Het stukje tekst waarin een model rekent: ongeveer een lettergreep, soms een heel woord. Je betaalt per token en je contextvenster telt in tokens, dus het is tegelijk de rekeneenheid en de ruimtemaat.

Vastleggen *(capture)* — Een bevinding op schijf zetten op het moment dat hij valt, niet aan het eind. Wat niet op schijf staat, bestaat niet: het overleeft de sessie niet en dus ook de volgende agent niet.

Vloot *(fleet)* — Alle agenten die tegelijk voor je aan het werk zijn, als één geheel bekeken. Zodra je er meer dan een handvol hebt, is de vraag niet meer wat er één doet maar wat ze sámen doen — en daar heb je een scherm voor nodig, geen gevoel.

Voorliggende oorzaak *(upstream cause)* — Wat er vóór een frats of een storing zit en hem mogelijk maakt. Wie net te laat komt heeft misschien geen wekker; een model dat een bibliotheek verzint heeft geen lijst gekregen van wat er wél is. De term komt uit de begeleiding en hij draagt een werkwijze: het geval oplossen kost de volgende keer weer evenveel, de voorliggende oorzaak wegnemen laat een hele soort verdwijnen. Dat betekent teruggaan in plaats van vooruit, en dat is precies waarom het zo vaak niet gebeurt.

Voorwaarde *(condition)* — Wat waar moet zijn vóórdat een plan iets oplevert. De grootheid die bijna iedereen vergeet op te schrijven, waardoor een plan niet minder oplevert dan gehoopt maar nul.