Whitepaper

2026

De stille rem

Waar ligt de grens van low-code in het AI-tijdperk?

Infosupport
Solid Innovator

Voorwoord

Veel organisaties hebben applicaties die ooit met low-code zijn gebouwd. Dat was destijds een logische keuze: snel resultaat, weinig ontwikkelcapaciteit nodig en de business kon zelf aan de slag. Maar naarmate zulke applicaties belangrijker worden, verandert de rekensom. In gesprekken met organisaties die met low-code werken, komen steeds dezelfde drie problemen terug.

De kosten stijgen mee met het gebruik.

Licenties schalen met elke gebruiker, elke applicatie en elke omgeving, terwijl ook het beheer steeds meer capaciteit vraagt. Hoe succesvoller de applicatie, hoe hoger en onvoorspelbaarder de rekening.

Er is geen zicht meer op de achterliggende logica.

De kennis over hoe de applicatie in elkaar zit, zit bij een handjevol mensen en is nergens geborgd. Verdwijnt die kennis, dan blijft de applicatie draaien, maar durft niemand er nog iets aan te veranderen.

Het platform beperkt de flexibiliteit.

Nieuwe wetgeving, een koppeling met een kernsysteem of een aanpassing in het proces: steeds vaker is de vraag niet hoe je het oplost, maar of het binnen de opties van het platform past.

Een low-code transformatie lost precies deze drie problemen op. Door de bestaande logica expliciet te maken en gecontroleerd over te zetten naar een omgeving die je zelf in de hand hebt, verschuiven de kosten van platformafhankelijke licenties naar transparante ontwikkel- en beheerkosten. De kennis wordt vastgelegd in documentatie, architectuur en geautomatiseerde tests in plaats van in hoofden. En jij bepaalt weer hoe je applicaties zich ontwikkelen, in plaats van het platform. Dat is de kern van onze aanpak.

De reden dat dit nu een reëel alternatief is: AI verandert de rekensom van zo'n transformatie. Het uitzoekwerk dat vroeger maanden kostte, doen AI-agents in een fractie van die tijd.

Onze ervaring is dat het transformeren van bestaande low-code applicaties daardoor tot wel **70 procent** sneller kan. Een sprong in het diepe is het daarmee niet meer, maar een reeks beheersbare stappen waarbij je operatie gewoon doordraait.

In dit whitepaper lees je hoe je herkent dat low-code bij jouw organisatie zijn grens heeft bereikt, wat een transformatie concreet oplost en hoe je in vier stappen weer grip krijgt op je applicatielandschap.

Frank Thiele
Unitmanager, Info Support

Inhoudsopgave

In het kort	05
1. Drie signalen dat low-code zijn grens heeft bereikt	06
2. De succesparadox: waarom juist goed gebruikte low-code vastloopt	10
3. Wat een low-code transformatie oplost	12
4. In vier stappen weer grip op low-code	17
5. Onze ervaring: wat een transformatie met AI oplevert	23
6. Weet je waar jouw organisatie staat?	25
Over Info Support	26
Bronnen	27

In het kort

Voor organisaties die merken dat hun low-code applicaties meer kosten dan ze opleveren en die willen weten hoe ze daar weer grip op krijgen.

- **Low-code loopt zelden vast op techniek, maar op succes.**

  Hoe belangrijker een applicatie wordt, hoe zwaarder de eisen die eromheen gaan gelden, terwijl het platform daar nooit op is gekozen.

- **Drie signalen geven aan dat de grens is bereikt:**

  kosten die meeschalen met het gebruik, kennis die niet is geborgd en een platform dat bepaalt wat mogelijk is.

- **Transformeren is geen alles of niets.**

  Een deel van je landschap hoort prima thuis op low-code, een deel vraagt om meer controle. Hoofdstuk 3 geeft daar een indeling voor.

- **AI versnelt vooral het begrijpen.**

  Agents halen functionaliteit en verborgen aannames uit bestaande applicaties, genereren tests en documentatie en leveren een eerste versie van de nieuwe code. Mensen beoordelen, testen en accepteren.

- **De aanpak bestaat uit vier stappen:**

  impact scan, feature-extractie, migratie en doorontwikkelen. Het bestaande systeem blijft draaien en je ziet per onderdeel resultaat.

- **Begin met feiten.**

  Een impact scan geeft binnen enkele dagen een beeld van haalbaarheid, doorlooptijd, kosten en de businesscase.

1. Drie signalen dat low-code zijn grens heeft bereikt

Low-code is ooit om een goede reden gekozen. Er was veel vraag naar software, te weinig ontwikkelcapaciteit en een IT-backlog waarin afdelingen structureel achteraan aansloten. Low-code bracht snelheid en voor veel organisaties heeft die keuze zichzelf terugverdiend. Dit whitepaper is dan ook geen pleidooi tegen low-code, maar gaat over wat er gebeurt als een low-code applicatie belangrijker wordt dan waar het platform ooit voor is gekozen.

Low-code wordt namelijk niet van de ene op de andere dag een probleem. Meestal gaat het jaren goed en groeit het landschap ondertussen langzaam naar een kantelpunt: applicaties worden bedrijfskritisch, het aantal koppelingen neemt toe en veranderingen worden steeds ingewikkelder om door te voeren, terwijl de impact ervan steeds minder goed te overzien is. Wat eerst dagen duurde, duurt nu weken. Dat weegt extra zwaar in het AI-tijdperk, waarin innovaties in IT elkaar in hoog tempo opvolgen en een applicatielandschap dat niet mee kan bewegen direct een achterstand betekent. De onderstaande drie signalen zijn een indicatie dat low-code zijn werk heeft gedaan, maar dat je organisatie nu tegen de grenzen aanloopt van het model dat ooit zo goed paste.

Signaal 1: de kosten schalen mee met het gebruik, de vrijheid niet

Wat begint als een overzichtelijke platformkeuze, verandert bij groei in een kostenmodel dat steeds lastiger te sturen is. Licentiekosten stijgen met elke nieuwe gebruiker, elke extra applicatie, elke upgrade en soms zelfs met aparte test-, acceptatie- en productieomgevingen. Daarmee wordt low-code een variabele kostenpost: hoe succesvoller de adoptie, hoe hoger en onvoorspelbaarder de rekening.

Ook de personeelskosten schalen mee. Een groeiend en complexer applicatielandschap vraagt steeds meer capaciteit

voor beheer en doorontwikkeling. En omdat het ontwerp meestal nooit expliciet is vastgelegd, kost elke wijziging extra uitzoekwerk. Zo betaal je twee keer: aan het platform en aan het doorgronden van wat er in dat platform is gebouwd.

Dat steeds meer organisaties in deze dynamiek terechtkomen, blijkt uit de groei van de markt. Gartner verwacht dat de markt voor low-code application platforms in 2027 uitkomt op 16,5 miljard dollar, met een gemiddelde jaarlijkse groei van 16,3 procent tussen 2022 en 2027.[1] De vraag die daarbij hoort, is niet of je te veel betaalt, maar waarvoor je betaalt: nog voor versnelling, of vooral voor het recht om te blijven draaien wat er al staat?

Signaal 2: de kennis is niet geborgd en de impact van wijzigingen is niet te overzien

Low-code maakt ontwikkeling toegankelijk, maar maakt de logica tegelijk minder zichtbaar. Beslissingen zitten in visuele modellen, configuraties en workarounds die ooit een goede reden hadden, maar die achteraf moeilijk uit te leggen zijn. Zolang de consultant of interne expert die het heeft ingericht beschikbaar is, voelt dat beheersbaar. Maar die kennis is nergens geborgd. Verdwijnt ze door verloop, pensioen of een nieuwe rol, dan blijft de applicatie functioneren, terwijl niemand meer precies weet waarom die zo is ingericht. Professionals die zowel het platform beheersen als de specifieke inrichting van jouw organisatie snel doorgronden, zijn bovendien schaars.

Het probleem is groter dan persoonsgebonden kennis alleen. Low-code dwingt je niet om ontwerpkeuzes vast te leggen. Er is zelden een architectuur uitgedacht of zijn er domeinen gedefinieerd die bepalen welke functionaliteit waar thuishoort en welke data bij wie hoort. Het gevolg is dat zaken door elkaar gaan lopen: een aanpassing op de ene plek heeft onverwachte gevolgen op een andere plek. Omdat niemand met zekerheid kan zeggen wat de impact van een wijziging is, duren wijzigingen steeds langer of worden ze uitgesteld.

In essentie is dit technische schuld: verborgen kosten en extra werk die ontstaan door een snelle oplossing van toen. Volgens CISQ ging het in 2022 alleen al in de Verenigde Staten om 1,52 biljoen dollar aan kosten van software van onvoldoende kwaliteit.[2] Bij low-code ontstaat die schuld niet doordat er slechte code is geschreven, maar doordat het ontbrekende ontwerp het systeem steeds duurder en risicovoller maakt om veilig aan te passen.

Signaal 3: het platform bepaalt wat mogelijk is en hoe

Bij de start voelt een low-code platform als een versneller, omdat het veel keuzes uit handen neemt. Een architectuur of domeindefinitie levert het platform echter niet voor je. Wat het wel doet, is grenzen stellen. Bij nieuwe regelgeving, een integratie met een kernsysteem of een aanpassing in het proces is de eerste vraag steeds minder vaak "hoe lossen we dit op?" en steeds vaker "kan dit binnen het platform?". Je zit vast aan de opties die het platform biedt, in plaats van dat je zelf bepaalt hoe je applicaties waarde toevoegen.

Dat is een strategisch risico, zeker nu eisen rond weerbaarheid, controleerbaarheid en ketenverantwoordelijkheid zwaarder worden door regelgeving als NIS2 en DORA. Veranderende wetgeving vraagt om aanpassingen op momenten die jij niet kiest. Daar komt bij dat platformen steeds vaker AI-functionaliteit inbouwen, waardoor testbaarheid en uitlegbaarheid nog relevanter worden: is aantoonbaar wat er gebeurt, waarom het gebeurt en wat de impact is van een wijziging? Onderzoek naar accountability bij low-code en no-code AI benoemt juist die controleerbaarheid als risico.[3] Als het platform de grenzen bepaalt waarbinnen jij mag ontwikkelen, is low-code geen versneller meer, maar een rem.

LOW-CODE TRANSFORMATIE MET AI

Zelfscan: waar staat jouw low-code landschap?

Kruis per stelling aan: ja, deels of nee.

1 Onze licentie- en beheerkosten zijn de afgelopen twee jaar sterker gestegen dan het gebruik.

KRUIS AAN ☐ JA ☐ DEELS ☐ NEE

2 Er zijn maximaal twee mensen die de kernlogica van onze belangrijkste low-code applicatie echt kennen.

KRUIS AAN ☐ JA ☐ DEELS ☐ NEE

3 We kunnen niet binnen een dag aantonen welke bedrijfsregels waar zijn vastgelegd.

KRUIS AAN ☐ JA ☐ DEELS ☐ NEE

4 Bij een wijzigingsverzoek is de eerste vraag of het binnen het platform past.

KRUIS AAN ☐ JA ☐ DEELS ☐ NEE

5 Een significante wijziging kost ons eerder weken dan dagen.

KRUIS AAN ☐ JA ☐ DEELS ☐ NEE

6 Er is geen geautomatiseerde testset die bij elke wijziging draait.

KRUIS AAN ☐ JA ☐ DEELS ☐ NEE

0–2 **Twee stellingen of minder**
Je situatie is waarschijnlijk beheersbaar. Blijf de ontwikkeling volgen.

3+ **Drie stellingen of meer**
Je landschap vraagt aandacht. Een gerichte analyse is verstandig vóór het volgende project start.

Zelfscan: hoe scoort jouw organisatie?

Herken je twee stellingen of minder, dan is je situatie waarschijnlijk beheersbaar en volstaat het om de ontwikkeling te blijven volgen. Herken je er drie of meer, dan vraagt je landschap aandacht en is een gerichte analyse verstandig voordat het volgende project op het platform start.

2. De succesparadox: waarom juist goed gebruikte low-code vastloopt

Dat low-code op een gegeven moment begint te schuren, betekent zelden dat de keuze destijds fout was. Integendeel: juist organisaties die er succesvol mee waren, lopen het vaakst tegen grenzen aan. Dat is de succesparadox van low-code. Hoe beter een applicatie landt in de organisatie, hoe meer mensen erop gaan leunen. Er komen koppelingen bij, uitzonderingen worden ingebouwd en processen die eromheen liepen, worden erin getrokken. Als een applicatie zo uitgroeit van handig naar bedrijfskritisch, wordt niet alleen het resultaat belangrijk, maar ook alles wat eromheen zit: testbaarheid, security, compliance en de vraag wie eigenaar is van alle onderliggende keuzes.

Daar wringt het, zoals bij signaal 2 al bleek. Low-code maakt bouwen toegankelijk, maar dwingt je niet om ontwerpkeuzes helder vast te leggen. Maatwerk wordt door de business zelf gebouwd, door mensen zonder IT-achtergrond die vaak niet op het belang van domeingrenzen, data-eigenaarschap en testbaarheid zijn gewezen. Het resultaat is meer maatwerk in plaats van minder, alleen dan zonder ontwerp. Vaak is low-code zelf dus niet het probleem, maar het gebruik ervan zonder begeleiding van software engineers die het ontwerp en de basis kunnen toetsen. Omdat die begeleiding meestal ontbreekt, zien we dezelfde vier valkuilen terugkomen zodra organisaties iets aan de situatie willen doen.

Vier valkuilen zodra organisaties iets aan de situatie willen doen

Dezelfde reflexen komen steeds terug. Ze voelen logisch, maar vergroten het risico.

1. Opnieuw beginnen met een big bang

Als de pijn groot wordt, is de verleiding groot om helemaal opnieuw te beginnen: een nieuwe omgeving, een nieuwe architectuur, een groot programma, liefst in een keer goed. Maar bedrijfskritische low-code applicaties kun je zelden pauzeren. De applicatie moet online blijven, de wetgeving loopt door en de business verandert ondertussen. Een big bang belooft overzicht, maar levert vaak juist extra risico: langere doorlooptijd, meer afhankelijkheden en pas laat zichtbare resultaten.

2. Letterlijke vertaling

Een andere reflex is het bestaande een-op-een overzetten, netter en met een overzichtelijke structuur. In low-code is dat extra verleidelijk, omdat het bestaande model al visueel is en dus makkelijk kopieerbaar voelt. Maar als je de oude complexiteit letterlijk meeneemt, moderniseer je vooral het jasje. Je houdt dezelfde fricties aan de binnenkant, met een nieuw prijskaartje.

3. Tooling zonder ontwerp

Low-code wordt vaak verkocht met de suggestie dat iedereen kan bouwen. In theorie klopt dat, want de drempel ligt lager. Maar in de praktijk blijkt dat goed bouwen iets anders is dan snel bouwen. Zonder bewuste keuzes over domeinen, data-eigenaarschap en grenzen tussen onderdelen ontstaat stapeling: extra logica in workflows, uitzonderingen en koppelingen die groeien per project. Dan is het platform niet het probleem, maar het ontbreken van ontwerp. Het resultaat voelt eerst efficiënt, maar wordt later moeilijk te testen, moeilijk te veranderen en nog moeilijker te blijven overzien.

4. Pilot zonder vangnet

Veel organisaties beginnen verstandig met een pilot. Het probleem ontstaat wanneer die pilot geen vangnet heeft: geen duidelijke meetlat, geen afspraken over beheer en eigenaarschap en geen plan voor wat er gebeurt als de pilot aanslaat. Dan wordt een succesvolle pilot onbedoeld het begin van een nieuw kernsysteem, zonder dat de kaders zijn meegegroeid. Precies dan ontstaat de paradox: het succes bewijst de waarde van de applicatie, maar maakt terugsturen steeds lastiger.

Low-code zorgt voor snelheid in een wereld met weinig ontwikkelcapaciteit. Maar zodra de applicaties de ruggengraat van processen worden, gaat het niet alleen meer om bouwen, maar om regie. En die regie vraagt om expliciete keuzes die in de beginfase vaak niet zijn gemaakt. Het goede nieuws is dat je die keuzes alsnog kunt maken, zonder alles opnieuw te bouwen.

3. Wat een low-code transformatie oplost

Veel organisaties schuiven modernisering lang voor zich uit. De reden is begrijpelijk. Overstappen heeft een hardnekkig beeld van een jarenlang traject met veel risico en hoge kosten, dat pas waarde oplevert als alles is afgerond. Toch klopt dat beeld steeds minder. Modernisering hoeft niet te betekenen dat je alles opnieuw bouwt of in een keer overstapt. Het is de afgelopen jaren concreter, sneller en beter beheersbaar geworden, omdat we het proces anders benaderen en omdat AI een groot deel van het zware voorwerk versnelt.

Drie problemen, drie antwoorden

Een transformatie is alleen zinvol als die een antwoord geeft op de drie signalen uit hoofdstuk 1. Daarom eerst de vraag wat er na afloop concreet anders is.

Kosten die meeschalen met gebruik worden kosten die je zelf stuurt.

Door functionaliteit uit het platform te halen, betaal je niet langer per gebruiker, applicatie en omgeving, maar voor transparante ontwikkeling en beheer. Je maakt bewuste keuzes over waar je in investeert, in plaats van dat de licentiefactuur die keuze voor je maakt.

Kennis die in hoofden zit, wordt kennis die in het systeem zit.

De bestaande bedrijfslogica wordt expliciet gemaakt en vastgelegd in begrijpelijke documentatie en geautomatiseerde tests. Daar komt een architectuur bij met heldere domeinen, zodat duidelijk is welke functionaliteit waar thuishoort en wat de impact is van een wijziging. Het vertrek van een sleutelpersoon of het wisselen van een externe partij is daarmee geen risico meer voor de continuïteit.

Een platform dat bepaalt wat kan, maakt plaats voor een landschap waarin jij dat bepaalt.

Nieuwe wetgeving, een integratie of een procesaanpassing wordt weer een ontwerpvraag in plaats van een platformvraag. Het resultaat is een uniform applicatielandschap dat mee kan bewegen met de organisatie, met aantoonbare kwaliteit en security.

Transformeren betekent niet dat je alles moet schrappen

Regie terugkrijgen betekent niet dat alle low-code software ineens weg moet. Vaak kunnen low-code applicaties voor standaard- of ondersteunende processen gewoon blijven. Functionaliteit die dicht op de kernprocessen zit, veel uitzonderingen kent en streng moet voldoen aan eisen rond auditability en security, vraagt om meer controle en testbaarheid. Dat is meestal makkelijker te organiseren in high-code, met expliciete architectuurkeuzes en flexibiliteit. Tegelijk kan low-code of een standaardpakket nog steeds een goede plek zijn voor onderdelen die ondersteunend zijn, relatief stabiel, of sterk lijken op wat veel organisaties nodig hebben.

Het gaat er dus om per applicatie te bepalen waar die thuishoort. Vier vragen geven daarbij de doorslag: hoe dicht zit de functionaliteit op het proces waarmee je je onderscheidt, hoeveel uitzonderingen kent die, hoe zwaar zijn de eisen aan aantoonbaarheid en hoe vaak moet er iets veranderen? Op basis daarvan kun je je applicaties in vier groepen verdelen.

LOW-CODE TRANSFORMATIE MET AI

Drie routes voor je low-code landschap

Bepaal per applicatie welke route past.

Behouden

KENMERKEN

- Bestaande functionaliteit voldoet
- Stabiele basis, beheersbare kosten en eisen
- Uitbreiding kan buiten het platform

PASSENDE ROUTE

Laat passende functionaliteit in low-code staan. Bouw nieuwe functionaliteit waar nodig buiten het platform en koppel die aan het bestaande.

Transformeren

KENMERKEN

- Bedrijfskritisch en veel uitzonderingen
- Hoge wijzigingsdruk
- Strenge eisen aan security, audit en testbaarheid

PASSENDE ROUTE

Zet functionaliteit gecontroleerd over naar een high-code omgeving met expliciete architectuurkeuzes en geautomatiseerde tests.

Uitfaseren

KENMERKEN

- Lage benutting
- Overlappende functionaliteit
- Goed gedekt door een standaardpakket

PASSENDE ROUTE

Zet de applicatie uit of vervang die door een passende oplossing. Migreer de data die je nodig houdt.

Bepaal per applicatie waar die thuishoort
Kijk naar kernproces, uitzonderingen, compliance-eisen, wijzigingsdruk en benutting.

Een derde route

Daarnaast zien we in de praktijk een derde route: organisaties blijven bij hun bestaande low-code landschap voor wat er is, maar kiezen voor uitbreiding bewust een ander fundament. Daarmee voorkom je dat het platform automatisch de standaard blijft voor alles wat nog komt. Door gerichte keuzes te maken over waar een transformatie echt nodig is, bespaar je veel tijd en onnodig werk.

AI-agents als versneller, niet als vervanging

Het grootste verborgen werk in modernisering is vaak niet het bouwen, maar het begrijpen. Wat doet de applicatie precies, welke functionele eisen zaten erachter en kloppen die vandaag nog met de praktijk? Dit reverse engineeren kostte vroeger maanden. AI-agents versnellen dit werk door code, modellen, logs, configuraties en documentatie te analyseren en samen te vatten. Daarnaast doen ze voorstellen voor herstructurering of herbouw. McKinsey beschreef in 2024 dat AI-gedreven modernisering van verouderde software kan leiden tot **40 tot 50 procent** versnelling en ongeveer **40 procent** kostenreductie.[4] De ervaring van Info Support is dat er inmiddels meer mogelijk is, tot wel **70 procent** versnelling, doordat gespecialiseerde agents niet alleen analyseren, maar ook een groot deel van de conversie, de testdekking en de documentatie voor hun rekening nemen.

Even belangrijk is wat AI niet oplost. Agents nemen het denkwerk niet over, maar ondersteunen bij het uitzoekwerk.

AI-agents als versneller, mensen aan het stuur

Agents nemen het uitzoekwerk over. Het denkwerk, de keuzes en de acceptatie blijft mensenwerk.

Wat agents goed doen

- Bestaande functionaliteit en logica uit code, modellen en configuratie halen en vertalen naar begrijpelijke bouwstenen
- Aannames zichtbaar maken die nergens zijn vastgelegd en signaleren wat ontbreekt of tegenstrijdig is
- Documentatie en geautomatiseerde tests genereren waar die er nooit zijn geweest
- Een eerste versie van de nieuwe code beoordelen en opleveren binnen gezette kaders qua architectuur, governance en veiligheidseisen

Wat mensen blijven doen

- De eerste versie van de code die agents opleveren beoordelen en waar nodig bijsturen
- Beoordelen of iets een functionele eis is, een bewuste keuze of een bug die jarenlang meeliep
- Bepalen wat mee moet naar de nieuwe situatie en wat je bij deze gelegenheid opruimt
- Valideren met de business of de gereconstrueerde logica klopt met de praktijk van vandaag
- Testen, beoordelen en accepteren, zodat de verantwoordelijkheid niet bij het model komt te liggen

Die tweedeling is geen slag om de arm. AI kan bedrijfslogica net verkeerd interpreteren of code opleveren die er goed uitziet en functioneel net anders uitpakt. Daarom zijn kwaliteitsborging en duidelijke documentatie onderdeel van de aanpak en niet iets dat achteraf komt: dezelfde testset draait op de oude en de nieuwe omgeving, zodat je bij elke stap kunt aantonen dat het gedrag gelijk blijft.

4. In vier stappen weer grip op low-code

Als je besluit dat je weer regie wilt over je low-code landschap, is de eerste vraag meestal niet technisch, maar praktisch: wat gebeurt er dan concreet? Het helpt om dit niet te zien als een groot transformatietraject, maar als vier overzichtelijke stappen. Elke stap verkleint risico, maakt keuzes explicieter, levert tussentijds al waarde op en heeft een duidelijk eindpunt dat de basis vormt voor de volgende stap.

LOW-CODE TRANSFORMATIE MET AI

In vier stappen weer grip op low-code

AI-agents versnellen de analyse, tests, documentatie en conversie. Mensen houden de regie.

De vier stappen in één beeld: elke stap heeft een duidelijk doel en een concreet resultaat dat de basis vormt voor de volgende.

Stap 1. Impact scan: wat doen je low-code applicaties echt en wat kost het je?

Doel
Snel inzicht in de haalbaarheid en impact van een transformatie, voordat je investeert.

Wat er gebeurt
In enkele dagen brengen we samen met jouw specialisten in kaart wat de applicaties in de praktijk doen, welke processen erop leunen en waar de technische complexiteit, afhankelijkheden, workarounds en koppelingen zitten. AI-agents ondersteunen deze analyse door de applicaties, configuraties en documentatie geautomatiseerd te doorzoeken; daardoor is in dagen mogelijk wat voorheen weken kostte. We onderzoeken welke gevolgen een transformatie heeft voor processen en bedrijfsvoering en we brengen het kostenbeeld in kaart: van licenties en beheerlast tot de doorlooptijd van wijzigingen en de inzet van schaarse experts. Afhankelijk van de applicatie beoordelen we ook of er een architectuur en domeinindeling nodig is die er nu nog niet is.

Wat je krijgt
Een onderbouwd beeld van welke onderdelen stabiel zijn en voldoende waarde leveren en welke juist de doorontwikkeling remmen door platformbeperkingen, oplopende kosten of lastig onderhoud. Daarbij hoort een indicatie van haalbaarheid, doorlooptijd en kosten plus een eerste businesscase: wat kost doorgaan op de huidige weg ten opzichte van transformeren en wanneer verdient de investering zich terug. Je krijgt bovendien een schets van het eindplaatje, zodat zichtbaar is hoe een uniform applicatielandschap er na de transformatie uitziet. Geen offerte en geen verplichting.

Wat mee gaat naar stap 2
Een gekozen eerste onderdeel. Bij een groot landschap begin je niet met de hele applicatie, maar met een afgebakend onderdeel dat als proof of concept dient voor de stappen 2 en 3.

Stap 2. Feature-extractie en ontwerp: zichtbaar maken wat de applicatie doet

Doel
Expliciet maken wat de applicatie daadwerkelijk doet en het fundament leggen voor de nieuwe omgeving.

Wat er gebeurt
AI-agents halen functionaliteit en achterliggende logica uit de applicatie, modellen, configuraties en documentatie. Daarbij komen ook aannames en functionele eisen boven water die nergens expliciet zijn vastgelegd. Business en engineers beoordelen vervolgens samen welke uitgangspunten nog kloppen, welke aanpassing nodig hebben en welke mogelijk wijzen op fouten in de bestaande applicatie. Parallel leggen we de referentiearchitectuur en de technische en functionele kaders voor de nieuwe omgeving vast. Waar het low-code deel nooit een architectuur of domeinindeling heeft gehad, definiëren we die nu: welke domeinen er zijn, waar de grenzen liggen en wie eigenaar is van welke data. We doen dit samen met jouw team, zodat de kennis direct in je eigen organisatie landt.

Wat je krijgt
Een expliciete beschrijving van je eigen bedrijfslogica, een referentiearchitectuur met heldere domeingrenzen en een gevalideerde scope voor het eerste onderdeel. Die beschrijving houdt haar waarde, ook als je daarna besluit om voorlopig niets te doen.

Wat mee gaat naar stap 3
De beschreven functionaliteit, de kaders en een testset die het huidige gedrag van het onderdeel vastlegt.

Stap 3. Migratie: stap voor stap, zonder de winkel te sluiten

Doel
Functionaliteit verplaatsen zonder de operatie te verstoren.

Wat er gebeurt
Het bestaande systeem blijft draaien terwijl de nieuwe omgeving wordt opgebouwd en getest. Het eerste afgebakende onderdeel wordt opnieuw gebouwd op een plek waar je meer controle hebt en teruggekoppeld aan het bestaande systeem. Oude en nieuwe functionaliteit bestaan tijdelijk naast elkaar en dezelfde tests draaien op beide omgevingen, zodat continu vaststaat dat het gedrag gelijk blijft. Werkt het eerste onderdeel zoals verwacht, dan volgt het volgende. Tegelijkertijd trainen we jouw ontwikkelaars in de nieuwe architectuur, de werkwijze en AI-ondersteund ontwikkelen, zodat zij vanaf het begin meebouwen en niet pas aan het einde het stokje overnemen.

Wat je krijgt
Een landschap dat geleidelijk verschuift, zonder moment waarop alles tegelijk om moet, met per onderdeel aantoonbaar resultaat en een team dat de nieuwe werkwijze beheerst.

Stap 4. Doorontwikkelen: grip op je processen, ook na de transformatie

Doel

Voorkomen dat je opnieuw afhankelijk wordt van een enkel platform, een enkele externe leverancier of een beperkt aantal sleutelpersonen.

Wat er gebeurt

De basis die in de vorige stappen is gelegd, wordt de standaard voor alles wat nog komt: begrijpelijke en gedocumenteerde applicaties, geautomatiseerde tests, een schaalbare architectuur en duidelijke kaders voor doorontwikkeling. Die basis maakt het mogelijk om gecontroleerd op te schalen als het aantal gebruikers, processen of koppelingen groeit. Geautomatiseerde tests bewaken niet alleen of bestaande functionaliteit blijft werken, maar maken het ook mogelijk om nieuwe wijzigingen gecontroleerd door te voeren.

Wat je krijgt

Een applicatielandschap dat weer mee kan bewegen met de organisatie, met voorspelbare kosten en aantoonbare kwaliteit. Voor sommige organisaties ligt het eigenaarschap daarna primair bij het eigen team, dat tijdens het traject heeft meegebouwd en kennis heeft opgedaan. Voor andere organisaties is ontzorging een logischere keuze. Dan kan Info Support het beheer en de doorontwikkeling geheel of gedeeltelijk overnemen. Kennisopbouw loopt in beide gevallen vanaf stap 2 mee, ondersteund door gerichte trainingen vanuit onze IT Academy, zoals Agentic engineering: specify and generate in de .NET- of Java-variant.

5. Onze ervaring: wat een transformatie met AI oplevert

De aanpak uit hoofdstuk 4 is geen theorie. Info Support paste dezelfde werkwijze eerder toe in een modernisatietraject, waarbij een verouderd applicatielandschap werd omgezet naar een moderne, flexibele stack. Dat was geen low-code transformatie, maar het patroon is hetzelfde: de codebase werd opgeknipt in behapbare delen en AI-modellen zetten die delen om naar de gewenste programmeertaal. Naast de codeomzetting genereerde AI automatisch tests en functionele documentatie die ook voor niet-technische stakeholders begrijpelijk was. Het team schatte vooraf dat het project **25 tot 30 procent** sneller zou gaan; achteraf bleek de tijdwinst op te lopen tot **50 procent**. Met de gespecialiseerde agents die we sindsdien inzetten voor analyse, documentatie, tests en conversie, is inmiddels tot **70 procent** versnelling mogelijk.

Hetzelfde patroon zien we bij JoinData, dat veilige datadeling in de agrarische keten faciliteert. Voor een deel van de applicaties was de organisatie afhankelijk van schaarse externe specialisten, waardoor wijzigingen langer duurden dan gepland en de kosten minder voorspelbaar werden. Info Support migreerde deze applicaties met behulp van AI, iteratief en met eerst een enkele applicatie voordat de rest volgde. Het resultaat: de doorlooptijd van wijzigingen ging van drie tot vier weken naar drie tot vier dagen, de afhankelijkheid van externe specialisten verdween, de onderhoudskosten werden lager en voorspelbaarder en het eigen team kreeg ruimte om zelf door te ontwikkelen.

Renze van de Kuilen, Team Lead Operations en Support bij JoinData, over de uitkomst:

> *“Nu ligt de regie weer bij ons eigen team, wat ons meer ruimte voor innovatie geeft.”*

Die combinatie van versnelling in het traject en een structureel kortere doorlooptijd van wijzigingen daarna, is waar de businesscase van een low-code transformatie op rust. De impact scan maakt die businesscase voor jouw situatie concreet.

6. Weet je waar jouw organisatie staat?

Het low-code landschap van je organisatie draait waarschijnlijk nog steeds. De echte vraag is of je het ook nog kunt aansturen en of je de consequenties van aanpassingen kunt overzien. De zelfscan in hoofdstuk 1 geeft daar een eerste antwoord op. Herken je drie of meer stellingen, of twijfel je over wat je low-code applicaties de komende jaren gaan kosten, dan is een impact scan de logische volgende stap. Geen verplichting en geen offerte, maar duidelijk inzicht in haalbaarheid, doorlooptijd, kosten en de businesscase. Blijkt uit de scan dat een transformatie kansrijk is, dan is de vervolgstap een pilot waarin je een afgebakend onderdeel overzet. Werkt dat zoals verwacht, dan besluit je op basis van resultaat of de rest volgt.

Plan een **impact scan** en weet binnen enkele dagen waar je staat.

infosupport.com/services/low-code-transformatie-met-ai

Joop Snijder
Head of AI
joop.snijder@infosupport.com

Frank Thiele
Unitmanager
frank.thiele@infosupport.com

Over Info Support

Info Support biedt end-to-end IT-oplossingen die jouw organisatie versterken. Onze ruim 500 experts nemen verantwoordelijkheid voor het volledige traject. Jouw kritische IT in veilige handen, gebouwd op vakmanschap, versterkt met AI in de hele lifecycle.

Wij helpen organisaties om businessvraagstukken te vertalen naar AI-functionaliteit die geïntegreerd wordt in bestaande applicaties, dataplatformen en bedrijfsprocessen. Met een beproefde aanpak en hoogwaardige trainingen zorgen we ervoor dat jouw organisatie klaar is voor de toekomst en optimaal kan excelleren en innoveren.

- **AI**
- **Cloud**
- **Data**
- **Software engineering**
- **Training**

Bronnenlijst

In dit whitepaper combineren we praktijkervaringen van Info Support met inzichten uit onderzoek en vakpublicaties. Hieronder vind je de vijf geraadpleegde bronnen. Ze bieden onderbouwing bij de genoemde cijfers en meer achtergrond over low-code, technische schuld en modernisering met AI.

[1] Gartner, Risk and Opportunity Index: Low-Code Application Platforms.

https://www.gartner.com/en/documents/5459763

[2] Consortium for Information and Software Quality (CISQ), The Cost of Poor Software Quality in the US: A 2022 Report.

https://www.it-cisq.org/the-cost-of-poor-quality-software-in-the-us-a-2022-report/

[3] Somer, P. e.a., Algorithmic Accountability of Low-Code/No-Code Artificial Intelligence: A Literature Review, 2025.

https://www.researchgate.net/publication/393967156_Algorithmic_Accountability_of_Low-CodeNo-Code_Artificial_Intelligence_A_Literature_Review

[4] McKinsey, AI for IT modernization: faster, cheaper and better, 2024.

https://www.mckinsey.com/capabilities/quantumblack/our-insights/ai-for-it-modernization-faster-cheaper-and-better

Relevante solutions bij dit onderwerp

Low-code transformatie met AI
Voor het analyseren en transformeren van bestaande low-code applicaties.

www.infosupport.com/services/low-code-transformatie-met-ai

Software modernisatie met AI
Voor verouderde technologie zoals COBOL, Silverlight en oudere .NET- en Java-versies.

www.infosupport.com/services/software-modernisatie-met-ai

De route naar digitale autonomie
Voor het verkleinen van de afhankelijkheid van leveranciers en platformen.

www.infosupport.com/services/de-route-naar-digitale-autonomie

Managed Cloud Platform Services
Voor het beheer en de doorontwikkeling na oplevering.

www.infosupport.com/services/managed-cloud-platform-services

Trainingen van onze IT Academy
Waaronder Agentic engineering: specify and generate.

training.infosupport.com

Hoofdkantoor Nederland
Kruisboog 42
3905 TG Veenendaal

T +31 (0)318 552 020

Hoofdkantoor België
Generaal de Wittelaan 17 | bus 30
2800 Mechelen

T +32 (0)15 28 63 70
www.infosupport.com