LOW-CODE VS. RPA

het verschil tussen de gebruikersapplicatie & de virtuele applicatiegebruiker

Klaas-Jan Molendijk
Partner & adviseur bij BCE

Met dank aan BCE-collega's Ivo Beniest, Joost van Bilsen, Maxine Hilhorst, Thomas van der Knijff Lisa Rijksen en Eric van der Wolk.

Februari 2020

Inhoudsopgave

INLEIDING

De wereld van digitale transformatie bevat een ongeëvenaard aantal buzzwords. Afhankelijk van wie je spreekt moet je als organisatie met AI, low-code, RPA, VR, blockchain of iets anders aan de slag. De verschillen lijken soms wat arbitrair, maar zijn allesbehalve dat. In dit artikel zoomen we specifiek in op low-code-applicatieontwikkeling versus Robotic Process Automation (RPA¬). Sommigen zien deze thema's als concurrenten. Anderen zien RPA als 'next step' voor low-code… of andersom. 1 ding is duidelijk: er is een hoog ik-heb-de-klok-horen-luiden-maar-ik-weet-niet-waar-de-klepel-hangt-gehalte. Dit artikel heeft als doel om RPA en low-code in kristalhelder en praktijkgericht perspectief te zetten.

"Ook al is IT jouw vakgebied niet, anno 2020 is/wordt dit toch wel jouw vakgebied! IT is te belangrijk om nog langer af te doen als 'iets technisch'. "

Anekdote

Ik maak dankbaar gebruik van een anekdote van een BCE-collega die een robotje heeft gebouwd die zijn vriendin iedere ochtend een -- naar ik aanneem: lief -- WhatsApp-berichtje stuurt. WhatsApp is een bestaande applicatie; de softwarerobot gebruikt deze bestaande applicatie. Eenvoudiger dan dat is het verschil tussen applicaties en softwarerobots niet uit te leggen.

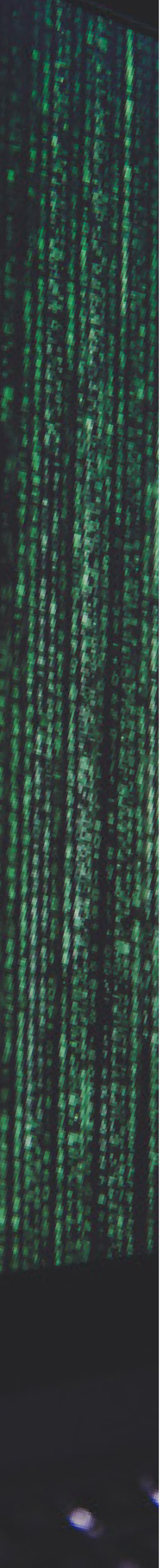
INZICHT 1

Een applicatie verschilt fundamenteel van een softwarerobot

Ja, het zijn beide softwareoplossingen, maar (low-code-)applicaties en softwarerobots zijn wezenlijk andere dingen. Een applicatie – volgens de Wikipedia-definitie een programma voor eindgebruikers – bestaat uit schermen, knoppen, tekstlabels, enzovoort. Het is iets wat je ziet. Het is iets waar je in/mee werkt. Denk aan Windows, een Mijn-omgeving, jouw e-mailprogramma, Photoshop, het intranet, PowerPoint, een specifiek dossiersysteem, … enzovoort. Het zijn allemaal applicaties.

Een softwarerobot is daarentegen een virtuele medewerker die, net als jij, in applicaties werkt. Hij opent net als jij schermen, klikt op buttons, zoekt informatie op internet, maakt bestanden aan, stuurt e-mails, … enzovoort. Het maakt niet uit wat voor applicatie het is. Het maakt ook niet uit waarmee/hoe de applicatie gebouwd is. Misschien is de applicatie gebouwd in een low-code-platform zoals Mendix of OutSystems. Of misschien is het een heel oude mainframe-applicatie. Of het is de Wordpress-website die jouw buurman heeft gemaakt. Kortom, je werkt niet in een robot, maar een robot werkt in/met applicaties (voor jou).

Oneliners als “ach, het zijn maar buzzwords, het maakt niet echt uit” of “RPA is the next step voor low-code” (of vice versa) slaan de plank mis. De thema’s lijken niet eens op elkaar, anders dan het feit dat het (complementaire) instrumenten zijn om digitaal te transformeren.



INZICHT 2

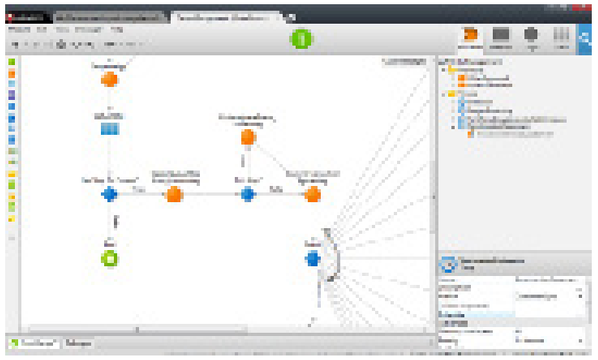
Heavy-code applicaties vs. low-code applicaties; het maakt voor een robot helemaal niet uit

Het *visuele* eindresultaat van heavy-code en low-code is gelijk. Het *technische* eindresultaat van low-code is ook heavy-code. Als jij in een low-code-platform bijvoorbeeld een scherm modelleert met daarop een aantal velden met daarachter een eenvoudige workflow, dan is er uiteindelijk ook gewoon sprake van allerlei heavy-code, zoals HTML, Javascript en CSS en nog wat andere technische zaken. Er zit in een low-code-platform namelijk een vertaler ('compiler') die de low-code-modellen omzet naar iets wat bijvoorbeeld browsers en andere systemen snappen. Dat technische eindresultaat is zeker niet precies gelijk aan hoe jij het zelf geprogrammeerd zou hebben, maar de functionaliteit/werking komt wel overeen.

Voor lezers die nieuw zijn in de jargon die dit artikel gebruikt, is voorgaande toelichting mogelijk toch nog te cryptisch. Daarom een heel concreet voorbeeld: als je een keer over iemands schouder meekijkt naar een beeldscherm, dan zie je daarop een of meerdere applicaties. Je ziet *niet* hoe deze gebouwd zijn. Grootste potentiële verschil is dat het proces van totstandkoming van die applicaties anders is. Waar heavy-code uitgaat van veel ambachtelijk coderen (het echte code schrijven zoals in afbeelding 1) gaat low-code uit van vooral modelleren (zie afbeelding 2) en alleen nog coderen als het echt moet.

Afbeelding 1:
Heavy-code (gebaseerd op Vue.js)

Afbeelding 2:
Low-code (gebaseerd op OutSystems)



Let op: van afbeelding 1 en afbeelding 2 merkt een softwarerobot al helemaal niets, want hij werkt op/in het zichtbare eindresultaat van heavy-code of low-code.

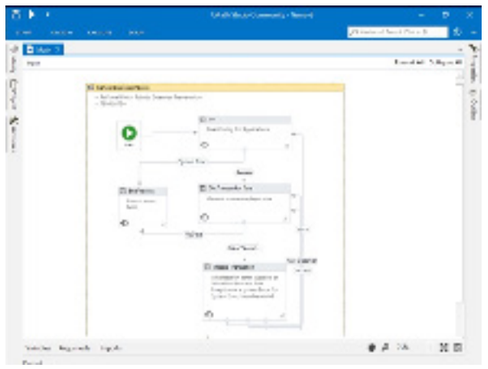
Het goede nieuws is dat het voor een robot allemaal niet uitmaakt; hij werkt vrolijk in alle applicaties ongeacht of deze via low-code of heavy-code tot stand zijn gekomen. Want alles wat zichtbaar is op het beeldscherm, is bruikbaar voor de softwarerobot.

INZICHT 3

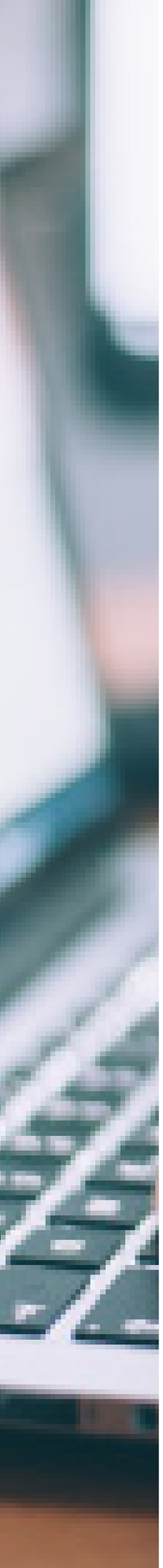
‘Low-code’ is een containerbegrip waarbinnen ook RPA valt

Low-code is een misleidende afkorting. Wat we met elkaar bedoelen is low-code *applicatieontwikkeling*. Gebruikersapplicaties zijn het eindresultaat van deze ontwikkeling. Echter zegt het begrip ‘low-code’ vooral iets over het creatieproces en nog niet zo zeer iets over het product dat wordt gemaakt. Low-code is namelijk net zo goed van toepassing op *robotontwikkeling*. In de laatste paar jaren is ook het bouwen van zo’n virtuele medewerker via modelleren toegankelijk gemaakt (zie afbeelding 3). ‘Vroeger’ moest je ook voor softwarerobots nog erg veel coderen. Inmiddels is het uitzonderlijk eenvoudig en bouw je softwarerobots in uren tot weken.

Afbeelding 3:
Bouw van een softwarerobot (in UiPath)



“ ‘Vroeger’ moest je ook voor software-robots nog erg veel coderen. Imiddels is het uitzonderlijk eenvoudig en bouw je softwarerobots in uren tot weken.”



INZICHT 4

RPA staat lager op de ‘digitale transformatie ladder’ dan low-code-applicatieontwikkeling

Als je als organisatie in de positie bent om te kiezen uit meerdere oplossingsrichtingen voor digitale transformatie, dan is RPA altijd je allerlaatste keuze. Waarom? Als je RPA je vertrekpunt maakt, dan ga je werk zoals dat nu georganiseerd is automatiseren. Je doet dan wat men nu doet, maar verschuift het werk van een medewerker van vlees & bloed naar een medewerker van bits & bytes. Terwijl digitale transformatie niet primair zou moeten gaan over automatiseren, maar over transformeren. Het echt anders doen.

Om het van filosofisch weer praktisch te maken: stel dat je een organisatie bent waar dagelijks talloze e-mails binnenkomen. Laten we zeggen bestellingen die handmatig worden verwerkt door medewerkers. Als je RPA als vertrekpunt neemt, dan kies je er voor dat die e-mails door softwarerobots worden verwerkt. Zo, je bent klaar… en realiseert mogelijk een fantastische business case.

Echter het gevoel dat je waarschijnlijk bekruipt is: maar kan dit proces sowieso niet anders? Wie mailen ons eigenlijk? Waarom sturen zij ons mails? Is dat wel het ideale proces voor die doelgroep? Enzovoort. Een naadloos met het back-office-systeem geïntegreerde Mijn-omgeving is vermoedelijk veel krachtiger… en daar komt ineens low-code-applicatieontwikkeling om de hoek kijken.

Pas vertelde iemand tijdens een sessie over RPA: “ik overweeg om een nieuwe onderneming neer te zetten en ik wil kijken wat RPA voor mij kan betekenen”. De open, leergierige houding wordt beslist gewaardeerd, maar zeker in een zgn. greenfield – waarin je niet/nauwelijks belemmerd wordt door legacy – is RPA oprecht wel het aller-aller-aller-laatste waar je aan moet denken. Er zijn uitzonderingen, maar wat ons betreft is een prima conclusie dat de toegevoegde waarde van RPA en mate van veroudering van een IT-landschap sterk gecorreleerd zijn.

INZICHT 5

RPA is krachtig als *the last mile of automation*

RPA wordt ook wel the last mile of automation genoemd. Onze praktische interpretatie daarvan is dat het zaken betreft die (1) je überhaupt niet op een andere manier op kunt lossen, (2) je niet betaalbaar op een andere manier op kunt lossen of (3) niet snel genoeg op een andere manier kunt oplossen.

Laat ik van deze criteria een aantal praktijksituaties schetsen:

VOORBEELD VAN #1
Überhaupt niet anders kunnen oplossen

1a. Technische onmogelijkheid: Applicatie X biedt geen koppelmogelijkheden en toch wil je de data in een specifiek ander systeem hebben. De softwarerobot functioneert feitelijk als koppeling (“overtyper”) en de gebruiker heeft het verschil met een echte koppeling in de praktijk vaak niet eens door. Zonder RPA zou de huidige situatie in stand blijven, namelijk de 2 applicaties moeten handmatig worden ‘gesynchroniseerd’.

1b. Afhankelijkheid leverancier: Applicatie X is een cloud-oplossing die een externe leverancier levert aan talloze gebruikersorganisaties. De leverancier heeft zijn eigen backlog en de gewenste wijzigingen zijn überhaupt nog niet gepland en/of laten lang op zich wachten. Met RPA kun je dit oplossen, zelfs zonder dat de leverancier nodig is.

VOORBEELD VAN #2
Niet betaalbaar kunnen oplossen

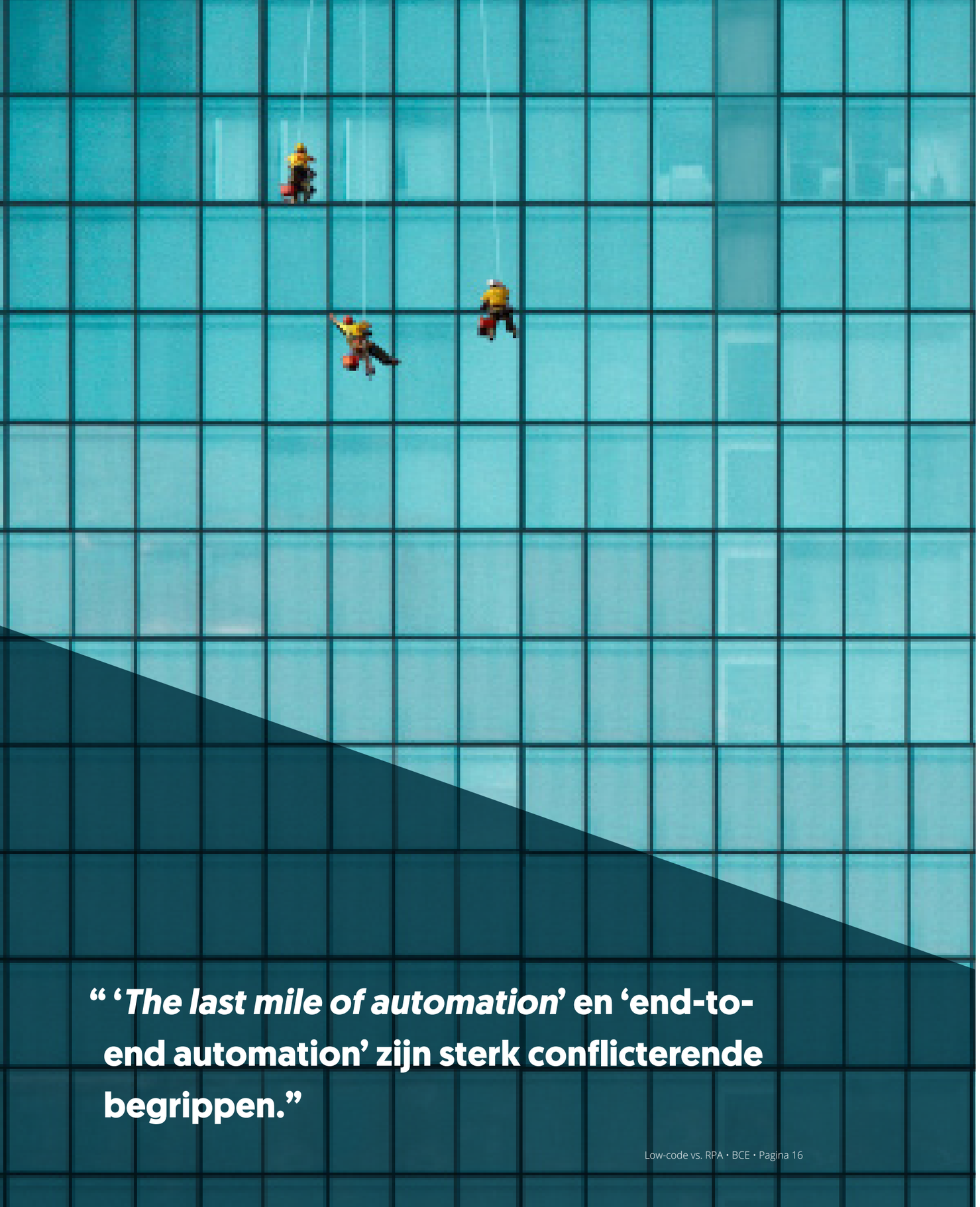
Applicatie Y moet eigenlijk vervangen worden; anno nu biedt deze helemaal niet meer wat nodig is. Echter, de vervanging kost tonnen waarvoor dit jaar helaas geen budget is. Toch moet op korte termijn een probleem opgelost worden. RPA biedt de wel betaalbare, haalbare oplossing (wellicht al voor 10 a 15K EUR).

VOORBEELD VAN #3
Niet snel genoeg kunnen oplossen

3a. Schaarste ontwikkelcapaciteit: Applicatie Z is gebouwd met mooie, moderne technologie. Jullie ontwikkelaars zijn echter schaars/overvraagd en hebben een lange backlog te gaan. Tegelijkertijd moet een specifiek werkproces echt op korte termijn aangepakt worden. RPA biedt een tijdelijke oplossing (en vraagt om een andere expertise en belast dus niet de schaarse ontwikkelaars).

3b. Schaarste operationele capaciteit: Er wordt nu gebouwd aan een toekomst-vaste oplossing, maar oplevering laat nog 5 maanden op zich wachten. Om allerlei redenen – bijvoorbeeld algemene werkdrukte of seizoendrukte – moet er voor die tijd al sneller en efficiënter worden gewerkt op de afdeling. RPA biedt in 1 à 2 weken al de tijdelijke oplossing.

Hoewel dit artikel erg kritisch is op RPA, illustreren de voorbeelden dat RPA in specifieke situaties enorm krachtig is in de gereedschapskist genaamd digitale transformatie. Immers, ook in het dagelijkse leven heb je bepaalde gereedschapstukken die je niet vaak gebruikt… maar wat ben je blij dat je ze hebt op het moment dat the usual suspects je niet verder kunnen helpen.



“ ‘*The last mile of automation*’ en ‘end-to-end automation’ zijn sterk conflicterende begrippen.”

INZICHT 6

Als een softwarerobot een end-to-end proces automatiseert, is dat een buitengewoon zorgwekkend signaal

Recent toonde iemand mij een demo van een softwarerobot die een ongelofelijk lang proces uitvoerde. Tientallen stappen. Ik was stomverbaasd. Er was maar 1 conclusie mogelijk: de reguliere bedrijfsapplicaties voldoen blijkbaar echt op geen enkele manier als ondersteuning van de processen. Ja, inzicht #5 maakt duidelijk dat er prima redenen zijn om RPA in te zetten, maar ‘the last mile of automation’ en ‘end-to-end automation’ zijn sterk conflicterende begrippen. Precies deze manier van inzetten zorgt namelijk voor RPA’s reputatie van “lelijke automatisering”. De vraag is zelfs ook of RPA’s klassieke voordelen (snel te realiseren, relatief goedkoop) nog wel standhouden in zo’n situatie. Misschien was een low-code-applicatie in deze ‘end-to-end’-situatie wel een beter en zelfs betaalbaarder idee…

Ten slotte

Al die verschillende subthema's van digitale transformatie zorgen voor een complexe uitdaging. Dit artikel wil positief bijdragen door er voor te zorgen dat je in ieder geval low-code applicatieontwikkeling en RPA nu goed uit elkaar kunt houden. Want in de gereedschapskist met de naam digitale transformatie horen ze echt allebei thuis. Niet als concurrenten, maar als partners.

"Low-code-applicaties en RPA zijn partners in de gereedschapskist genaamd digitale transformatie."

EEN KORTE INTRODUCTIE

BCE helpt middelgrote organisaties succesvol digitaal te transformeren

Onze missie // organisaties helpen om echt het verschil te maken. Met behulp van moderne technologie, bedrijfskundige kennis en gewoon gezond boerenverstand.

Onze mensen // gedreven, breed geïnteresseerde en sociale professionals met een achtergrond in o.a. bedrijfskunde, innovatiemanagement en/of informatica.

Onze rol // al sinds 2002 helpen wij onze klanten met hun grote IT-vragen. Als architect, product owner, programmamanager, projectleider, analist, consultant, strategisch sparringpartner, interim CIO; we zijn niet makkelijk in een ‘hokje’ te stoppen. Wij zijn niet verbonden aan specifieke IT-oplossingen en geven dan ook een volledig onafhankelijk IT-advies.

We geven onafhankelijk IT-advies

Wij zijn niet verbonden aan specifieke IT-oplossingen of partnerships en zijn daardoor volledig onafhankelijk. Van ons dus geen mooie praatjes, puur omdat we er zelf aan kunnen verdienen. We kijken objectief naar de beste oplossing voor jouw ambities. Daarbij houden we een open blik en leggen we altijd meerdere opties naast elkaar. Zeldzaam in IT-land.

MEER WETEN?
Lees **hier** hoe wij onze onafhankelijkheid waarborgen.

BCE

WAT IS JULLIE AMBITIE?

Maak een vrijblijvende afspraak

Klaas-Jan Molendijk

Partner

klaas-jan.molendijk@bce.consulting / info@bce.consulting

06 81 14 13 72

www.bce.consulting