JH

ARCHITECTUUR · WHITEPAPER

# Reuse, buy of build: de juiste volgorde

Waarom je het goedkoopste rendement meestal al hebt

Jaap Hoeve

De Prompters B.V.

# Inhoudsopgave

## Koop je het of bouw je het?

De architectuurdiscussie begint bijna altijd bij de vraag of je iets koopt of bouwt. In je enthousiasme om iets bouwen wordt meestal vergeten te kijken naar wat je al hebt en hoe je dat kan hergebruiken. In zeven jaar bij banken en verzekeraars heb ik die vraag vaker overgeslagen zien worden dan gesteld. De volgorde die ik altijd aanhoud is:

- Eerst reuse,
- Dan buy,
- Build alleen als het niet anders kan

## Wat staat er al?

Vraag bij een verzekeraar wat je al kan met applicaties en functies die zij hebben al kunnen. Dat is een andere gedachtegang waarop je vaak geen antwoord krijgt. Voor elk project wordt er een nieuwe applicatie bedacht terwijl het probleem vaak opgelost kan worden door de functies van verschillende applicaties aan elkaar te knopen door Power Automate flow of een Azure function. Minder leuk dan een compleet nieuwe applicatie te bouwen maar wel sneller en goedkoper.

Reuse wordt vaak overgeslagen omdat het niet top of mind is. Er is geen leverancier die het komt verkopen, geen business case die iemand moet goedkeuren en geen implementatiepartner met een offerte. Als je een pijnpunt hebt moet je zelf opzoek naar een oplossing en of beoordelen of het mogelijk is. Dat kost meer moeite en tijd maar dat betaald zicht terug.

Begin daarom bij de inventarisatie:

- Welke systemen raken dit proces al,
- Welke koppelvlakken zitten daarop,
- Wat kunnen we met de licenties die we deze maand toch betalen.

Zo krijg je snel inzicht in wat je al hebt en of er een mogelijkheid is om de bestaande infrastructuur te hergebruiken om je probleem op te lossen.

## Waarom buy voor build gaat

Met een pakket koop je de doorontwikkeling van alle andere klanten mee. Tweehonderd kantoren dienen wensen in waar jij nooit aan hebt gedacht en de helft daarvan komt in jouw versie terecht zonder dat je er iets voor doet. Bij eigen bouw krijg je precies wat je zelf hebt bedacht en niets meer.

De ontwikkelkosten en de marge van de leverancier worden over al die afnemers verdeeld. Dat maakt kopen duurder en over vijf jaar vaak goedkoper zodra je onderhoud meerekent. Onderhoud is ook meteen het tweede argument: het is er niet. Patches, security, pentests en aanpassingen aan nieuwe regelgeving zitten in het abonnement.

Dat laatste weegt in de financiële sector zwaarder dan elders en vooral wat betreft AI. Bouw je zelf een AI-systeem en zet je het onder je eigen naam in productie, dan ben je provider. Als je het koopt ben je een deployer. Als provider moet je veel meer geregeld hebben. Denk aan een risicomanagementsysteem, technische documentatie, logging, conformiteitsbeoordeling en registratie. Als deployer hoef je dit niet te doen want dit regelt je provider. Dus veel minder werk voor jou! Je maakt je met kopen afhankelijk van je provider en daar krijg je iets voor terug.

## Vier vragen, in deze volgorde

Loop elk bouwverzoek langs deze vier poorten. Je stopt bij de eerste die een antwoord geeft.

| | Vraag | Uitkomst |
|---|---|---|
| 1 | Raakt dit proces een systeem dat we al hebben en heeft dit een koppeling? | **Ja: reuse.** Bouw de koppeling, niet de applicatie. Nee: door naar 2. |
| 2 | Onderscheidt dit proces ons van een concurrent die hetzelfde pakket koopt? | Nee: door naar 3. Ja: door naar 4. |
| 3 | Dekt een bestaand pakket ruwweg 80 procent en kunnen wij ons proces op die 80 procent aanpassen? | **Ja: buy.** Pas het proces aan, niet het pakket. Nee: door naar 4. |
| 4 | Kunnen we dit vijf jaar onderhouden, ook nadat de bouwer vertrokken is? | **Ja: build.** Nee: alsnog kopen of het niet doen. |

Het gaat vaak mis bij poort 3. Iedereen vindt zijn eigen proces uniek en vanuit automatisering gezien is dat vaak niet het geval. Koop de standaard en bouw hooguit de uitzonderingen erin. Dat kan vaak via instellingen of andere kleine aanpassingen.

Build wint wel degelijk in de volgende drie situaties:

- Als het proces echt je onderscheidend vermogen is. Bijvoorbeeld een acceptatiemodel op je eigen schadehistorie.
- Als er domweg geen markt is en daarom is er geen aanbieder voor oplossing.
- Als de koppeling met je kernsysteem zo diep gaat dat elk pakket meer integratiewerk kost dan zelf bouwen. Anders bouw je iets wat iemand anders al voor je kan onderhouden.

## Reken over een tijdsspanne van vijf jaar

De discussie kantelt zodra je de bouwsom naast de kosten van een licentie en andere aanvullende kosten legt. Hieronder een fictief voorbeeldberekening voor een middelgroot documentcontroleproces.

| Kostenpost | Reuse | Buy | Build |
|---|---|---|---|
| **Inrichting of bouw, jaar 1** | € 15.000 | € 25.000 | € 120.000 |
| **Licentie per jaar** | In bestaande tenant | € 30.000 | € 0 |
| **Onderhoud per jaar** | € 4.000 | In abonnement | € 21.000 (18%) |
| **Security en pentests** | Op platformniveau | In abonnement | € 6.000 |
| **Wetgevingsaanpassing** | Platformleverancier | Leverancier | € 15.000 per keer |
| **Totaal over 5 jaar** | **€ 31.000** | **€ 175.000** | **€ 243.000** |

Die 18 procent is een vuistregel: lager bij een stabiel proces, hoger zodra er koppelingen in zitten die met elke release van de tegenpartij worden aangepast. Daarnaast ben je zo nooit afhankelijk van de kennis die samen met je bouwer vertrekt.

## De nuance van reuse

Reuse zonder beheer wordt duur. Je hebt nog steeds onderhoud maar een flow onderhouden is goedkoper dan een gehele applicatie onderhouden. Je moet dus wel minimaal een omgevingsstrategie, een naamgevingsafspraak en een aanwijsbare eigenaar per flow hebben. Anders eindig je met 30 flows in productie waarvan niemand weet wat die ook alweer doen.

Kopen verplaatst het risico niet want door de DORA-wetgeving blijf je als financiële instelling aanspreekbaar op je ICT-risico bij derden. Je houdt een informatieregister bij van je contractuele afspraken, je legt auditrechten vast en je hebt een uitgewerkte exitstrategie voor kritieke functies. Dit hoort bij het beheren van je applicaties en de DORA-wetgeving is strenger als je zelf AI-applicaties gaat ontwikkelen.

Om het mezelf nog wel even moeilijk te maken; AI-codeassistenten hebben bouwen de afgelopen twee jaar goedkoper gemaakt en dit zorgt dat keuzes richting build verschuift. Het raakt alleen de bouwkosten, patches, pentests, koppelingen die aangepast moeten worden en een aanspreekbare beheerder blijven precies even duur.