Top 10 punten om op te letten bij selectie softwarepakketten

Anton Boonstra, Mei 2012

Inleiding

Ik heb het al eens eerder een artikel voor 'Logistiek' over pakketselectie geschreven. Toen stond de vraag centraal: Is pakketselectie nog wel echt nodig? Insteek was vooral, dat een aantal 'spelers' de markt van ERP-software domineren met het aanbod van zeer complete pakketten, die een veelheid van eisen en wensen kunnen afdekken. Is pakketselectie daardoor niet een achterhaald idee en moeten we wellicht andere methodes hanteren om keuzes gefundeerd te kunnen maken? Antwoord in dat artikel was dat pakketselectie voor de meer algemene ERP-pakketten wellicht enigzins achterhaald is, maar dat voor de meer specialistische pakketten en bepaalde sectoren dat nog niet het geval is, en daarom pakketselectie dus zeker nog een nuttige functie kan hebben.

Als je bijvoorbeeld kijkt naar meer specifieke gebieden zoals 'forecasting' of 'productconfiguratoren' zie je nog stevige ontwikkelingen en discussies gaande hoe daar het beste vorm aan te geven. Aan de andere kant is op het gebied van CRM bijvoorbeeld sprake van een zeer goed aanbod van pakketten, een functionele vragenlijst zal weinig verschillen opleveren. Het is dus erg afhankelijk van het gebied en/of de sector wat je nu het beste kunt doen.

Daar waar software-oplossingen nog stevig uit elkaar kunnen lopen, is een uitgebreide functionele vragenlijst handig om in kaart te kunnen brengen wat wel en wat niet aan eisen en wensen is afgedekt om vervolgens daar een keuze op te baseren.

In dit artikel wil ik nader ingaan op een Top10 selectiecriteria voor software-pakketten. Ik hanteer daarbij een hoofdindeling van leverancier (aanbieder van de software), markt (van de afnemer) en techologie. Met technologie doel ik op een aantal basisgedachten die ten grondslag liggen aan het ontwerp van de software en die in het gehele pakket doorwerken tot in de details.

Leverancier

1. *Het is belangrijk dat de aanbieder een global player is, financieel sterk en redelijk wat geld investeert aan R&D (verbetering van de software, bijhouden van ontwikkelingen)*
   Global player. Je moet er vanuit kunnen gaan dat de partner van jouw keuze niet met de eerste beste partner fuseert en dat het door jouw gekozen pakket door zo'n overname in de verdrukking komt. Veel software-partijen zijn de afgelopen gefuseerd, overgenomen etcetera. Er wordt dan veel tijd gestoken in niet-product zaken zeg maar, of in het integreren van de software waardoor minder tijd wordt besteed aan nieuwe ontwikkelingen en uitbouwen van het softwareproduct, integratie is dan het hoofddoel geworden. Wat je wil bereiken met jouw keuze is dat de software state of the art is en blijft. Main focus van jouw leverancier moet dus zijn: product, product en product. Des te meer gebruikers er zijn, des te meer kans op verbetervoorstellen vanuit die gebruikers, des te beter de kosten van ontwikkeling verspreid kunnen worden over die gebruikers. Hoe groter de partij hoe groter de kans is dat het pakket

mee evolueert en het liefst haar markt voor blijft. Dit zie je ook vaak weergegeven in R&D – getallen. Zo geeft SAP 16% van haar omzet uit aan R&D. Dat is relatief veel, het gemiddelde zal zo rond 10-12% liggen.

2. *Hoe lang is deze leverancier al op de markt met het pakket van je keuze*. Dit geeft enigzins de stabiliteit van het pakket aan en daarmee sluit ik eigenlijk aan op het vorige punt. Waar ik met dit punt graag aandacht voor wil is het feit dat als een leverancier langer op de markt is ze wellicht niet volledig meegeevolueerd is, ze heeft misschien 'stilgestaan'. Ik heb de afgelopen jaren veel pakketten gezien, zeker de wat oudere die eigenlijk niet meer goed in deze tijd passen en die onvoldoende flexibel zijn om toekomstige ontwikkelingen relatief gemakkelijk op te vangen. Dus hoe langer het pakket op de markt des te stabieler waarschijnlijk maar de keerzijde is dat het pakket niet mee geevolueerd is.
3. *Installed base.* 'Installed base' is het aantal keren dat het pakket al is geinstalleerd. Dit geeft aan hoeveel klanten met het pakket werken. Hoe groter de 'installed base', hoe lastiger zal het zijn om klanten in de kou te laten zitten, dus zal het de leverancier er alles aan gelegen zijn deze klanten optimaal te bedienen. Veelal wordt aan de installed base relatief goed verdiend. Sowiezo betalen klanten 20% onderhoud per jaar maar zullen ze ook regelmatig licenties bijkopen. Hoe hoger de installed base hoe gemakkelijker je kan meeliften op de ontwikkelingen die voor collega-klanten zijn gemaakt.
4. *Licentiestructuur.* Concurrent user of per user? Niets is zo 'mistig' als de licentiestructuur maar de meeste varianten zijn terug te brengen naar de basis-varianten: 'concurrent user' of 'per user'.

   'Per user' houdt in dat voor iedere user een licentie moet worden betaald ongeacht de intensiteit van het gebruik. Dus of de gebruiker nu 1 uur per week gebruikt maakt of 20uur dat maakt voor de prijs niet uit. In geval van concurrent user wordt daar wel rekening meegehouden. 1 concurrent user kan in werkelijkheid dus meerdere users betekenen. In het verleden was het lastig vast te stellen hoeveel concurrent users er werkelijk waren maar inmiddels is goede software op de markt om dit te meten, dus dat mag geen probleem meer zijn.

   Maar pas dus op, leveranciers zijn zeer inventief als het om licentiestructuren gaat, er gaat gewoon veel geld in om. Vorig jaar maakte ik bijvoorbeeld een partij mee die werkte in staffels van 5. Dus voorbeeldje. Als je als klant 16 licenties nodig zou hebben val je in de staffel van 15-20 en betaal je voor 20. Het is belangrijk om vooraf dit soort constructies goed te begrijpen, zodat je op dat moment dat nog kan uitonderhandelen. Vaak wordt gezegd dat dat dan niet kan, neem van mij maar aan dat is echt flauwekul! Alles is onderhandelbaar!

   Softwarepartijen verdienen vooral hun geld met verkopen van licenties en daaraan gekoppeld onderhoud. Onderhoud is over algemeen een vast bedrag ergens tussen 15-20%, je mag verwachten dat dat dus ook grotendeels in R&D en productverbetering wordt gestopt. Het is vaak lastig te zien in de financiele verslagen of dat ook werkelijk gebeurd. Veelal wordt bij aanschaf korting verleend maar worden de onderhoudskosten aan de aanschafprijs gerelateerd om daarmee toch een wat hogere maintenance-fee te kunnen incasseren. Zorg dat je dus niet extra betaald voor features die eigenlijk ook onder onderhoud zouden kunnen vallen. Pas ook op dat je geen onderhoud betaald over maatwerk, dat is eigenlijk onzinnig.

Vaak wordt gezegd dat als een feature voor meerdere klanten tegelijkertijd wordt gemaakt dat dat dan onder maintenance-fee valt. Dat valt moeilijk te controleren. Daarom is een user-group die echt onafhankelijk is (dus niet gesponsord wordt of zo door de leverancier, of door de leverancier wordt georganiseerd) belangrijk. Een user-group biedt een platform om ideeen en gegevens uit te wisselen en dus ook om wijzigingsvoorstellen te formuleren die onder de 20% maintenance fee gemaakt moeten worden.

Markt

5. *Begrijpen ze jouw markt, passen ze ook bij jou?* Iedere sector heeft zo zijn eigenschappen die kenmerkend zijn voor die sector en die in een andere sector een minder prominente rol spelen. Neem in de Food de 'houdbaarheid' bijvoorbeeld en daarmee samenhangend 'lottracking- en tracing'. Je mag verwachten dat pakketten die zich specifiek op de Food richten voorzieningen hebben getroffen om met de houdbaarheid van producten om te gaan. Als dat niet in het pakket zit, is dat nog niet zo eenvoudig om dat even aan te passen. Zo heeft iedere sector wel een eigenschap die ook in de softwareoplossing weerspiegeld dient te worden.
   Het is niet alleen belangrijk dat de software 'jouw' sector weerspiegeld maar dat ook de leverancier laat zien dat ze de sector begrijpt. Daarmee bedoel ik dat ook de lokale partij waar jij als afnemer mee praat laat zien dat ze de problematiek begrijpt. Vroeg of laat moeten er wellicht aanpassingen worden gemaakt. Je moet dan wel een partij hebben die je begrijpt en met je mee kan denken in de oplossingen.
   Je moet ook het gevoel hebben dat de marktpartij jouw taal spreekt. Als je zelf een kleine partij bent zie nog al eens een wat arrogante houding van een grote marktpartij die daar tegenover staat. Als je het idee hebt dat de marktpartij niet bij je past moet je er gewoon niet aan beginnen. Software kopen is een langdurige relatie aangaan met een marktpartij die met je mee moet denken en met je mee moet bewegen. Ik ben er van overtuigd dat software mits goed toegepast zeker een concurrentiewapen kan zijn, het is dus een strategische partnership wat je met een softwarepartij aangaat. Het klinkt misschien wat zwaar allemaal wellicht maar een product als software is behoorlijk aan veranderingen onderhevig omdat de markt ook voortdurend veranderd. Je moet je verzekerd weten dat een softwarepartij er dus alles aan zal doen om met jouw mee te denken en mee te bewegen.
6. *Referenties.* Referenties is de beste en ook snelste manier om vast te stellen of de software bij jouw past of niet. Referenties worden graag afgegeven. Partijen zijn vaak trots om over hun toepassingen te praten. Terecht zou ik bijna zeggen. Software kopen is een ding maar software implementeren is een tweede en dat is zeker geen sinicure. Als je er dus een succes van weet te maken mag je daar terecht trots op zijn. En daar wil je natuurlijk ook graag over praten. Ik heb in mijn dagelijkse praktijk eigenlijk weinig gemerkt dat partijen er tegen zijn om jou in hun keuken te laten kijken. Echt 100% concurrenten zijn natuurlijk begrijpelijkerwijs wat voorzichtiger, maar over het algemeen zie ik alleen maar positieve ervaringen.
   Het enige waar ik voor wil waarschuwen is dat referenties niet gauw de vuile was zullen buiten hangen dus wees daarom op je qui vive. Ook moet je je verzekerd weten dat er klanten zijn die

niet als referentie willen optreden en soms kan het ook wel handig zijn om die een belletje te geven. Ik heb dat zo nu en dan wel aan de hand gehad en bedenk dat niet altijd dat een zaak van een slechte relatie met de leverancier hoeft te zijn. Er kunnen ook andere plausibele redenen zijn waarom men je niet graag in de keuken wil laten kijken.
Wat ik zelf altijd wel doe is even kijken wat mijn directe concurrenten voor softwareoplossingen gebruiken. De vraag is dan of je dezelfde software wil hebben als je concurrent of juist bewust niet.

Technologie

7. *Relationele database met daarop bedrijfsinformatie*. Wat je vaak ziet, zeker bij oudere marktpartijen dat voortgeborduurd wordt op een datastructuur die inmiddels al decennia oud is. Voor de marktpartijen is het belangrijk om regelmatig de datastructuur goed onder de loep te nemen en te bepalen of die nog wel juist is en de tand des tijds kan doorstaan.
Datastructuren maken is an sich niet zo moeilijk maar vervolgens worden aan de datastructuur allerlei programma's vastgeknoopt en dan je datastructuur veranderen is meestal niet zo eenvoudig. Je kan het vergelijken met een huis bouwen. Als je een huis bouwt kun je vaak nog tijdens de bouw vrij eenvoudig een muurtje verplaatsen of een raam of kozijn, maar als het huis eenmaal staat, wordt dat toch lastiger. Zo is dat dus ook met datastructuren.
Als zo'n datastructuur staat worden er vervolgens programma's aan vastgeknoopt en ook bedrijfsinformatiesoftware. Datastructuren vertellen je hoe gegevens worden opgeslagen, bedrijfsinformatie ontsluit die informatie als het ware. Let bij het bekijken en beoordelen van software dat de datastructuur en de bedrijfsinformatie een geheel vormt. Helaas wordt er nog steeds veel software op de markt gebracht waar veel tijd en geld wordt besteed om data vast te kunnen leggen maar waarbij het moeilijk is om de informatie ook daadwerkelijk eruit te halen, en daar was het uiteindelijk allemaal toch om begonnen.
8. *Consistentie van processen, toegang en opbouw schermen.* Zeker de oudere pakketten zijn aan elkaar geknoopt. Verschillende software is in de loop der tijd aangebouwd, aangekocht en vervolgens verbonden met de andere delen. Verschillende technologieen zijn gebruikt. Vaak zie je dit aan dat de toegang tot schermen maar ook de schermen zelf anders van opbouw zijn. Dat is reuze irritant voor de gebruiker. Die moet dan onthouden hoe bepaalde schermen werken. Dat is dus niet goed. De menustructuur maar ook de schermen moeten een logische vaste opbouw hebben. Terminologie moet hetzelfde zijn. Ik zie bijvoorbeeld heel vaak dat artikel en product bijvoorbeeld door elkaar wordt gebruikt terwijl het hetzelfde is. Dat is reuze irritant en leidt tot ineffitiviteit en uiteindelijk natuurlijk ook tot inefficiency
9. *Open Systeem*. Koppelbaar met andere systemen. Ik ben zoals ik vaker heb laten zien een voorstander van 'best of breed'. Dat wil zeggen dat je de beste oplossingen zoekt en vervolgens wel een interface moet leggen tussen de verschillende systemen. Interfacing is minder griezelig dan mensen vaak denken, vergeet niet dat ook modules zijn geinterfaced. Belangrijk ook hier dat je interfaces strak, eenvoudig en logisch maakt. Ook hier geldt natuurlijk dat interfaces op een gelijke manier opgelost moeten zijn, het moet voordehandliggend zijn en dus ook consistent .

10. *Onderhoud.* Er wordt door klanten grof geld betaald voor onderhoud. Zorg dus ook dat je krijgt waar je recht op hebt. Vaak proberen softwarebedrijven oplossingen los te verkopen en extra te verkopen. Zorg dat je een uservereniging hebt die op dit soort zaken let. Omdat je er grof geld voor betaald mag je ook verwachten dat je waar voor je geld krijgt. Je moet dus regelmatig veranderingen en ontwikkelingen 'gratis' kunnen krijgen

Afsluiting

Softwareselectie is geen eenvoudige zaak. Er zijn een hoop criteria te bedenken die van belang zijn bij de keuze en die verschillen ook nog eens van persoon en discipline. Ook de weging (het belang) van criteria is lastig. Daarom zie je dat er veelal gewerkt wordt in een 2-traps raket: Je maakt een grove schifting op basis van een knock-out lijst om vervolgens voor een pakket of drie in de details te duiken.

Knockout betekent dat als aan een van de gestelde critiria niet voldaan kan worden de kandidaat uit de selectie verdwijnt. Om toch enigzins tegemoet te komen aan de persoonlijke invulling aan zo'n lijst zou je iedere factor een cijfer kunnen geven op een schaal van 10 en kunnen afspreken dat een pakket niet acceptabel is wanneer er een factor onder de 6 komt.

Bovenstaande punten zouden mijnsinziens punten van de knock-out lijst kunnen zijn. Deze lijst zou je nog kunnen aanvullen met specifieke functionele knock-out criteria om de knock-out lijst daarmee compleet te maken. De top-10 moet je dus helpen om een grove schifting te maken.