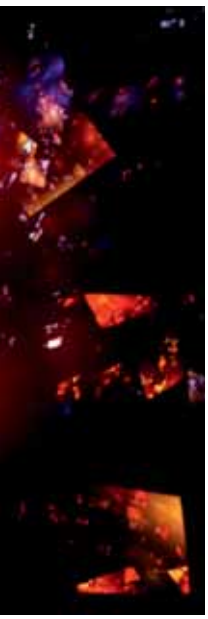


Veel meer vertrouwen tussen partijen wordt vereist

Agile en BI – het perfecte huwelijk?

Tom Breur

'Agile BI' lijkt een nieuwe hype. Als een leverancier wil suggereren dat er snel resultaten zullen worden geboekt, dan lijkt Agile BI het toverwoord. Zoals met zoveel hypes kan een klein beetje kennis gevaarlijk zijn, want de meeste van de promotieverhalen zijn zo evident non-agile, dat het pijn aan je ogen doet.

Ten eerste is 'Agile' geen IT-feestje, het is een principiële keuze voor een andere samenwerking tussen business en IT. Er bestaat niet zoiets als 'Agile IT.' Agile teams worden gekenmerkt doordat zij multidisciplinair zijn. Business gebruikers en IT zitten daar full-time in.
Agile BI is ook *zeker geen* responsieve BI. Dit misbruik van de term Agile BI roept het belachelijke beeld op van: "Vroeger kregen we nooit iets gedaan bij IT, maar sinds ze Agile werken reageren ze wel op onze verzoeken." Het is te pijnlijk voor woorden als je even stilstaat bij het zelfbeeld dat achter dit gebruik van de term schuil gaat.

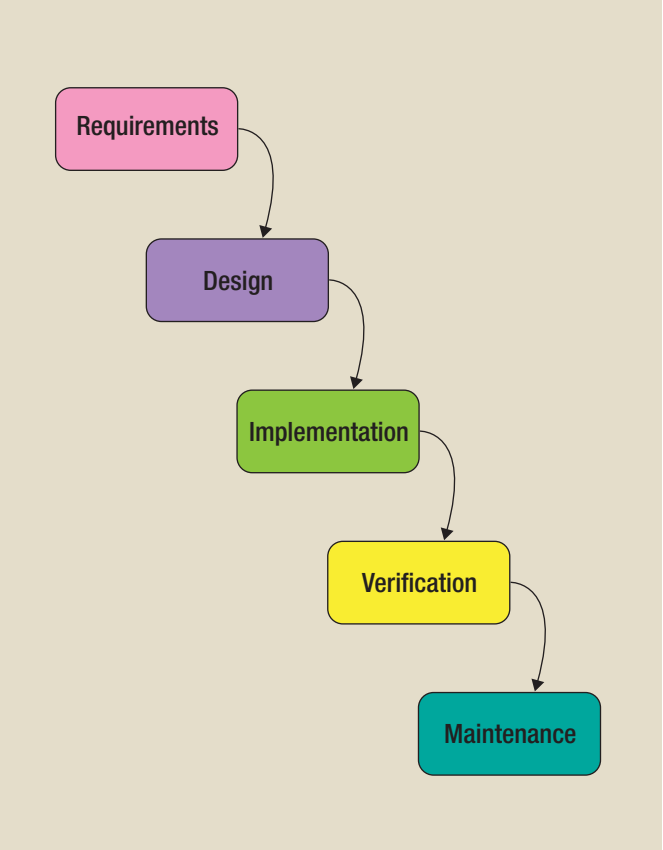
Oorsprong

De historie van software-ontwikkelmethoden begon in de jaren zestig van de vorige eeuw met een attitude van 'code and fix'. Vanaf de jaren zeventig deden meer formele methoden voor ontwikkeling hun intrede, bijvoorbeeld Jackson (Principles of Program Design, 1975).
'Agile' is een verzamelnaam voor een familie van software-ontwikkelmethoden die onderling veel overeenkomsten vertonen. Meer overeenkomsten dan verschillen, in ieder geval. Als beweging zetten zij zich af tegen watervalmethoden zoals Prince II. Sommige van de bekendste zijn eXtreme Programming (XP), Scrum, Lean software development, Crystal, of DSDM. Oorspronkelijk sprak men van 'lightweight methods' als reactie op 'heavyweight methods' (waterval) waar de nadruk ligt op gedetailleerde documentatie. De term Agile heeft zijn oorsprong in februari 2001, toen het Agile Manifesto (www.agilemanifesto.org) werd opgesteld. Deze methoden bestonden dus al voor de term Agile zelf werd geïntroduceerd. De term eXtreme Programming is in gebruik sinds 1997, het DSDM consortium werd al in 1994 opgericht.
De ontwikkeling van Agile was een logische evolutionaire ontwikkeling. Enerzijds als een verlengstuk van iteratieve methodes, bijvoorbeeld 'A Spiral Method of Software Development and Enhancement' (Boehm, 1986), of 'Rapid Application Development' (Martin, 1991). En anderzijds als een reactie op prescriptieve methodes die er op gericht zijn de menselijke variantie te beteugelen door ontwerpen te vertalen naar gedetailleerde instructies aan ontwikkelaars.

Elke keer als een ontwerpbeslissing tijdens de bouw tot problemen leidt (of erger nog: later) ontstaat er spanning in het watervalmodel. Het water 'moet' weer een stukje terug omhoog stromen en dat gaat niet goed. En hoe verder je terug omhoog moet, hoe moeilijker en duurder dat wordt. Het watervalmodel is er ook op gericht om dat moeilijk te maken, zie afbeelding 1. Binnen Agile geldt het omgekeerde principe: neem ontwerpbeslissingen *zo laat mogelijk*, als je meer weet omdat het product verder ontwikkeld is (en mogelijk gedeeltes van de functionaliteit al in gebruik zijn genomen).
Laat bouwen en implementeren hangt samen met het 'YAGNI' principe: You Ain't Going To Need It. Je wilt je zo laat mogelijk committeren aan ontwerpbeslissingen omdat je in de tussentijd meer hebt kunnen leren over wat er *eigenlijk* nodig is. Dit is dus het tegenovergestelde van "alvast een database inrichten omdat we die later *toch* nodig hebben". Niet doen. Zelfs als je weet dat je deze nodig zult hebben, ken je nu nog niet alle requirements. Vaak blijkt achteraf dat je deze net iets anders had willen inrichten.

Hoe 'werkt' Agile?

Agile is er op gericht om zo snel mogelijk waarde te creëren door werkende software op te leveren. Nu zal niemand er prat op gaan dat hij laat wil opleveren, dus hoe werkt dat in de praktijk? Agile gaat uit van verandering, verwacht verandering, en



Afbeelding 1: De watervalmethode.

beschouwt dat als een waardeverhogende factor in het bouwproces. Dus moet de methode daar in voorzien. Daarom wordt er zo min mogelijk gedocumenteerd. Want elke verandering veroorzaakt een cascade aan extra werk om al die documentatie weer up-to-date te houden. En als het er op aan komt, wat geloof je dan? Wat er in de documentatie staat, of in de code? De code geeft per definitie de laatste stand van zaken weer.
Een ander principe binnen Agile is dat grote projecten worden opgeknipt in hele kleine, sequentiële deelprojecten. Bij Scrum, bijvoorbeeld, spreek je dan van 'sprints', waarbij in een maand tijd werkende functionaliteit wordt opgeleverd. Features die zijn getest en gedocumenteerd.

Wat ook essentieel is voor Agile, is code en database 'refactoring.' Binnen elk increment (of sprint) haal je een slag over de code heen die er op gericht is deze onderhoudbaar te maken. Bijvoorbeeld subroutines verwijderen die niet (meer) worden gebruikt. Maar ook het hernoemen van variabelen en herschrijven van code om het programma leesbaarder te maken. Denk ook aan het toevoegen van commentaar binnen de code, ook als je weet dat je zelf degene bent die het programma verder gaat uitbouwen.
Soms kan de verleiding ontstaan om te 'bezuinigen' op refactoring, maar daaraan moet je weerstand bieden. Want als je je eenmaal op die helling begeeft is de weg terug omhoog extra lang en moeizaam. Hoe eerder je begint met refactoring, hoe goedkoper het is. En het doel van Agile is om zo veel mogelijk waarde voor de business te creëren.
Doordat met elk increment meer waarde wordt gecreëerd leer je binnen het (multidisciplinaire) team waar het geld verdiend moet worden. Daarvoor moet je BI-oplossingen ontwikkelen die het meeste waarde genereren voor de business. De beste manier om daar achter te komen is door er in hele kleine stapjes naar toe te groeien.

Agile en budget

In traditionele (waterval) benaderingen van software-ontwikkeling stel je een projectplan op door functionele eisen te vertalen in technische specificaties. Je schat de benodigde capaciteit daarvoor, en maakt een planning. Hierop komt akkoord, en je gaat aan de slag. Beide partijen committeren zich aan resources, budget en planning. Tot zover de theorie.
In een 'Agile' project wordt vooraf veel minder geïnvesteerd in planning, in plaats daarvan wordt een 'metafoor', of 'beeld' van het gewenste einddoel geschetst. Per increment (of 'sprint') bepaal je volgens het MoSCOW principe welke features absoluut geleverd moeten worden, en welke gewenst zijn.

Het is cruciaal dat de 'must have' wensen beperkt blijven. Ook als alles tegen zit, er een griepepidemie uitbreekt, sleutelpersonen binnen het team ontslag nemen, een hond alle specificaties verorbert, en de brug open stond, *moeten* de 'must haves' gerealiseerd kunnen worden. Zo niet, dan moet de scope van de 'must haves' worden verkleind.
Binnen het team heb je de autonomie om te kiezen welke van de gewenste opties je realiseert op basis van bevindingen tijdens de bouw. De belangrijkste overweging daarbij is welke features het 'beste' blijken te werken voor de business, dat wil zeggen het meeste waarde genereren.
Daarom is het *essentieel* dat de verhouding tussen 'must haves' en 'should haves' voldoende ruimte biedt. Alleen zo kun je optimaal profiteren van het multidisciplinaire team, en efficiency voordeel halen doordat verantwoordelijkheden zo laag mogelijk zijn belegd.
Dit illustreert waarom vertrouwen belangrijk is, en waarom zo'n hechte samenwerking tussen business en IT in het team een sleutelrol speelt. Een hoegenaamd 'Agile aanpak' met louter IT'ers heeft nooit dit resultaat, en is dus kolder.

De praktijk. Wat nu als in een watervalproject de planning niet gehaald wordt? Het is lang niet altijd mogelijk (vaak te laat) om opnieuw over de scope te onderhandelen. Dus heeft de business meestal weinig andere keuzes dan extra budget toe te kennen, de uitloop lijdzaam te ondergaan, of anders het project als geheel te staken (met forse verliezen).
De keerzijde bij Agile is dat vooraf weliswaar de lengte van de iteratie bekend is, en dus ook het bijbehorende budget, maar dat de beloofde functionaliteit beperkt is tot de 'must haves' in het eerstvolgende increment – en dat is weinig!

De tegenstelling tussen waterval en Agile komt neer op ofwel een vaste scope maar variabele planning en budget (al doet de projectleider je vooraf ongetwijfeld iets anders geloven), ofwel een vast budget en planning, maar met een variabele scope. Als het niet uit de lengte komt, moet het uit de breedte komen.
Niet weten waar je van tevoren in stapt is eng. Zeker voor managers die gewend zijn om IT op armlengte te houden. 'Agile' vereist (veel) meer vertrouwen tussen partijen. Precies dáár zit de fundamentele draai die je samen moet maken om 'Agile' te laten werken. Voor IT betekent dit dat verstoppen niet meer kan. En dat is (voor sommigen) best wennen. Maar de meeste ontwikkelaars die hier eenmaal van hebben 'geproefd' willen niet meer terug.

Waarom Agile voor BI?

Waarom kunnen Agile en BI nu zo'n gelukkig huwelijk vormen? De belangrijkste redenen hiervoor zijn dat BI-projecten (doorgaans) gekenmerkt worden door risicovolle projecten, en dat de manieren waarop achteraf waarde gecreëerd wordt onvoorspelbaar blijken.
Naarmate BI-projecten groter van omvang worden, is het soms bijna onmogelijk om vooraf tot in detail te plannen. Voor een independent data mart (de gemiddelde QlikView implementatie, zeg maar) is de scope nog wel te overzien. Maar zodra er substantiële data-integratie bij komt kijken wordt het moeilijker. Als bronnen nog niet eerder met elkaar zijn geconfronteerd weet je nooit wat je onderweg allemaal tegenkomt.
De tweede reden waarom Agile en BI zo voorspoedig samen kunnen gaan heeft te maken met de business case voor BI. Hoe vaak maak je niet mee dat de uiteindelijke 'cash cows' uit onverwachte hoek opduiken? Een BI-project begint natuurlijk 'gewoon' met een business case. Tenzij er een gezonde verwachting is dat de investeringen zichzelf ruimschoots zullen terugverdienen, moet je nooit beginnen aan risicovolle projecten, zoals een datawarehouse bijvoorbeeld. Maar voorspellen is moeilijk, aanpassen is gemakkelijker.

Je levert zo vroeg mogelijk werkende functionaliteit op. Weliswaar nog niet gepolijst, ronduit kaal, maar je zoekt de kleinste set van features waarmee de business aan de slag kan. Zo leer je 'live' *waar* je geld kunt verdienen (of besparen), maar ook *hoe*. Door het adaptieve karakter van Agile bepaal je per increment wat de beste koers is. Het project meandert zoals een rivier kronkelt naar de zee.
Uiteindelijk gaat commitment aan een datagedreven beliscultuur verder dan de inrichting van je BI-oplossing. Kimball suggereert dat de keuze voor datawarehousing eerder een *proces* behelst, dan een *project* met een vast begin en eind. Agile impliceert een vorm van samenwerking met verregaande autonomie binnen het team om de meest winstgevende route te vinden. Dit is wat Peter Drucker in zijn klassieker 'The Effective Executive' bedoelde met effectiviteit van kenniswerkers. Door autonomie zo laag mogelijk te beleggen kan daar waar kennis en inzicht ontspringt ook (wendbaar) ingespeeld worden op kansen in de markt.

Conclusie

Het modewoord 'Agile BI' wordt tegenwoordig vaak gebruikt om te suggereren dat Agile een soort van responsieve, succesvolle aanpak behelst. Is dat in tegenspraak met 'traditionele' BI? Zoiets in de trant van: "Vroeger namen ze de telefoon nog niet eens op, maar sinds ze Agile werken krijgen we ineens van alles gedaan." Een goedkope en doorzichtige verkooptactiek.
Zoals zo vaak in het leven, krijg je niets voor niets. Partijen die met Agile experimenteren laten wisselende resultaten zien. Soms omdat men Agile niet goed begrijpt, en verkeerd implementeert. IT laaft zich aan de nieuwste haarlemmerolie en richt teams 'Agile' in. Leuk geprobeerd, maar zinloos tenzij business partners zich niet alleen in *woord*, maar vooral in *tijd* willen committeren.

Laatst werd mij in een business meeting gevraagd wat er nodig was om een bepaalde (heel) ambitieuze planning te realiseren. Ik vroeg om een klein team met 50/50 hooggekwalificeerde IT'ers en business experts. De CFO brieste: "Maar daar hebben we hier geen tijd voor! Die capaciteit is er eenvoudigweg niet". (Hij doelde op de business mensen). Hij had een probleem met ineffectieve IT, en impliceerde dat *zij* dat maar moesten oplossen. Iets klopt er volgens mij niet aan die managementbenadering, tenzij je gelooft in de 'carrot and stick' benadering. Gelukkig had deze organisatie wel volop tijd om mislukte projecten opnieuw te doen.

Agile is op zijn best als je met kleine groepen hooggekwalificeerde mensen heel nauw kunt samenwerken; in dezelfde ruimte. Maar ook in grote organisaties kan het werken. De roots van XP gaan terug naar het befaamde 'C3' project bij Chrysler in 1995 (Chrysler Comprehensive Compensation system); een groot, internationaal project om de salarisadministratie te automatiseren. Full-time betrokkenheid van business partners met voldoende mandaat is essentieel voor het welslagen van Agile. Zonder die mix krijgt een team nooit het benodigde vertrouwen in de organisatie.
BI-projecten worden (dikwijls) gekenmerkt door weerbarstige planningen (met name de ETL-fase) en ambigue business cases. Door in hele kleine stapjes te groeien naar de gewenste eindsituatie vergaar je onderweg de meest valide feedback die je kunt wensen: functionaliteit die 'live' in productie gaat.

Het zoeken naar de uiteindelijke invulling van je business case vereist creativiteit, vindingrijkheid, en ondernemingslust. Dit stimuleer je door autonomie en beslisbevoegdheid binnen Agile-teams te beleggen. Bedrijfsdoelstellingen fungeren als kompas, het team bepaalt de optimale koers naar de best mogelijke BI-oplossing.

Tom Breur is eigenaar van XLNT Consulting.