Ontwikkelmethodieken geven voortschrijdend inzicht

Agile Business Intelligence

Sander Hoogendoorn en Sandra Wennemers

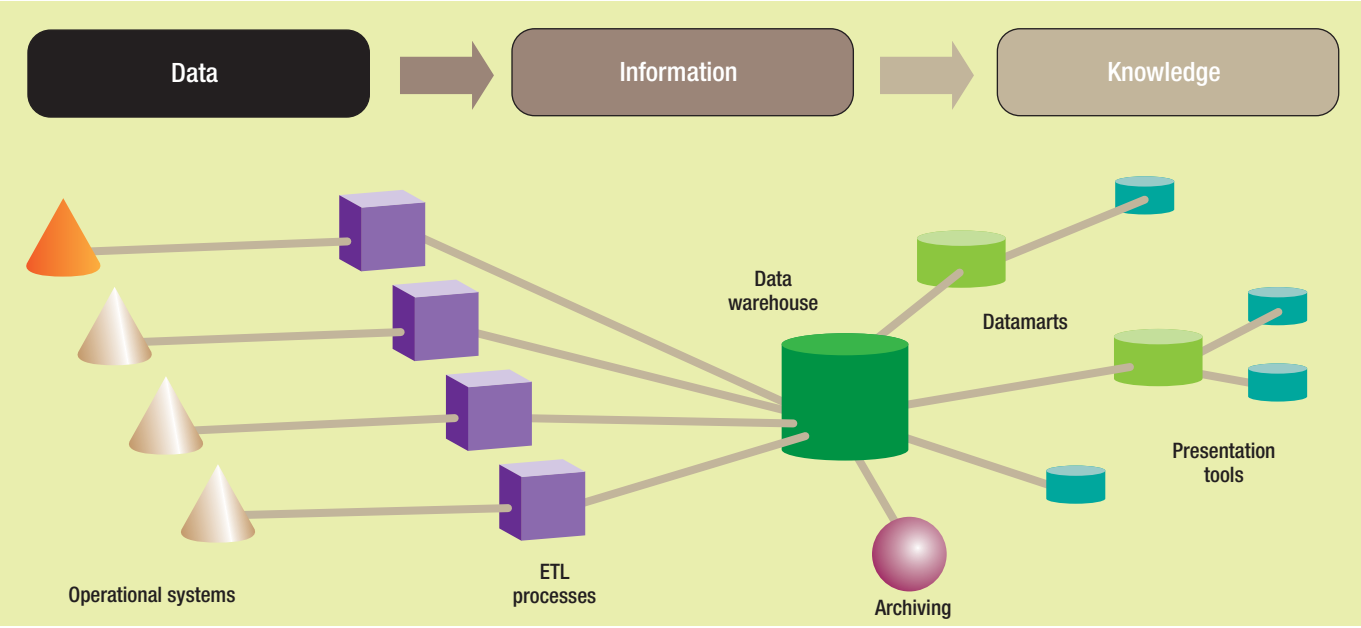
Om kosten te besparen wilde een bekende overheidsinstantie weten hoe effectief de bestrijding van uitkeringsfraude was. Hierbij speelde een interessant fenomeen. Het onderzoeken van mogelijke fraude kost de instantie geld, maar het vinden van fraudeurs levert daarentegen direct geld op. En dus ging men op zoek naar de optimale verhouding tussen het aantal onderzoeken en het aantal opgespoorde fraudes.

Men wilde met zo min mogelijk onderzoeken zoveel mogelijk fraudeurs vinden. De genoemde doelstelling is karakteristiek voor BI-projecten. Hoewel dergelijke doelstellingen vaak nog redelijk zijn te definieren, is de realisatie hiervan vaak schimmig. Welke rapportages moeten worden ontwikkeld en wat staat daar op? Welke bronsystemen moeten hiervoor worden geraadpleegd? En wanneer het project eenmaal loopt, doen zich doorlopend nieuwe inzichten voor. Bijvoorbeeld over het ontbreken van benodigde informatie in bronsystemen. En anders doet de opdrachtgever wel inspiratie voor nieuwe wensen en eisen op uit feedback over opgeleverde rapportages en analyses. BI-projecten kenmerken zich derhalve door onvolledige requirements en doorlopend voortschrijdend inzicht.

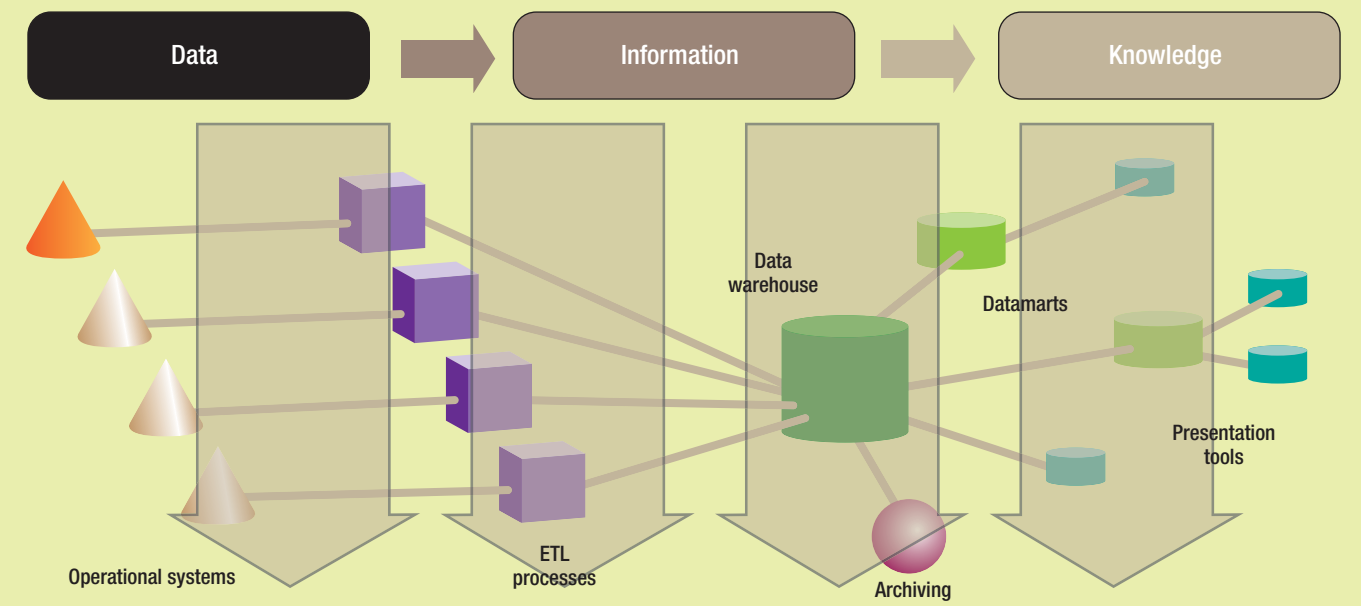
Op het gebied van systeemontwikkelmethodieken hebben zich de laatste jaren enorme veranderingen voorgedaan. Meer en meer organisaties en projecten stappen over op een nieuwe generatie methodieken, die voortschrijdend inzicht niet langer schuwen, maar juist omarmen en zich kenmerken door coöperatieve samenwerking en het frequent, in korte iteraties opleveren van software. Deze methodieken worden ook wel agile methodieken genoemd. De vraag is hoe deze methodieken, afkomstig uit regulier software development, ook een positieve bijdrage kunnen leveren aan het uitvoeren van BI-projecten.

Kenmerken project

De doelstellingen van BI-projecten zijn altijd direct aan de business gerelateerd, zoals het minimaliseren van verzekeringsfraude of het behouden van klanten. Tijdens een project worden analyses en rapportages gedefinieerd die ondersteunen bij het beheersen en optimaliseren van de bedrijfsprocessen van de opdrachtgever. Deze rapportages en analyses worden – meestal dagelijks – gevoed door het te ontwikkelen datawarehouse. Kenmerkend voor dit type projecten is dat lastig is vast te stellen



Afbeelding 1.



Afbeelding 2.

welke concrete bijdrage deze analyses en rapportages uiteindelijk leveren. Neem bijvoorbeeld de eerder genoemde overheidsinstantie, waar niet op voorhand was uit te drukken hoeveel geld men kon besparen bij het vinden van de optimale verhouding tussen het aantal onderzoeken en het aantal gevonden fraudeurs. Uiteindelijk bleek na afloop van het project dat de resultaten nog beter waren dan van te voren was geschat.
Alhoewel vroegtijdig in projecten nog is vast te stellen welke analyses en rapportages benodigd zijn, is de exacte invulling hiervan nauwelijks te formuleren. Wat wil de opdrachtgever nu echt zien in zijn rapporten? Neem als voorbeeld een rapportage over de verhouding tussen inkomende en uitgaande berichten bij een telecom operator. Pas toen de opdrachtgever het rapport onder ogen kreeg, bleken er diverse soorten inkomende berichten te zijn, die ook weer gekoppeld zijn aan diverse soorten uitgaande berichten. Een typisch voorbeeld van voortschrijdend inzicht.

Een interessant fenomeen is ook het Extractie-, Transformatie- en Laadproces (ETL). Hierbij worden in een aantal stappen de gegevens uit bronsystemen verzameld, geïntegreerd en geaggregeerd tot een formaat dat voor de rapportages en analyse benodigd is. In de meeste BI-projecten beslaat dit type werk zo'n 80 procent van de ontwikkeltijd, zie afbeelding 1.
Toch blijft dit werk meestal onzichtbaar voor de opdrachtgever. Deze concentreert zich – terecht – vooral op de op te leveren rapportages en analyses. Maar doordat de bulk van het werk in een BI-project op ETL is gefocust, worden meestal ook de fasen van zo'n project rond ETL ingericht. Dat wil zeggen, dat eerst alle extracties worden ontwikkeld, aansluitend de transformaties, om vervolgens alle gegevens te laden. Pas nadat dit is gelukt worden de rapportages en analyses gedefinieerd, zoals weergegeven in afbeelding 2.

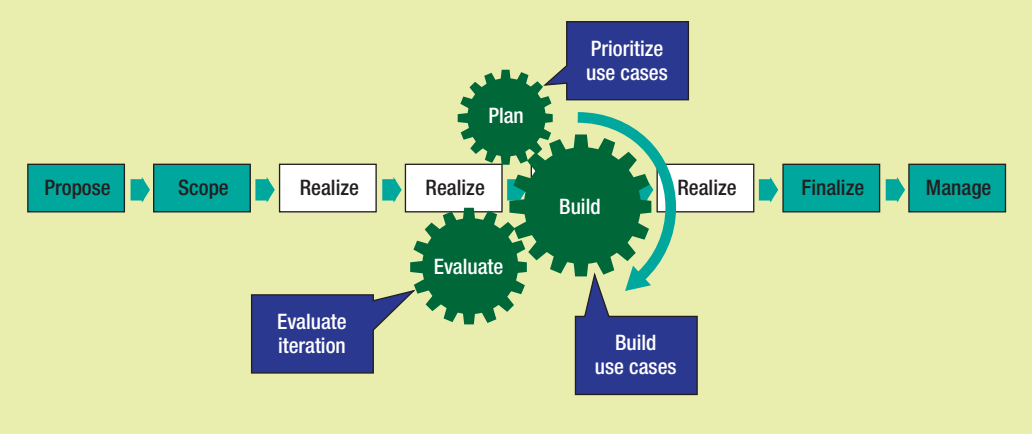
Met als gevolg dat pas in deze laatste fase van het project iets wordt opgeleverd waarmee de opdrachtgever aan de slag kan. Dit heeft een belangrijk nadeel. Pas in een laat stadium geeft de klant feedback. Mogelijke aanpassingen aan rapportages en analyses als gevolg hiervan zijn nu lastig te realiseren, omdat alle ETL al is opgeleverd. Bovendien is het niet uitgesloten dat als gevolg van deze feedback sommige stappen uit de ETL overbodig blijken. In dit laatste geval is er zelfs werk voor niets uitgevoerd. Dan wordt het BI-project een dure onderneming. Tenslotte kan het nog voorkomen dat voor de gewenste rapportages en analyses gegevens nodig zijn die niet direct uit de bronsystemen zijn af te leiden. Aanvullende gegevens worden dan vaak handmatig toegevoegd tijdens de ETL. Vaak worden hiervoor gaandeweg het project kleine administratieve applicaties ontwikkeld. Nog los van het feit dat deze applicaties door de verkeerde ontwikkelaars worden ontwikkeld – BI-specialisten en geen reguliere ontwikkelaars – worden de hiaten in gegevens meestal pas lopende het project ontdekt. Met alle gevolgen van dien.

Kenmerken agile software development

In regulier software development is de afgelopen jaren een hele nieuwe generatie aan methodieken ontstaan, die de best practices van vorige generaties koppelen aan een sterk iteratief en coöperatief karakter. Deze methodieken, zoals DSDM, extreme programming, Scrum, Smart en feature driven development (FDD) kenmerken zich alle als volgt:

- Korte iteraties. Software wordt opgeleverd in korte iteraties, variërend van twee weken tot een maand. Tijdens ieder van

Tijdens Trends in Informatiemanagement op 18 maart 2008 geven de auteurs een lezing over Agile Business Intelligence.



Afbeelding 3.

deze iteraties wordt een klein deel van de software geanalyseerd, ontworpen, gebouwd, getest en vaak zelfs opgeleverd. Pas bij de start van een iteratie wordt vastgesteld welke functionaliteit tijdens de komende iteratie wordt ontwikkeld. Hiermee verkorten projecten de feedback-lus met de opdrachtgever rigoureus. Zo verbetert de kwaliteit van de ontwikkelde software in hoog tempo. Dit in tegenstelling tot traditionele projecten, waarin de software vaak als een *Big Bang* wordt opgeleverd aan het eind van het project;

- Compacte eenheid van werk. Bovenstaande kan alleen worden bereikt als er een eenduidige en kleine eenheid van werk wordt gehanteerd in projecten. Hierbij geldt dat individuele work items directe waarde vertegenwoordigen voor de opdrachtgever en bovendien zo klein zijn dat er meerdere zijn te realiseren tijdens een individuele iteratie;
- Snel en frequent opleveren van software. Tijdens agile projecten wordt al tijdens de eerste iteraties software opgeleverd aan de opdrachtgever, al dan niet direct in productie. Dit zorgt ervoor dat problemen, bijvoorbeeld rond de architectuur, al vrij snel na de start van het project boven water komen;
- Incorporeren van voortschrijdend inzicht. Anders dan in traditionele projecten, waar voortschrijdend inzicht zoveel mogelijk wordt uitgebannen, is het in agile projecten mogelijk en zelfs gebruikelijk nieuwe requirements nog tijdens het lopende project mee te nemen. Hiertoe wordt pas bij de start van een iteratie vastgesteld welke work items tijdens deze iteratie worden opgepakt. Nieuwe work items kunnen hierbij worden meegenomen, boven al eerder benoemde werk items;
- Nauwe samenwerking klant en opdrachtnemer. Het snel en frequent opleveren van software in korte iteraties vraagt om een intensieve samenwerking tussen klant en opdrachtnemer, waarbij bij voorkeur op dagelijkse basis overleg plaatsvindt, bijvoorbeeld om de nieuwe gerealiseerde work items te bekijken;
- Geïntegreerd testen. Omdat software frequent en al vroegtijdig wordt opgeleverd in projecten, is het testen van de software van cruciaal belang vanaf dag één in een project.

> Meer organisaties stappen over op een nieuwe generatie methodieken

Smart

Een bekende agile methodiek is Smart, die veel is toegepast in reguliere systeemontwikkelprojecten. Deze methodiek is ooit ontwikkeld als implementatie van DSDM, maar is inmiddels uitgegroeid tot een kernachtige agile methodiek, waarin best practices zijn ondergebracht uit bijvoorbeeld Rational Unified Process, Scrum en extreme programming. Naast typische agile karakteristieken die hierboven zijn beschreven, kent Smart een aantal additionele technieken, voortgekomen vanuit de praktijk. Deze technieken geven projecten een eenduidige eenheid van werk, maar helpen ook om de voortgang te bewaken:

- Smart use cases. De eenheid van schatten, plannen, requirements-analyse, ontwerp en ontwikkeling is een gestandaardiseerde vorm van use cases, die wel smart use cases worden genoemd. Smart use cases worden gebruikt om de functionele requirements van een project uit te drukken. Interessant aan deze smart use cases is dat ze gelijkmatig van grootte en complexiteit zijn, waardoor ze gemakkelijk zijn in te schatten, en zo klein zijn dat er diverse binnen het tijdbestek van een enkele iteratie worden gerealiseerd;
- Smart estimation. Aanvullend hierop is een schattingstechniek die de totale complexiteit en omvang van een project uitdrukt in smart use case punten. De smart use cases worden hiertoe uitgedrukt op een eenvoudige schaal, die van 1 tot 5 gaat, maar voor uitzonderingssituaties 8 en 10 heeft gereserveerd. Er zijn voor diverse typen projecten standaardtypen use cases geïdentificeerd, die al zijn uitgezet op deze schaal. Dit geldt voor het ontwikkelen van bedrijfsapplicaties, service-oriëntatie, content management en reporting, bijvoorbeeld met behulp van Sharepoint. Inmiddels zijn er ook standaardtypen voor ETL opgesteld;
- Monitoring. De voortgang in Smart-projecten wordt voortdurend gemonitord via een tweetal pragmatische gereedschappen.

Zo wordt een agile dashboard gebruikt waarin de status van de smart use cases uit het project is uitgedrukt. Bovendien hanteren projecten een *burn down chart*, waarin steeds de verwachte einddatum is te extrapoleren.

Fasen van Smart

Smart kent een beperkt aantal eenvoudige fasen, zoals is weergegeven in afbeelding 3. Een project is verdeeld over twee voorbereidende fasen genaamd Propose en Scope, een kort-cyclische hoofdfase Realize, een afrondende fase Finalize en eventueel een fase voor het beheren van de applicatie die Manage wordt genoemd. Deze laatste fase start zodra het project de software heeft opgeleverd. In iets meer detail:

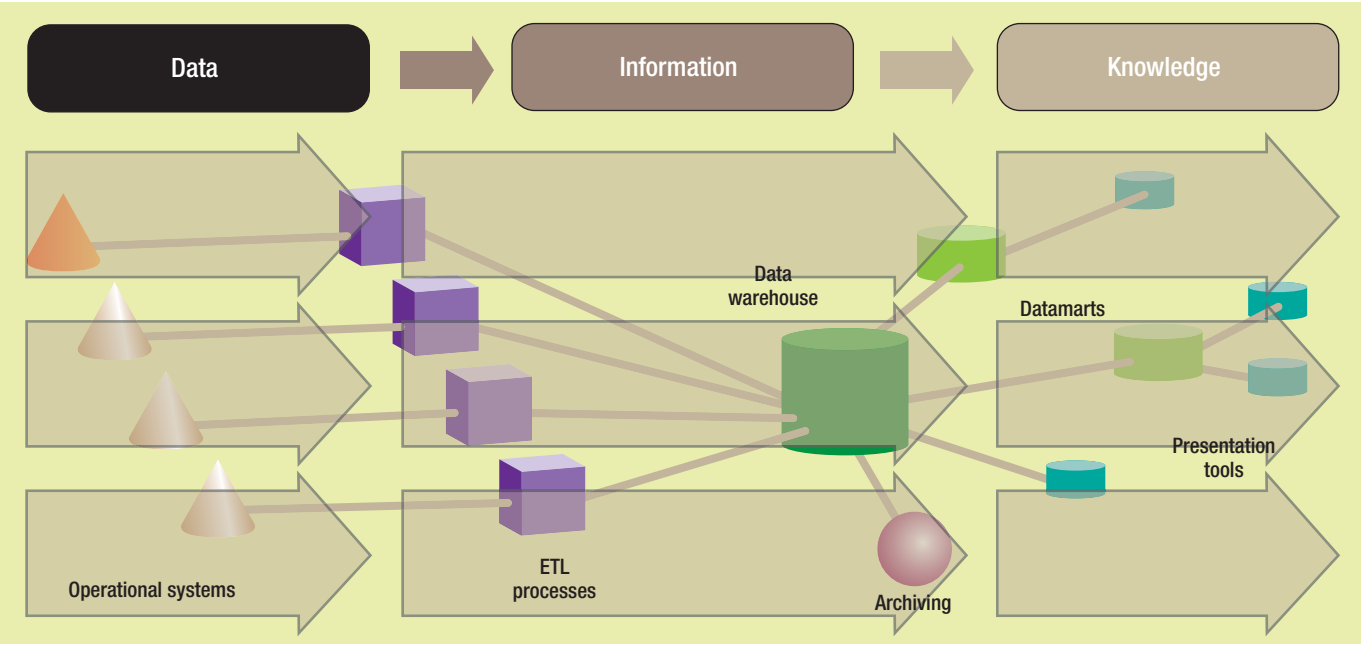
- Propose. Tijdens deze fase worden de scope, de omvang en de complexiteit van het project vastgesteld, uitgedrukt in smart use cases. Propose mondt uit in een projectvoorstel en een eerste schatting.
- Scope. Tijdens Scope wordt het projectvoorstel verder uitgewerkt, waarbij onder meer herbruikbare componenten en niet-functionele requirements op het menu staan. Deze fase leidt tot het plan van aanpak voor de uitvoering van de rest van het project.
- Realize. De fase Realize is gericht op het interactief realiseren van de software. Deze fase is opgedeeld in korte iteraties, liefst van twee weken. Tijdens iedere van deze iteraties wordt een aantal smart use cases gerealiseerd. Welke use cases dit zijn wordt vastgesteld bij de start van een iteratie (in de deelfase Plan). Daarna worden ze gerealiseerd tijdens een dagelijkse sub-iteratie in Realize. De iteratie wordt uiteindelijk afgerond en geëvalueerd (in deelfase Evaluate).
- Finalize. Tijdens de iteraties van de fase Finalize wordt de software afgerond. Er wordt geen nieuwe functionaliteit meer ontwikkeld. Vervolgend wordt het project afgerond en geëvalueerd.
- Manage. Deze fase beschrijft het onderhoud van de opgeleverde software. Ook hierbij gelden de smart use cases als uitgangspunt, en wordt meestal gewerkt in maandelijkse of tweemaandelijkse iteraties.

> Liever wordt er per rapportage ontwikkeld

Hoewel de meeste agile methodieken zijn voortgekomen uit de reguliere systeemontwikkeling, zijn de karakteristieken, technieken en fasering goed toepasbaar in Business Intelligence-projecten – wat in principe natuurlijk ook software development is. Laten we eens kijken.

Snel en frequent opleveren

Het snel en frequent opleveren van voor de klant relevante software is een aspect van agile methodieken dat in Business Intelligence bijzonder goed van pas komt. In plaats van de verticale fasering in traditionelere BI-projecten, kiezen we ervoor om het realiseren van analyses en rapporten juist horizontaal aan te pakken. Niet langer worden alle extracties eerst opgeleverd, aansluitend de transformaties en het laden van de gegevens om pas daarna de rapportages te ontwikkelen. Liever wordt er per rapportage ontwikkeld.
Hierbij kiest de opdrachtgever welke rapportages de hoogste prioriteit hebben, en start het team met het uitsluitend realiseren van de daartoe benodigde extracties en transformaties, en het



Afbeelding 4.

opstellen van de rapportage. Deze kanteling van werkzaamheden vergroot de mogelijkheden op feedback van de opdrachtgever, en de toepasbaarheid van het opgeleverde. Belangrijk bijkomend voordeel is dat de rapportages zodra ze beschikbaar zijn, soms al enkele weken na aanvang van het project, direct kunnen worden aangewend om de bedrijfsprocessen te verbeteren. Zo toont het project al op heel korte termijn zijn benefits.

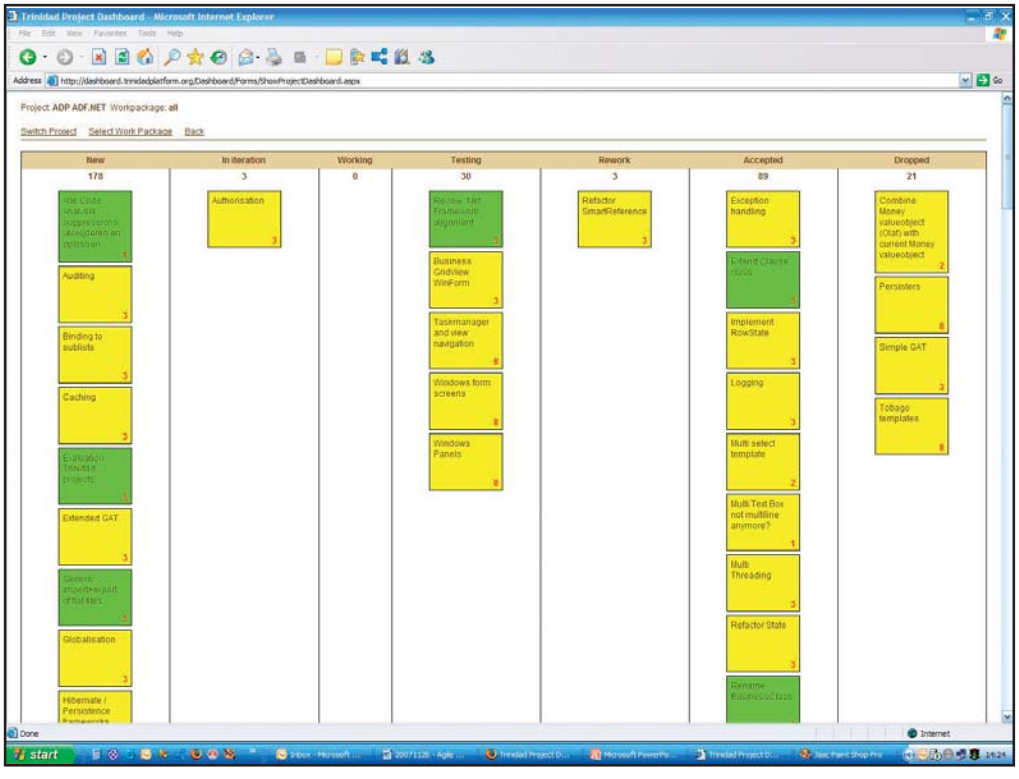
Korte iteraties

Agile software development promoot het werken in heel korte iteraties, bijvoorbeeld van twee weken of dertig dagen. Bij de start van iedere iteratie wordt beschouwd welke work items de hoogste prioriteit hebben. Dit zijn rapportages, maar ook individuele smart use cases die van pas komen bij de realisatie van een rapport. Zo wordt voortschrijdend inzicht nu eens niet uitgebannen, zoals in traditionele werkwijzen, maar kunnen nieuwe inzichten direct worden geïncorporeerd.
Hoewel het principe geldt dat tijdens de lopende iteratie de scope niet mag wijzigen, kunnen nieuwe requirements al tijdens een eerstvolgende iteratie op de rol worden gezet. Zodra de opdrachtgever de eerste versie van een rapportage of analyse onder ogen krijgt, kan hij of zij direct feedback formuleren, die ook weer direct wordt geïmplementeerd; zonder daarbij afbreuk te doen aan de structuur en voortgang van het project. We hebben in projecten gemerkt dat op die manier het in de praktijk benutten van analyses en rapportages (zeg maar de acceptatie) sneller tot stand komt. Tenslotte is het mogelijk nieuwe inzichten rond eventueel benodigde data entry direct op te pakken, zodra deze gaande het project worden onderkend.

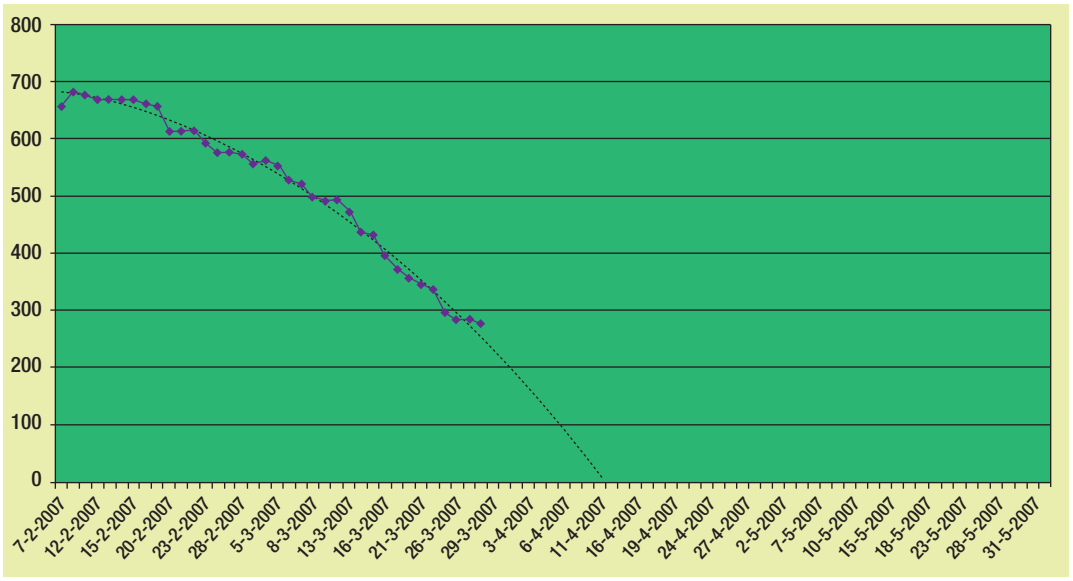
Compacte eenheid van werk

Kort door de bocht gesteld is een BI-project uit te drukken in drie typen ontwikkelwerk: datamodellering en ETL, het definiëren van analyses en rapporten en het ontwikkelen van data entry applicaties. In Smart geldt de smart use case als eenheid van werk, zowel voor het beschrijven van de requirements, het maken van schattingen als het realiseren van voor de gebruiker relevante functionaliteit – en zelfs als eenheid voor het testen van deze functionaliteit. Voor het modelleren van smart use cases gelden strikte richtlijnen, die ertoe bijdragen dat smart use cases een vergelijkbare lage granulariteit hebben, en al vroeg in een project zijn toe te passen – al tijdens de fase Propose. Er is een groot aantal standaardtypen smart use cases beschreven, die use case stereotypes worden genoemd. Voorbeelden hiervan zijn *manage* (onderhoud van een object), *search* (zoeken naar objecten), *graph* (rapportage) of *file import*.

Opvallend genoeg is gebleken dat smart use cases ook prima in agile BI-projecten kunnen worden aangewend. Wat betreft het definiëren van rapportages en het ontwikkelen van aanvullende data entry ligt dit voor de hand, omdat dergelijk werk niet wezenlijk verschilt van regulier software development. Maar ook voor het vaststellen van analyses en zelfs voor het uitvoeren van ETL zijn inmiddels use cases stereotypes vastgesteld, zoals *collect, integrate* en *aggregate*.
Smart use cases worden al vroeg in een agile BI-project geïdentificeerd, tijdens de Propose en Scope fasen. Daarbij wordt alleen de overview van use cases vastgelegd; details worden pas uitgewerkt tijdens realisatie. Aan de hand van deze overview



Afbeelding 5.



Afbeelding 6.

is de omvang en doorlooptijd van het project in te schatten. Aansluitend worden de iteraties uit de fase Realize ingepland. Tijdens deze iteratie ligt de focus op het realiseren van individuele rapportages, op basis van de hiertoe benodigde smart use cases voor ETL, eventuele data entry en de definitie van de gewenste rapportage, zie afbeelding 4.
Deze wordt zo snel mogelijk gerealiseerd en met de opdrachtgever afgestemd. Bijkomend voordeel is dat de onderliggende dataflows, die nu zijn uitgedrukt in smart use cases, beter individueel te testen zijn, en zelfs is door het modelleren van de use cases hergebruik van dataflows snel geïdentificeerd.
In BI-projecten is het snel en frequent opleveren van nieuwe rapportages en analyses van groot belang voor de opdrachtgever. Immers, iedere nieuwe rapportage kan direct worden benut in de praktijk en levert zo direct toegevoegde waarde voor het optimaliseren van de bedrijfsprocessen van de opdrachtgever. Daarnaast hebben BI-projecten baat bij het voortschrijdend inzicht dat onstaat dankzij de korte iteraties in agile projecten. Beter dan dit uit te bannen, zoals traditioneel wordt gepoogd, is het om juist effectief gebruik te maken van deze inzichten en feedback. Daarbij is de juiste keuze om in projecten te focussen op de realisatie van individuele rapportages en analyses, in plaats van laag voor laag de benodigde ETL te ontwikkelen.

Agile dashboards

Daarbij biedt Smart een tweetal pragmatische gereedschappen om de voortgang van het project te bewaken; een agile dashboard en een burn down chart. Tijdens de uitvoering van een project verkeren de individuele smart use cases in verschillende, opvolgende statussen, zoals New, In Iteration, Working, Testing, Rework en Accepted. Het agile dashboard biedt een overzicht over de smart use cases in het project, die in kolommen zijn verdeeld, afhankelijk van de status waarin ze zich bevinden, zie afbeelding 5. In een oogopslag is zo, ook voor de opdrachtgever, te zien hoe ver het project is.

Omdat de smart use cases worden uitgedrukt in smart use case punten, is bij iedere statuswijziging direct na te gaan hoeveel punten (en uren) nog benodigd zijn om het project te voltooien. Een burn down chart toont de dagelijkse momentopname van deze uren, uitgezet in de tijd. Een eenvoudige extrapolatie kan nu de verwachte einddatum van het project calculeren, zie afbeelding 6.
In agile BI-projecten is het overigens niet alleen zeer zinvol een burn down chart te gebruiken voor het gehele project, maar ook om een dergelijke chart per rapportage te projecteren. Juist omdat de rapportages zo snel mogelijk moeten worden opgeleverd, levert dit laatste de opdrachtgever directe informatie over de voortgang en het tijdstip wanneer het nieuwe rapport is in te zetten in het besturen van zijn bedrijfsprocessen.

Afsluitend

De agile karakteristieken, fasen en technieken zoals hier gepresenteerd, spelen bijzonder goed in op de snel wijzigende en uitbreidende wensen en eisen aan BI-projecten. Het toepassen van smart use cases biedt projecten daarnaast een gestructureerde, maar vooral pragmatische manier om in het hele project met eenzelfde eenheid van schatten en werken te opereren. De gereedschappen die in Smart-projecten worden gebruikt om de voortgang te meten, zoals het agile dashboard en de burn down charts per rapport, bieden bovendien direct inzicht in de realisatie van de rapportages en analyses, met snel resultaat tot gevolg, en met snel groeiende tevredenheid van de klant.
En daar was het toch allemaal om te doen?

Zie ook wiki.trinidadplatform.org voor meer informatie over Smart, smart use cases (inclusief stereotypes) en smart estimation.

Sander Hoogendoorn en Sandra Wennemers
Sander Hoogendoorn is Principal Technology Officer bij Capgemini.
Sandra Wennemers is Managing Consultant bij Capgemini.