In de nieuwste versie van DSDM (4.2) is een raamwerk opgenomen voor het gebruik van XP binnen de DSDM aanpak. Beide benaderingen hebben veel gemeen. In mijn praktijk heb ik regelmatig voor praktische zaken uit XP geput binnen een DSDM project. XP is een benadering van software ontwikkeling die is opgezet rondom de ontwikkelaar. DSDM stelt de bruikbaarheid van het product centraal. Als eenvoud een kenmerk van het ware is, dan is voor de combinatie veel te zeggen.

DSDM 4.2 en XP: samen breder inzetbaar

Combinatie van methoden kan zeer effectief zijn

Beiden streven naar minimalisme: Minder functionaliteit is beter zolang het bruikbaar is (DSDM), waar XP zich hard maakt voor eenvoudigere code als de beste optie, zolang het functioneel transparant is (dezelfde tests doorstaat).

Je kunt tegen het ontwikkelproces aankijken als de wagenmaker tegen het wiel (zie kadertekst 'De Chinese wagenmaker'): het wiel en het boek erover zijn niet hetzelfde. Als het om het wiel gaat, wat is dan de waarde van het boek? Zonder feedback van de auteurs is die nihil. DSDM is pleitbezorger van het gebruik van prototypen (liefst geen weggooi-artikel overigens). Zo krijg je feedback aan de hand van het echte product. Natuurlijk krijg je sneller feedback als je eerst het zichtbare stuk bouwt en met de gebruiker afstemt. En zo zijn de *Functional Model Iteration* (FMI, het voor de gebruiker zichtbare deel) en de *Design and Build Iteration* (DBI, het uitwerken van de achterliggende logica) als twee aparte stappen geboren. XP is iets radicaler in zijn aanpak: Stel een story zo goed en zo kwaad als het gaat op, stel tests vast, bouw, integreer met bestaand geheel en krijg en verwerk continue de feedback: een snel lerend proces met relatief kleine stappen. Beide methoden eisen permanente deelname van een gevolmachtigde gebruiker, naast de bouwer in het project. Beide methoden proberen zo snel mogelijk naar het echte deelproduct te komen, waar de meest waardevolle feedback te verkrijgen is. Het verschil is hoofdzakelijk het medium (prototype, story boards) en relevanter dan dat, de volgorde van handelen. Waar DSDM per partijtje functionaliteit eerst functioneel de breedte in gaat, neigt XP juist naar gelijk de diepte in voor zo klein mogelijke stukjes (story's) tegelijk.

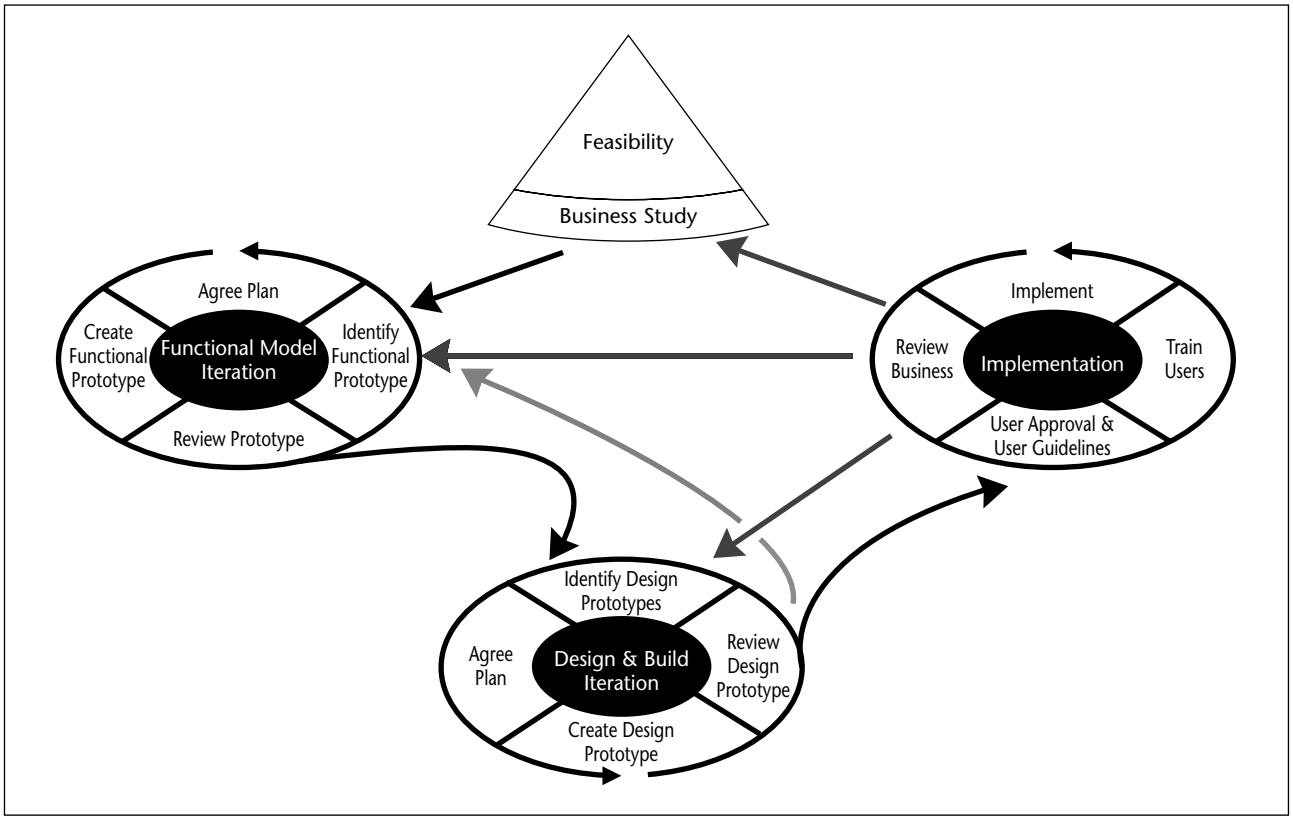
XP (Extreme Programming) XP (Extreme Programming) is een software ontwikkelmethode die enkele best practises rond softwarebouw *extreem* heeft geformuleerd. XP is in beginsel erg vrij en veronderstelt

De Chinese wagenmaker

Software ontwikkeling als communicatieprobleem

Hertog Hwan zat boven in zijn ontvangzaal en las een boek. Beneden in de hof was de wagenmaker P'hien bezig met een wiel. Zijn hamer en beitel neerleggende, ging P'hien de treden op en zei: ‚Ik veroorloof mij uw Hoogheid te vragen wat voor woorden zij leest." De hertog sprak: „De woorden der wijzen." ‚Leven die wijzen nog ?" vroeg P'hien. ‚Zij zijn dood," was het antwoord. „Dan, o heerser," hernam P'hien, „zijn de woorden, die Uw Hoogheid leest, slechts de droesem en het bezinksel der Ouden." De hertog zei: „Hoe kun jij, een wagenmaker, iets hebben aan te merken op een boek, dat ik lees! Kun je nadere uitleg geven, goed: zo niet, dan gaat het om je leven." De wagenmaker sprak: ‚Uw dienaar zal het beschouwen uit een oogpunt van zijn eigen vak. Wanneer ik een wiel maak en ik ga voorzichtig te werk, dan lijkt dat wel aardig, maar de uitkomst is onsterk; als ik hard toesla, is het vermoeiend, en de verbindingen sluiten niet. Als de bewegingen van mijn hand niet te zacht zijn en niet te onstuimig, wordt het denkbeeld in mijn geest verwerkelijkt. Maar ik kan dit niet in woorden brengen; er komt nog een zekere slag bij. Dien slag kan ik mijn zoon niet leren en mijn zoon kan hem niet leren van mij. Daarom: hoewel ik zeventig jaar ben, maak ik nog altijd wielen op mijn ouden dag. Wanneer nu de Ouden met wat zij niet zeggen konden dood zijn en heengegaan, dan moet wat Uwe Hoogheid leest, wel de droesem en het bezinksel der Ouden wezen."

Uit de werken van Tsjwang-Tze ~ door Ir A. J. Blok, Uitg. N. Kluwer, Deventer, bladzijde 48-49.



FIGUUR 1. De klassieke DSDM lifecycle

goede professionals. Er zijn twaalf principes die overeind gehouden worden:

1. Programmeren en testen lopen gelijk op.
2. Continue integratietesten.
3. Pair programming: twee programmeurs achter een toetsenbord. De kans op fouten verminderd drastisch terwijl de focus scherp blijft.
4. Klant met beslisbevoegdheid continue aanwezig op de werkplek.
5. Gezamenlijke planning sessies.
6. Regelmatige oplevering van hoogst geprioriseerde requirements. Deze zijn in de vorm van verhalende tekst vastgelegd, meestal samen met de bijbehorende tests.
7. Normale werklast. Overwerken dient alleen een korte termijn doel.
8. Collectieve verantwoordelijkheid voor de sourcecode. Iedereen kan alles wijzigen, zolang het de bijbehorende tests onveranderlijk blijft doorstaan.
9. Refactoring. Continue proces om sourcecode te vereenvoudigen. Onkruid wieden.
10. Eenvoudig ontworpen code. Samen met een collectieve standaard ondersteund dit het collectieve eigenaarschap.
11. Gebruik een metafoor: de metafoor ondersteunt coherentie van het team en van de releases.
12. En tenslotte: een standaard codeerformaat.

DSDM DSDM kent naast negen principes een procesmodel met productbeschrijving en een rolbeschrijving in en rond een project. Het procesmodel komen we later op terug. De principes zijn:

1. Actieve gebruikersinzet: gebruikers doen echt mee in het project.
2. Beslisbevoegdheid gebruikers binnen de scope bepaald door de requirements op hoog niveau (7).
3. Frequent opleveren van producten: de hartslag van een project.
4. Business-gebruik als meetlat: het gebruik bepaalt het nut van een oplossing.
5. Iteratief en incrementeel ontwikkelen: niets gaat in een keer goed.
6. Alle wijzigingen zijn omkeerbaar; dit is niet alleen een versiebeheer-vraagstuk, maar ook een attitude van alle teamleden.
7. Requirements zijn op hoog niveau vastgelegd: evolutionair ontwikkelen is iets anders dan scope-vervaging.
8. Testen is geïntegreerd in de cyclus.
9. Samenwerking is essentieel, no blame.

THE AGILE MOVEMENT The Agile Movement is een beweging bestaande uit vertegenwoordigers van diverse lichtgewicht methoden, die hun gezamenlijk gedachtegoed hebben weergegeven in het Manifesto for Agile Software Development: We ontsluiten betere soft-

ware-ontwikkelmethoden door het te doen, en anderen te helpen het te doen. Hierbij hebben we geleerd.

- Mensen en interactie meer te waarderen dan processen en tools,
- Werkende software meer dan uitgebreide documentatie,
- Samenwerking meer dan contracten en
- Reageren op veranderingen meer dan het volgen van een plan.

XP en DSDM zijn beide lid van de Agile Movement. Dat binnen deze groep de vraag is gerezen om de diverse geledingen te combineren, met de verwachting daar betere methodes uit te verkrijgen, zal niemand verbazen. Een werkgroep heeft de handschoen met betrekking tot DSDM en XP opgepakt.

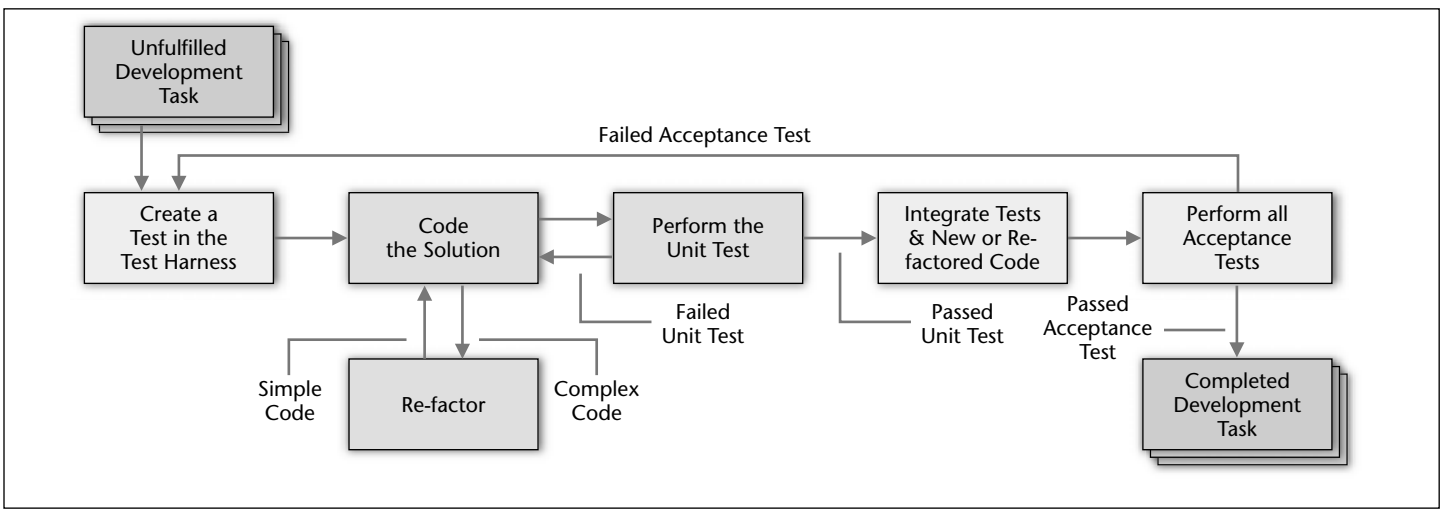
DSDM en XP De combinatie is overduidelijk nuttig wegens het domein van beide methoden, dat op elkaar aansluit: DSDM zwijgt over hoe er gebouwd moet worden, XP is wat stil op het functionele vlak. Maar voordelen gaan verder dan louter dekking van de hele keten tussen requirement en realisatie. XP is sterk in het tijdig opleveren van werkende, goed onderhoudbare software. Het proces is beheersbaar door een goede planning-game, waarin de requirements in de vorm van story's worden geschat op impact en prioriteit. De story met de meeste prioriteit wordt het eerst gebouwd. Prioriteit kan wijzigen gedurende het proces.

DSDM is sterk in zijn omgang met onzekere functionaliteit in een strak tijdsframe. Het proces is beheersbaar omdat het voorziet in specificatieruil en het opleveren van meer of minder functionaliteit. DSDM hangt sterk op de modelleerbaarheid van gebruikersinteractie in (werkende) prototypen. Wat je ziet is wat je hebt gespecificeerd.

In de combinatie zal een breder spectrum requirements kunnen worden verwerkt. Bedenk dat XP ten opzichte van DSDM veel kort-cyclischer van aard is. Wil je een confrontatie met de uitwerking van een klein deel, dan is XP de manier. Heb je een visualisatie nodig van een groter geheel, dan is het DSDM prototype wellicht de optie. Een hybride aanpak waarbij een deel rechtstreeks met XP en een deel via een prototype wordt uitgewerkt is een welkome aanvulling op het gamma.

Nieuwe DSDM manual Hier ga ik wat nader in op de uitwerking die DSDM aan de combinatie met XP heeft gegeven. Uitgangspunt is de projectcyclus van DSDM. In de opstartfase spreekt DSDM over Pre-Project, Feasibility Study en Business Study. XP kent op dit punt geen duidelijke structuur, maar stelt een aantal principes en axioma's ter beschikking. DSDM stelt in deze fasen heel duidelijk geschiktheid van product, methode, projectscope en risico's aan de orde. Aan het eind hiervan is op hoog niveau scope, omgeving en realiseerbaarheid (met DSDM en eventueel XP als aanpak) vastgesteld. Hier kan ook uitkomen dat het product niet, anders of met andere middelen dan DSDM en XP moet worden ontwikkeld. De erkenning dat DSDM niet het wondermiddel voor alles is kan ik alleen maar waarderen als een krachtig instrument van DSDM. In de Business Study worden een aantal producten gemaakt die van belang zijn voor de integratie van DSDM en XP: Business Area Definition, System Architecture Definition, Prioritised Requirements List en Development Plan.

Business Area Definition DSDM legt nadruk op de omgeving waarin het uiteindelijke product functioneert. "The proof of the pudding is in the eating" is een bekend engels spreekwoord. Alleen met het succesvol gebruik, dus optimale aansluiting bij de omgeving, van een product zullen voordelen worden gerealiseerd



Figuur 2. De XP realisatiekring

FIGUUR 3. Voor XP is FMI en DBI een geheel

voor de business. De *Business Area Definition* legt de context en werking van de te bouwen oplossing vast. Daarbij wordt de impact van wijzigingen op de business aangegeven. Deze positionering van het projectdoel in een groter geheel geeft het evolutionaire ontwikkelen een doel, realistische begrenzing en een hogere acceptatiegraad tijdens en na de implementatie.

SYSTEM ARCHITECTURE DEFINITION Met de *System Architecture Definition* komen we op een gevoelig terrein. De vraag in hoeverre een ontwerp vooraf nodig is, is natuurlijk sterk situationeel. Waar XP de neiging heeft te bouwen en feedback af te dwingen, geeft DSDM de voorkeur aan enig ontwerp om risico's te mijden. XP adresseert architectuureisen door:

- Uitstel van architectuurbesluiten, daarmee wijzigingskosten reducerend
- Story's met zware architectuurbesluiten naar voor te halen en
- Geautomatiseerd testen van niet-functionele eisen

DSDM streeft naar een functioneel prototype wat in DBI wordt uitgebreid om niet-functionele eisen te incorporeren. Dit neigt ook naar uitstel van architectuurbesluiten. Echter, DSDM eist een voldoende uitgewerkte software-architectuur om risico's te mijden rond de realisatie van met name niet-functionele eisen. De System Architecture Definition biedt met name bouwers een houvast als technisch raamwerk voor de ontwikkeling van het eindproduct. Andere belanghebbenden kunnen informatie behoefte hebben met betrekking tot hardware, middleware, operating systemen, Databases et cetera.

PRIORITISED REQUIREMENTS LIST Met de *Prioritised Requirements List* legt DSDM, bij aanvang van het project, de scope van het project vast. Alle eisen worden vastgelegd, en volgens MoSCoW principes geprioriteerd. Dat wil zeggen: het proces van vaststellen wat echt nu nodig is en wat net iets minder hard, is in hoge mate gecultiveerd en verankerd in DSDM. In het goed toepassen van dit instrument schuilt een groot deel van het succes van een DSDM project. Het project dat met een te grote verzameling "must haves" begint, kan al snel de oplevering in timeboxen niet meer waarmaken. En dit is één van de pijlers waar DSDM op rust: frequente oplevering van bruikbare producten. De eisen die in deze lijst staan weergegeven moeten voldoende gedetailleerd zijn om de scope van het project vast te leggen en liever niet meer dan dat. Het is dus geen uitputtende opsomming van alle requirements. Verder detailleren gebeurt in de Functional Model Iteration, waar we later op terug komen. XP kent niet expliciet het begrip "minimum usable Subset" (alle "Must Have" requirements) zoals DSDM die hanteert. XP story's laten zich goed met dit instrument prioriteren.

DEVELOPMENT PLAN In het *Development Plan* staan de timeboxen en de bijbehorende producten gepland. Behalve een weergave van opleveringen vermeldt dit plan ook hoe het project beheerst gaat worden (testen, configuratie management, risicobeheer, change control) en ingevuld (taken, bevoegdheden en verantwoordelijkheden).

REALISATIE Realisatie geschiedt in een reeks timeboxen van FMI, DBI en vervolgens implementatie. DSDM kent varianten in de volgorde, maar daar besteden we geen aandacht aan. De eerste cyclus, de *Functional Model Iteration*, moet vooral worden gezien als een belangrijke stap om requirements helder te krijgen. Hier wordt dus vastgelegd met welke functionaliteit de beste invulling gegeven wordt aan de high level requirements die voor de timebox zijn vastgesteld. Het vastleggen hiervan in een "bewegend" prototype of in "statische" story's moet op zijn eigen merites ten aanzien van de inhoud worden gekozen. Een mix is zeer wel denkbaar, omdat interactie weliswaar optimaal in

een werkend prototype wordt vastgelegd, maar er ook requirements zijn die niet in een dergelijk prototype zijn te vatten. In de laatste gevallen is aanvulling met story's (en de bijbehorende testcase) welkom.

De *Design and Build Iteration* levert het Tested System op, het product gereed om in productie gebracht te worden. Het is het volledig geëvolueerde, geteste prototype dat in FMI is opgeleverd. Uiteraard is het uitgebreid met alle nog niet verwerkte eisen, zoals achtergrondtaken en niet-functionele zaken als bijvoorbeeld performance. Voor XP is FMI en DBI een geheel.

REFACTORING XP is expliciet in de toepassing van *Refactoring*. Verander het systeem zodanig dat het ontwerp vereenvoudigd, maar het gedrag ongewijzigd blijft. Een eenvoudig ontwerp van het eindproduct is een uiterst waardevolle kwaliteit die bewezen heeft onderhoudskosten voor lange tijd te reduceren. Daarbij is een eenvoudig ontwerp inzichtelijker. Voor verdere ontwikkeling van het systeem is dat een erg aantrekkelijke eigenschap. Dit proces is een continu proces, en heeft betrekking op alle code die is geschreven in alle cycli, dus zowel FMI als DBI. Voor de DSDM methodologen aanleiding het onderscheid te laten vervagen. Hierdoor is het een reële praktijk FMI en DBI te combineren in een timebox.

TIMEBOX DSDM en XP verschillen in hun omgang met de *Timebox*. DSDM verdeelt de timebox in een drietal iteraties (Investigate, Refine en Consolidate) waarbinnen naar een bruikbare oplossing wordt geëvolueerd. Met de MoSCoW regels zijn de requirements over de timeboxen verdeeld. Het is binnen DSDM niet waarschijnlijk dat alle requirements zullen worden opgeleverd. Wel is zeker dat de "minimum usable subset" wordt opgeleverd (met name door een timebox voor maximaal zestig procent te vullen met "must haves").

De XP timebox is iets meer rechttoe rechtaan: alle story's worden in volgorde van prioriteit gerealiseerd. Doordat de story's als ze groot zijn worden opgebroken, en allen tot een gelijk niveau van planbaarheid zijn gedetailleerd in het "planning-game" zal deze reeks wat beter planbaar zijn, en de inhoud van de timebox wat voorspelbaarder dan die van DSDM. De story's zullen in het algemeen wat homogener zijn met betrekking tot hun grootte. Hierdoor is het begrip "velocity" (aantal gerealiseerd (getest en geaccepteerde) story's per periode een bekende prestatie-eenheid die binnen XP gehanteerd kan worden. Het oppervlakkiger analyse niveau van DSDM staat dit niet toe.

Beide benaderingen hebben hun voor- en nadelen. Voor welke benadering gekozen wordt, hangt af van de situatie en van de mogelijkheden (is alles in story's gevangen in deze timebox of juist niet?) en de daarmee - waarschijnlijk - samenhangende zekerheden en onzekerheden van het project op dat moment. Het is in ieder geval belangrijk een keuze te maken en ambivalentie te vermijden.

OVERIG Voor *Implementation* en *Post-Project* zijn geen bijzonderheden met betrekking tot XP in de nieuwe versie. In de Implementation-fase brengen we het eindproduct bij de klant, leiden op en dragen over. In Post-Project leren we van onze successen en fouten.

CONCLUSIE Methodes staan en vallen met een concrete implementatie. DSDM biedt een raamwerk, zodat over de invulling steeds moet worden nagedacht. Dat is tijdrovend, maar tegelijkertijd een kracht. Met versie 4.2 is een wat concretere mogelijkheid voor het realisatiegedeelte van DSDM geschetst. Het toepassingsgebied voor software-ontwikkeling is - indien gekozen wordt voor XP - duidelijk breder geworden. De DSDM manual kent een waslijst met aandachtsgebieden waar DSDM niet of met speciale aandacht ingezet moet worden. Deze lijst kan met de combinatie XP korter worden, en zal zeker minder vaak het gebruik in een specifiek project ontraden.

Voor de managers, gebruikers en coördinatoren onder ons is het een uitbreiding van zowel aandachtsgebieden als mogelijkheden. Laat het woord extreem je niet afschrikken, het gedachtegoed van XP staat dicht bij dat van DSDM.

Voor de bouwers onder ons is de toepassing van DSDM (met XP) wat concreter geworden. De voordelen van XP blijven behouden in het DSDM raamwerk. Eigenlijk is de combinatie minder extreem dan DSDM en XP ieder voor zich zijn.

Met de combinatie DSDM/XP hebben we een middel om de zoon het wiel van de wagenmaker te laten maken. Eindelijk hebben we voor de klant de oplossing van een al vijfentwintig eeuwen oud probleem...

Dolf Waagmeester is projectmanager bij LogicaCMG, heeft een systeemontwikkelingsachtergrond en heeft daarnaast gebruikersposities bekleed in diverse projecten. Dolf is geaccrediteerd PRINCE 2 trainer en heeft zich verdiept in diverse methodes zoals DSDM, RUP en XP.