overdruk informatie september '00

Eigenschappen van moderne ontwikkelmodellen

Vier modellen vergeleken

Auteurs: Danny Greefhorst en Mark van Elswijk

Eigenschappen van moderne ontwikkelmodellen

Vier methodes vergeleken

Ontwikkelmodellen zijn raamwerken die aangeven hoe ontwikkeltrajecten kunnen worden ingericht. Ze omschrijven welke fasen, activiteiten en producten er bestaan en welke personen of rollen hieraan invulling dienen te geven. Er is de laatste jaren een duidelijke trend waarneembaar naar lichtere en iteratieve ontwikkelmodellen. Deze ontwikkelmodellen zijn ontstaan uit ontevredenheid met oude modellen, die te bureaucratisch zijn, die uitgaan van stabiele requirements en die veronderstellen dat het verloop van een ontwikkeltraject precies te voorspellen is.
De auteurs geven in dit artikel een beknopt overzicht van een aantal moderne ontwikkelmodellen en hun belangrijkste karakteristieken.

Danny Greefhorst

Mark van Elswijk

Iterative application development

Iterative application development (IAD) is een methode die de gebruiker en de ontwikkelaar als gelijkwaardige partners ziet. De I in IAD staat naast iteratief (ontwikkelen in een aantal rondjes) ook voor interactief (samen met de gebruiker) en voor incrementeel (stapsgewijs uitbouwend tot een steeds grotere bruikbaarheid).
IAD bestaat uit drie iteratieve fasen (figuur 1). Het doel van de *definitiestudie*fase is de analyse van de doelen, beperkingen en requirements, op basis waarvan een systeemconcept en een pilotplan worden opgesteld. Het pilotplan geeft aan welke pilots (incrementen) in welke iteratie worden ontwikkeld. In de *pilotontwikkeling*fase worden het ontwerp en de implementatie van het systeem uitgewerkt. De uiteindelijk resulterende pilot wordt in de fase *invoering* operationeel gemaakt, geaccepteerd en ingevoerd in de organisatie.

Figuur 1: IAD

Dynamic systems development method

De dynamic systems development method (DSDM) is in 1994 ontstaan doordat een aantal bedrijven in het Verenigd Koninkrijk behoefte had aan een standaardontwikkelmodel voor rapid application development (RAD). Het DSDM-consortium, waarin inmiddels meer dan 1000 bedrijven deelnemen, heeft ervoor gezorgd dat DSDM in korte tijd dé de facto standaard voor RAD is geworden.
Het idee achter DSDM is eenvoudig: ontwikkeling is een groepsinspanning die de kennis van domeinexperts combineert met de technische kennis van IT-professionals. DSDM bestaat uit vijf fasen, ook wel bekend als 'de drie pizza's en een kaas' (zie figuur 2). Het project doorloopt het *haalbaarheidsonderzoek* en het *bedrijfsonderzoek* sequentieel, gevolgd door drie iteratieve fasen. Het haalbaarheidsonderzoek bepaalt of DSDM de juiste aanpak is voor een specifiek project. Het bedrijfsonderzoek brengt het bedrijfsmodel in kaart. De focus van de *functioneel model*iteratie is de verfijning van de bedrijfsaspecten van het systeem. Het daadwerkelijk bouwen van het systeem vindt plaats in de iteratie *ontwerp en bouw*. De implementatiefase ten slotte dekt de overdracht van ontwikkeling naar operationele omgeving.

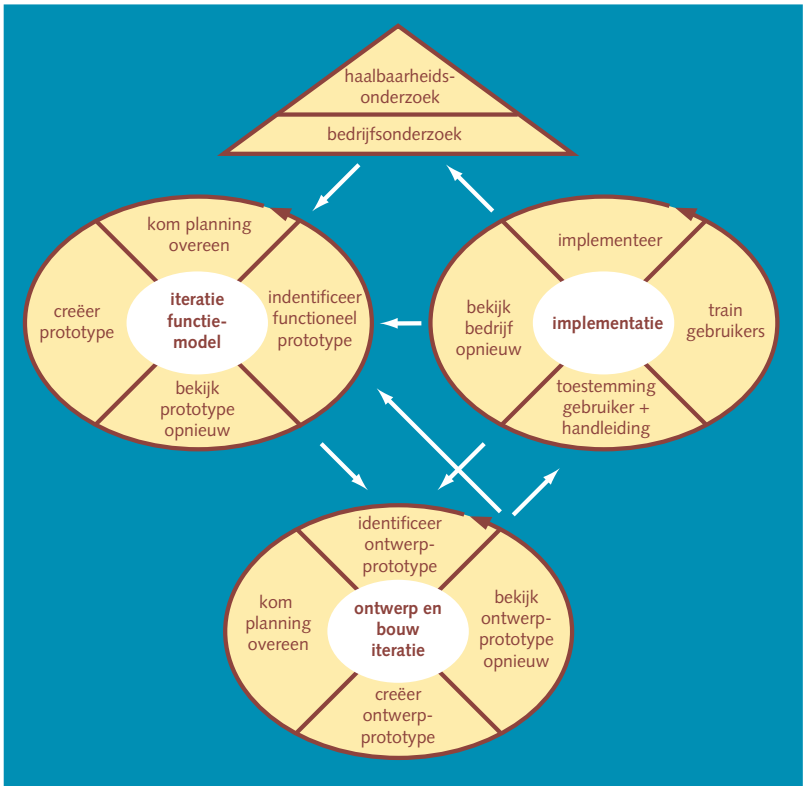
Rational unified process

Het rational unified process (RUP) komt voort uit verschillende objectgeoriënteerde ontwikkelmethoden. De belangrijkste voorouder is Objectory, een methode voor softwareontwikkeling die eind jaren tachtig is ingevoerd.
Een RUP-proces bestaat uit het achtereenvolgens uitvoeren van cycli, waarbij elke cyclus wordt afgesloten met de oplevering van een product. In feite levert elke cyclus een nieuwe versie van het te bouwen systeem op, inclusief verschillende modellen, documenten en diagrammen. Een cyclus is onderverdeeld in vier fasen: aanvang, detaillering, constructie en transitie. Deze fasen geven de voortgang van de cyclus aan, die begint bij het uitwerken van een idee en de eerste functionele requirements en eindigt bij een concreet softwareproduct, bijvoorbeeld in de vorm van een bètaversie.

Extreme programming

Extreme programming (XP) is ontstaan in de Smalltalk-wereld en is ontwikkeld in de jaren tachtig en negentig. Sinds eind jaren negentig is het een volwaardige methodologie. In diezelfde jaren begint een trend naar het inzetten van 'lichte' ontwikkelmethoden. De grote, zware ontwikkelmethoden blijken in de praktijk niet het gewenste resultaat te leveren.
XP beschrijft een aantal fasen waaruit een project bestaat, maar schrijft ze niet voor. Elk project – zo is de filosofie – zal (en misschien moet) anders zijn dan andere projecten. Globaal begint elk project met een ontdekkingsfase, waarin programmeurs ervaring opdoen met de te gebruiken tools, en experimenteren met technologie en architectuurideeën. De gebruiker (klant) experimenteert ondertussen met het schrijven van 'verhalen', die later nodig zijn voor het vastleggen van de requirements. Na de ontdekkingsfase volgt de planningfase, waarin de klant en de programmeurs afspraken maken over de datum waarop de eerste requirements

Danny Greefhorst en Mark van Elswijk zijn beiden als senior consultant werkzaam bij het Software Engineering Research Centre (SERC) te Utrecht.



Figuur 2: DSDM

beschreven zullen zijn. Deze fase duurt (mits voorbereid in de voorgaande fase) een à twee dagen. Daarna volgen de iteraties van ontwikkeling, die elke tussen de een en vier weken duren. Tijdens elke iteratie worden de belangrijkste requirements aangepakt. De klant heeft hierbij de bevoegdheid prioriteiten te stellen. Na oplevering van het product begint het onderhoud. Vanuit XP gezien bevindt het project zich bijna vanaf het begin in de 'onderhoudsfase'.

De eigenschappen

Requirements

Alle methoden besteden veel aandacht aan requirements en het verfijnen daarvan in de loop van het project. Tijdens iteraties is er steeds tijd en aandacht voor het preciezer vaststellen van de requirements. De requirements worden in nauwe samenwerking tussen klant en ontwikkelaar bepaald en geprioriteerd. Voor het bepalen van deze prioriteiten kunnen de MoSCoW-regels worden gebruikt. In ieder geval moeten de 'Must have'-requirements in een increment worden gerealiseerd, gevolgd door de 'Should have'- en 'Could have'-requirements. De 'Would be nice to have'-requirements zullen slechts incidenteel worden gerealiseerd. Voor het aansturen van prototyping-activiteiten is het van belang periodiek een baseline van de requirements te maken.

De cruciale requirements worden in **IAD** opgesteld en geprioriteerd tijdens de pilotplanworkshop in de definitiestudiefase. In **DSDM** worden de requirements op hoog niveau tijdens het bedrijfsonderzoek vastgelegd. **RUP** gebruikt usecasediagrammen om de requirements vast te leggen. Deze diagrammen beschrijven (schematisch) de gewenste functionaliteit, gezien vanuit gebruikers. Over de wijze waarop de diagrammen totstandkomen, schrijft RUP niets voor. In **XP** wordt tijdens elke iteratie een aantal requirements uitgewerkt. De klant geeft daarbij de prioriteiten aan.

Iteratief en incrementeel

Alle hier behandelde methoden werken op iteratieve en incrementele wijze. Een iteratie bevat een aantal vaste activiteiten, en een project bestaat uit het meermalen doorlopen van iteraties. Aan het eind van een iteratie is een nieuwe versie van het product beschikbaar (een 'increment'), die voldoet aan minstens een deel van de requirements. Doordat een systeem in incrementen wordt opgeleverd, kunnen er continu tastbare producten worden getest. Het aantal iteraties dat nodig is om een eindproduct te maken, is onder meer afhankelijk van de omvang en complexiteit van de te bouwen software.

In **IAD** worden de fasen definitiestudie, pilotontwikkeling en invoering iteratief doorlopen. IAD onderkent vier varianten van de ontwikkelcyclus: evolutionair ontwikkelen, incrementeel opleveren, big-bang-invoeren en incrementeel ontwikkelen. In **DSDM** worden de iteratie functioneel model, de iteratie ontwerp en bouw en de implementatie iteratief doorlopen. Hoe de drie fasen overlappen en samenkomen, wordt per specifiek project bepaald. In **RUP** wordt een iteratie gezien als een volwaardig project. Aan het begin van een iteratie bepalen de ontwikkelaars het doel, gebaseerd op één of meer usecasediagrammen, en proberen daarbij de belangrijkste risico's af te dekken. Tijdens een iteratie in **XP** worden de belangrijkste requirements (geselecteerd door de klant) aangepakt en ingevuld. Een iteratie kan niet eerder eindigen dan wanneer alle tests volledig en succesvol zijn uitgevoerd.

Gebruikersparticipatie

Moderne ontwikkelmethoden onderkennen het belang om gebruikers te betrekken bij het opstellen van een systeem. Niet alleen zorgt dit voor bredere acceptatie, het faciliteert tevens de aanpassing van de eisen aan de hand van nieuwe inzichten en vergroot de kans dat het uiteindelijke product ook daadwerkelijk het juiste probleem oplost. Voorwaarde is wel dat gebruikers worden betrokken die genoeg inzicht hebben in het doel van het systeem, maar die tevens detailkennis hebben van de bedrijfsprocessen en de daarvoor benodigde informatie. Daarnaast is

voor een snel en goed verloop de beslissingsbevoegdheid van de betrokken gebruikers belangrijk. Het management moet de gebruikers hierbij voldoende ruimte voor onderhandeling bieden.

Gebruikers worden in **IAD** optimaal bij het ontwikkelproces betrokken door formatie van zogenaamde U-teams (gebruikers), die sterk samenwerken met de A-teams (ontwikkelaars). Gebruikersparticipatie is ook een van de belangrijkste principes van **DSDM**. **RUP** doet weinig uitspraken over de wijze waarop gebruikers in het proces betrokken dienen te worden. Ze zijn in elk geval medeverantwoordelijk voor het opstellen van de usecasediagrammen (de specificaties). De klant, die de rol van gebruiker speelt, heeft binnen een **XP**-project een prominente plaats. Hij is vanaf het begin bij het project betrokken en bepaalt de prioriteiten binnen de ontwikkeling. Daarnaast schrijft hij speciale tests voor die moeten worden uitgevoerd om te bepalen of het softwareproduct voldoet aan de requirements.

Samenwerking

Joint application design (JAD)-workshops zijn een goed middel om binnen zeer korte tijd overeenstemming te bereiken tussen gebruikers en ontwikkelaars. Workshops verminderen in sterke mate de hoeveelheid geproduceerd papier, zorgen voor een betere, realistische afstemming en verankeren alle deelnemers optimaal in het ontwikkelproces.

IAD wordt gekarakteriseerd door een veelvuldige toepassing van workshops, waarin systeemontwikkelaars, opdrachtgever en diverse categorieën gebruikers het informatiesysteem in nauwe interactie specificeren en zelfs gedeeltelijk ontwikkelen. Samenwerking tussen alle betrokken partijen is een belangrijk principe binnen **DSDM**. Workshops kunnen op elk moment in het ontwikkelproces worden toegepast. **RUP** is een modelgedreven methode die voorschrijft welke modellen en diagrammen van belang zijn. De methode geeft daarbij wel aan wie bij het opstellen ervan betrokken dient te worden, maar doet weinig uitspraken over de wijze waarop dat moet gebeuren. Eén van de peilers waarop **XP** rust, is het zogenaamde 'pair-programming'. Dat betekent dat programmeurs getweeën werken achter hetzelfde beeldscherm. Ze kijken met elkaar mee en nemen om beurten het initiatief om code te veranderen of toe te voegen. De programmeerteams zitten dicht bij elkaar, het liefst in dezelfde ruimte, om de communicatie zo efficiënt mogelijk te laten verlopen.

Prototyping

Prototyping versnelt de communicatie tussen gebruikers en ontwikkelaars doordat er wordt gecommuniceerd in een taal die gebruikers begrijpen. Een werkend model van een systeem zegt immers veel meer dan een document of een diagram.

Prototyping wordt in **IAD** gedurende alle fasen van de systeemontwikkeling toegepast. Prototypes kunnen tijdens workshops met gebruikers worden opgesteld, maar kunnen ook al eerder gedeeltelijk zijn uitgewerkt. In **DSDM** wordt prototyping primair op een evolutionaire manier gebruikt, waarbij evoluties leiden tot het uiteindelijk opgeleverde systeem. Prototyping kan in vrijwel alle fasen van DSDM worden toegepast, maar zal voornamelijk tijdens de functioneel model- en de ontwerp en bouwiteratie plaatsvinden. Er worden verschillende soorten prototypes onderkend: bedrijfsprototypes, bruikbaarheidsprototypes, performance/capaciteitsprototypes en ontwerpprototypes. Vaak worden ze in combinatie met elkaar gebruikt. Het maken van prototypes is geen vast onderdeel van **RUP**, maar prototypes kunnen wel worden ingezet. In de meeste gevallen gaat het om prototypes ter illustratie van bijvoorbeeld een gebruikerinterface of een nieuw systeemconcept. Na gebruik worden ze weggegooid.

nieuwe ontwikkelmodellen schenken veel aandacht aan gebruikersparticipatie

XP spreekt niet over prototypes. Omdat de methode productgedreven is, is het echter heel goed mogelijk een prototype te bouwen. Omdat de belangrijkste requirements het eerst worden behandeld, is al betrekkelijk snel een applicatie (prototype) beschikbaar die geschikt is voor de beeldvorming over het uiteindelijke programma.

Testen

Testen maakt een integraal deel uit van moderne ontwikkelmethoden; niet zozeer als aparte fase, maar meer als een continue activiteit. Hierdoor kunnen problemen snel worden gesignaleerd, en wordt voorkomen dat het testen door tijdgebrek wordt ingekort.

IAD definieert een speciale interactieve beoordeling en een testworkshop, waardoor de noodzaak tot gedetailleerde specificaties vervalt. De acceptatie van het pilotdeel in de acceptatieworkshop van de invoeringsfase is dan ook vaak

Literatuur

Beck, K.: *Extreme Programming Explained: Embrace Change*, Addison-Wesley Pub Co, October 1999.

Jacobson, I., J. Rumbaugh en G. Booch: *The Unified Software Development Process*, Addison Wesley, January 1999.

Stapleton J.: *Dsdm Dynamic Systems Development Method: The Method in Practice*, Addison-Wesley Pub Co, August 1997.

Tolido, R.J.H.: *IAD – Het evolutionair ontwikkelen van informatiesystemen*, Academic Service, 1996.

niet meer dan een formaliteit. Zowel in **DSDM** als in **XP** worden componenten zo snel mogelijk door ontwikkelaars en gebruikers getest. DSDM kent daarnaast zowel integratie- als regressietesten. Tests zijn in XP één van de fundamenten van de methode en moeten voor elke verandering worden gecontroleerd. Ze worden zoveel mogelijk automatisch uitgevoerd door testcode, die door de ontwikkelaars is geschreven. De hoeveelheid tijd die in **RUP** aan testen besteed wordt, neemt in de loop van het project relatief gezien toe.

Configuratiebeheer

Configuratiebeheersystemen richten zich op het beheren van versies van allerlei soorten softwareproducten, zoals broncode en documentatie. Hierbij worden ook relaties tussen elementen in kaart gebracht om de impact van wijzigingen op het systeem te kunnen bepalen. Iteratieve werkwijzen stellen nieuwe uitdagingen op het gebied van configuratiebeheer. Zo worden elementen vaak aangepast, kunnen elementen tegelijkertijd worden aangepast en dienen vaak meerdere personen over wijzigingen te worden geïnformeerd.

In **IAD** wordt onderkend dat softwareconfiguratiebeheer onderdeel is van de bredere discipline van het beheren en besturen van informatiesystemen. Een belangrijk principe in **DSDM** is de omkeerbaarheid van wijzigingen tijdens de ontwikkeling. Dit betekent dat er, als een verkeerde richting is ingeslagen, teruggekeerd moet kunnen worden naar een bekend, veilig punt in de ontwikkeling. Uiteraard is gedisciplineerd configuratiebeheer hierbij onontbeerlijk. Alles wat geproduceerd wordt dient in een configuratiebeheertool te worden opgeslagen. **XP** stelt het gebruik van tools voor versie- en configuratiebeheer verplicht, maar dit is met name bedoeld voor integratie. 'Terugkeren' naar oude versies is in deze methode zeer ongebruikelijk. **RUP** besteedt geen nadere aandacht aan configuratiebeheer.

Documentatie en modellen

Moderne methoden, en lichte methoden in het bijzonder, richten zich voornamelijk op activiteiten en code, en in mindere mate op documenten.

In **IAD** wordt bij elke deelactiviteit aangegeven welke soorten documentatie er worden opgeleverd. Het maakt tevens een onderscheid tussen documentatie die is opgesteld volgens een traditioneel ontwikkelmodel en documentatie die van belang is voor objectgeoriënteerde ontwikkelmodellen. IAD kan daarom als een lichtgewicht objectgeoriënteerde ontwikkelmethode worden gezien. **DSDM** beschrijft welke soorten producten worden opgeleverd in de verschillende fasen. Hierbij wordt, naast een aantal kwaliteitscriteria, aangegeven wanneer een product wordt gemaakt, wie verantwoordelijk is voor de acceptatie en wat het doel is. Zowel DSDM als IAD geeft niet aan hoe producten precies totstandkomen en er uitzien. Gezien de historie van **RUP** is het niet verwonderlijk dat deze methode verschillende soorten modellen voorschrijft, waar ze aan moeten voldoen en wanneer ze worden opgesteld. **XP** schrijft geen vorm van documentatie of modellen voor en legt het zwaartepunt op de broncode. In tegenstelling tot modellen legt broncode de bedoeling van de software eenduidig vast. De broncode op zichzelf moet voldoen aan een vooraf bepaalde, afgesproken stijl.

Nieuwe iteratieve en incrementele ontwikkelmodellen schenken veel aandacht aan de participatie van gebruikers in het ontwikkeltraject. Ook is er continue aandacht voor de requirements en het testen. Daarnaast lijkt de focus steeds meer te liggen op code dan op documenten. Opvallend is dat nieuwe, lichte methoden niet proberen het verloop te voorspellen, maar aanpassing van het proces zelf onderdeel maken van dat proces.