Dennis Ham

Applicatieontwikkeling met OBA

SNEL APPLICATIES BOUWEN OP BASIS VAN OFFICE SYSTEM

In dit artikel gaat Dennis Ham in op Office Business Applications, kortweg OBA. Hij geeft aan wat de aandachtspunten, mogelijkheden en onmogelijkheden hiervan zijn. Bovendien geeft hij aan waar nog programmeerwerk nodig is om toekomstbestendige systemen te creëren en geeft hij handvatten hoe dit vanuit de architectuur op verantwoorde wijze gedaan kan worden.

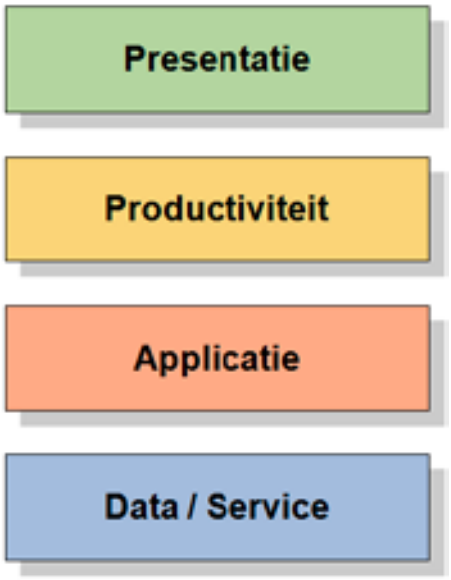
Al lange tijd grijpen ontwikkelaars bij het ontwikkelen van een nieuwe applicatie naar hun ontwikkelomgeving en gaan ze in Cobol, C++, Java, C# of Visual Basic de gewenste applicatie programmeren. De reden dat veel .Net-ontwikkelaars standaard naar Visual Studio grijpen, komt omdat hun kennis van programmeertaal vaak zeer goed is. In elk geval zijn ze er bekend mee en voelen ze zich daarin comfortabel.
Een applicatie bouwen met out-of-the-box producten, zoals in het 2007 Microsoft Office system, vereist veel minder programmeerkennis, maar des te meer kennis van producten die zich in het pakket bevinden. Deze producten zijn zeer rijk. Als je weet hoe de verschillende features aangezet kunnen worden, kun je er erg veel mee en zijn de producten efficiënt te gebruiken. Doordat bij hergebruik van componenten vaak naar de eigen code library wordt gegrepen, valt het optimaal gebruiken van de functionaliteit van de Microsoft Office en SharePoint-functionaliteit buiten de boot. Hierdoor worden onnodige arbeidskosten gemaakt en staat de time-to-market onder druk.

Office Business Applications

Met de komst van 2007 Office System zijn er veel mogelijkheden bijgekomen om bedrijfsspecifieke applicaties te creëren, waarbij geen of bijna geen programmeerwerk meer komt kijken. De applicatie kan door middel van standaard Office-componenten in elkaar gezet worden. Microsoft noemt dit Office Business Applications, kortweg OBA. Een OBA is een oplossing die erop is gericht Information Workers te laten samenwerken gebruikmakend van (één of meer) bekende Microsoft Office-servers en client-applicaties, waaronder Excel, Outlook, SharePoint en PerformancePoint Server.
Wat krachtig is aan een OBA is dat de eindgebruikers door middel van de bekende interfaces (zoals Excel of SharePoint) toegang kunnen krijgen tot belangrijke Line of Business (LOB) informatie, zonder kennis te hebben van LOB-systeem zelf. Deze zijn door de bank genomen alleen direct toegankelijk voor een paar specialisten. Voor een veel groter aantal medewerkers is deze informatie belangrijk voor het uitvoeren van hun dagelijkse werkzaamheden. Met een OBA kunnen Information Workers zelf de informatie ophalen in bijvoorbeeld Excel om daarna verder te analyseren, op basis waarvan beslissingen kunnen worden genomen.
Het is een nieuwe manier van applicaties maken die aansluit bij de software-architectuureisen van deze tijd en de toekomst. OBA-applicaties zijn opgebouwd volgens een multi-tier model, maken gebruik van open XML-formaten en haken naadloos in op de uit te breiden gebruikersinterface van de Office-client. Bovendien is ook het advies aan klanten anders. In een .Net-ontwikkeltraject kon je aan bijna elke wens van de klant voldoen, zolang die maar bereid is de prijs voor die flexibiliteit te betalen. Het voordeel van een OBA is dat je veel sneller en beter de 80 procent (of misschien wel de 95 procent) oplossing kunt bouwen. Met andere woorden, de achterliggende business case is over het algemeen veel gezonder dan bij traditionele custom build-systemen. Het is altijd mogelijk het resterende deel van de oplossing door middel van custom build-componenten of services toe te voegen wanneer dit echt noodzakelijk is.

Config Buy Build

Veel klanten stellen mij de vraag: “Wanneer moet ik nu kiezen voor out-of-the-box ontwikkelen en wanneer voor custom build?” Aan deze twee keuzemogelijkheden kan nog een derde logische optie worden toegevoegd: functionaliteit (bijvoorbeeld een SharePoint-webpart of een webservice) aankopen die op de markt beschikbaar is. Dit is doorgaans goedkoper dan zelf de component ontwikkelen. Daarbij is mijn ervaring dat de meeste functionaliteit die wordt gevraagd helemaal niet specifiek is voor die organisatie, maar dat het vaak om zaken gaat waar veel meer organisaties behoefte aan hebben. Natuurlijk is het voor het garanderen van de stabiliteit van de omgeving waarin de component wordt gebruikt, noodzakelijk dat ook deze componenten zorgvuldig door een testcyclus heen gaan. Uiteindelijk zijn er dan drie keuzes: Config, Buy, Build. Onder Config wordt verstaan dat het systeem kan worden gebouwd door puur de features van out-of-the-box software ’bij elkaar te klikken’ binnen de software die in de organisatie beschikbaar is. Dit vereist een grondige kennis van de producten die onderdeel uitmaken van het systeem. Het kan altijd voorkomen dat een bepaalde gevraagde functionaliteit niet beschikbaar is in de standaardoplossing, maar dat de functionaliteit zo belangrijk is dat deze wel gerealiseerd moet worden. Op dat moment is het interessant om eerst te kijken wat er in de markt te verkrijgen is (Buy) en of die aan de wensen voldoet. Het is natuurlijk altijd mogelijk zelf een component of service te ontwikkelen die niet op de markt zijn te verkrijgen (Build), of waarvan de requirements zo specifiek zijn dat andere oplossingen geen soelaas bieden. Zoals bekend, zijn hierbij de mogelijkheden oneindig (anders dan bij aangekochte software), maar de kosten van het ontwikkelen zijn doorgaans hoger dan bij het aankopen van een al bestaande component.
Indien gebruikgemaakt wordt van een OTAP-ontwikkelstraat moet je vooraf te bepalen wat de eisen zijn aan deployment en weten wat de consequenties zijn van de genomen keuzes tijdens de bouw van het project. Bijvoorbeeld als het verplicht is om software te deployen, is het noodzakelijk na te gaan of alle onderdelen van de applicatie door middel van deployment packages op de verschillende omgevingen gedeployed kunnen worden. Het kan zijn dat een workflow bijvoorbeeld wel qua functionele requirements gebouwd



Afbeelding 1. Applicatie-architectuurlagen

kan worden met een tool als SharePoint Designer.
De deployment van deze workflows wordt momenteel echter niet ondersteund door Microsoft, waardoor het toch noodzakelijk kan zijn de workflow in Visual Studio te ontwikkelen, omdat die workflows wel als feature gedeployed kunnen worden. Dit soort overwegingen moet in de architectuurfase meegenomen worden en onderdeel uitmaken van de keuze tussen Config, Buy en Build.

OBA-architectuur

Het is raadzaam een meerlagige applicatiestructuur te gebruiken bij het ontwikkelen van een OBA. Enterprise architectures behelzen doorgaans drie lagen. Het enige wat hierin ontbreekt voor een OBA is een laag waarin de interacties tussen gebruikers van de applicatie, die nodig zijn om een businessproces te voltooien, worden vastgelegd. Om deze reden is het voor een OBA-architectuur noodzakelijk een extra laag toe te voegen: de productiviteitslaag.
In de presentatielaag worden de Office client-applicaties (Word, Excel, PowerPoint, InfoPath, enzovoort) en webbrowser gebruikt om de gebruikers het systeem te laten gebruiken.
In de productiviteitslaag worden services gebruikt die direct voorhanden zijn voor eindgebruikers in bijvoorbeeld in SharePoint 2007, waaronder document-libaries, lijsten, dashboards, maar ook Forms Services en Excel Services.
In de applicatielaag treffen we onderliggende services aan die niet direct door eindgebruikers te gebruiken zijn, maar wel noodzakelijk zijn voor het juist functioneren. Hieronder vallen bijvoorbeeld workflows, webservices en legacy-applicaties.
In de data / service-laag zitten de datastores en externe services, zoals identity management.
De productiviteitslaag is de onderscheidende laag van een OBA. Hier worden de Office Server-producten ingezet om het samenwerken van Information Workers maximaal te laten gebeuren. De nadruk van een OBA ligt dan ook vooral op deze laag, samen met de presentatielaag: het ontsluiten en het delen van informatie, zodat eindgebruikers in staat worden gesteld om samen businessprocessen te voltooien.

Voorbeeld 1: Formulier met workflow

Een voorbeeld van een vaak toepasbare, mogelijke OBA-applicatie is deze: een InfoPath 2007-formulier dat via Forms Services, wat een onderdeel is van Microsoft Office SharePoint Server 2007 Enterprise Edition, wordt ontsloten. Het voordeel om het InfoPath-formulier te ontsluiten via Forms Services is dat het een web-enabled formulier wordt, waardoor het niet noodzakelijk is om InfoPath 2007 op alle client-computers te installeren. Het InfoPath-formulier haalt via webservices informatie op uit onderliggende systemen. Dat kunnen onder meer databases, ERP-systemen, Active Directory, SharePoint, enzovoort zijn. Als het ingevulde formulier vervolgens door de gebruiker wordt opgeslagen en verzonden, wordt er een workflow getriggerd. Afhankelijk van de inhoud van het formulier doorloopt het formulier de workflow en zijn er mensen die hun goedkeuring moeten geven. Uiteindelijk, als het formulier de definitieve status heeft gekregen dat het is goedgekeurd, wordt er informatie naar een CRM-systeem gestuurd waar de aangevraagde actie kan worden opgevolgd en wordt de aanvrager via een e-mail op de hoogte gesteld dat de aanvraag is goedgekeurd.
Afhankelijk van de complexiteit van de verschillende onderdelen, bijvoorbeeld het formulier en de businesslogica die daar in zou kunnen zitten, de gebruikte datastores of de workflow, zou bekeken moeten worden of hier custom build-componenten moeten worden ontwikkeld, of dat dit allemaal door middel van het configureren van standaard onderdelen gerealiseerd kan worden. Zoals eerder gesteld is het ook afhankelijk van het deployment-model en wat daar de eisen aan zijn in de organisatie. Of dit een vorm is van Config, Buy, of Build, is afhankelijk van hoe deze oplossing gerealiseerd kan worden.

Voorbeeld 2: Duet

Duet is een andere implementatie van een Office Business Applicatie. Duet is een product dat is ontwikkeld door Microsoft en SAP. Na installatie is het voor eindgebruikers mogelijk om via de Office System-applicaties als Outlook en Excel informatie in SAP te bekijken en te bewerken. Dit is dus een vorm van een Office Business Applicatie die onder het kopje Buy, van het Config, Buy, Build-principe valt. Na installatie biedt Duet momenteel elf zogenaamde scenario's, waaronder Time Management, Sales Management en Contract Lifecycle Management.
Het werkt als volgt. In het geval van bijvoorbeeld Time Management, zet de gebruiker nog steeds zijn afspraken in Outlook. Maar na installatie van Duet is er in Outlook een extra werkblak te zien en wanneer de gebruiker bij het inplannen van een afspraak in Outlook in die werkbalk aangeeft dat de afspraak een 'Duet'-item is, dan komen er meteen een aantal extra velden in beeld. Velden zoals projecten die in SAP bekend zijn en waarop de gebruiker zijn uren kan boeken. Uiteindelijk wordt de tijd in de agenda ook in SAP geboekt op het projectnummer dat in Outlook is gekozen.
Een ander scenario biedt de mogelijkheid rapporten uit SAP te halen die vanuit Outlook zijn te benaderen en vervolgens in Excel zijn te bewerken.
Het voordeel van deze integratie tussen Office en SAP is dat de eindgebruiker zijn oude vertrouwde interface niet hoeft te verlaten en ook geen andere manier van werken hoeft aan te leren. Ook hoeft hij geen kennis van SAP of de SAP UI te hebben om toch met het systeem te kunnen omgaan. Ook hier zijn beperkingen zoals bij elke pakketimplementatie. Een duidelijke beperking is dat alleen

Afbeelding 2. Voorbeeld van een OBA-applicatie: formulier met workflow

de geboden scenario's gebruikt kunnen worden en dan eigenlijk ook alleen nog maar op de manier zoals deze door Microsoft en SAP zijn opgezet. Voor het merendeel van de gebruikers waarvoor dit is bedoeld, is dat geen beperking. Die willen alleen de voor hen noodzakelijke informatie in SAP stoppen of eruit halen en meer niet. Zogenaamde power users hebben niet genoeg hieraan. Zij zullen de SAP-schermen moeten blijven gebruiken om hun dagelijkse werkzaamheden in SAP te kunnen uitvoeren.

Schone taak

Custom development is met de komst van OBA niet meer de enige manier van ontwikkelen van applicaties en daarmee ook niet altijd de meest optimale. Met OBA kunnen applicaties worden gebouwd die gebruikmaken van de rijke functionaliteit die in het Microsoft Office System-platform beschikbaar is. Veel van de benodigde functionaliteiten die in bedrijven vereist zijn, zoals het feit dat het tegenwoordig voor veel businessprocessen noodzakelijk is om gebruikers te laten samenwerken, worden zo op een presenteerblaadje aangeboden. Een OBA voorziet daarin met de genoemde productiviteitslaag. Het is wel belangrijk over voldoende productkennis te beschikken, zodat ontwikkelde componenten herbruikbaar zijn. Hier ligt een schone taak voor zowel de softwarearchitect, de functioneel verantwoordelijke persoon in het project en niet in de laatste plaats de softwareontwikkelaar, om dit in goede banen te leiden.

Dennis Ham is als Microsoft Solutions Architect werkzaam bij Giraffe IT (www.giraffe.nl). Zijn interesse ligt vooral op het gebied van Microsoft Office SharePoint Server 2007 en Duet. Voor vragen en opmerkingen is hij te bereiken via dennis.ham@giraffe.nl.

Referenties

OBA-product informative: www.microsoft.com/oba
MSDN over OBA: msdn.microsoft.com/oba
OBA Central: www.obacentral.com
Duet: www.duet.com