VX Company deed een .NET-project bij een woningcorporatie. Het project begon eigenlijk met noodzaak om bij de fusie van twee woningcorporaties applicaties te schiften en te integreren, daarna het verzoek een ontwikkelstraat op te zetten, maar mondde uiteindelijk uit in een volwaardige portal met applicatiekoppelingen via webservices. Software Release Magazine sprak met Robin C.J. Stolle, de teamleider en informatieanalist en ir. Reinhard P. Brongers, architect van het project.

Van ontwikkelstraat naar portal

.NET-project bij woningcorporatie

Het project is ontstaan uit de fusie van twee woningcorporaties. Beide hadden hun eigen software-verzameling. Op korte termijn was er behoefte aan een portaal, waarin alle informatiebehoeften die elke medewerker van een woningbedrijf zou kunnen hebben met een druk op de knop terug te vinden zou zijn.

De oude applicaties bestonden uit klassieke ASP- en Paradox-applicaties en daarnaast uit deels zeer gecompliceerde en moeilijk onderhoudbare spreadsheets. Al met al waren er heel veel verschillende oude applicaties en spreadsheets. Daardoor ontstond echter de behoefte om een afdeling op te zetten binnen het woningbedrijf waar allerlei applicaties konden worden ontwikkeld volgens betere standaarden dan voorheen. De bouw daarvan werd uiteindelijk deels door mensen van VX-company gedaan en deels door mensen van het woningbedrijf. Uiteindelijk moeten de IT-ers van de woningcorporatie volledig de ontwikkelstraat gaan bemannen.

Brongers: 'Je kunt je voorstellen dat wanneer twee corporaties samengaan veel software elkaar overlapt. Uiteindelijk is een keuze gemaakt waarbij sommige applicaties van de ene, en andere van de andere corporatie gebruikt zijn. We hadden niet de illusie dat het allemaal standaardpakketten zouden kunnen worden, het wordt doorgaans toch maatwerk. Het *core business* systeem is overigens wel een aangekocht product.'

Stolle: 'Oorspronkelijk waren er plannen een nieuw product te implementeren. De producent ervan is er echter niet in geslaagd dat goed en op tijd voor elkaar te krijgen. De *pilot* waar ze mee bezig waren die zouden ze afmaken en alle ander *pilots* werden teruggeschoven. De woningcorporatie werkt dus nog steeds met het oude systeem, waardoor er vraag ontstond naar kleinere applicaties, deelproducten met de ontbrekende functionaliteit die aan het oude systeem toegevoegd had moeten worden.'

VERVANGING De beginfase kwam neer op het schiften van bestaande applicaties en het nemen van besluiten voor wat betreft de vervanging ervan. Voor een aantal applicaties is toen vastgelegd dat vervanging ervan echt noodzakelijk was. Die applicaties ontbraken of waren niet goed genoeg. Op korte termijn ontstond aan het begin ook de noodzaak om een specifiek registratiesysteem te bouwen, omdat daar op dat moment ineens dringend behoefte aan bestond.

Stolle: 'Ik zie dat als een andere fase, want in korte tijd moest er iets afkomen. Toen de fusie een feit was, zijn we echt die ontwikkelstraat goed gaan inrichten en hebben we gekeken: wat moet er nog meer gebeuren en hoe kunnen we daar nu een goede architectuur voor neerzetten om ervoor te zorgen dat al die applicaties ook daadwerkelijk met elkaar kunnen communiceren volgens de moderne Microsoft-principes.'

Alle applicaties zijn .NET-applicaties. Met uitzondering van één Windows-(tablet-PC) applicatie voor woninginspecties zijn het ook allemaal webapplicaties. Men is bezig om met behulp van webservices ervoor te

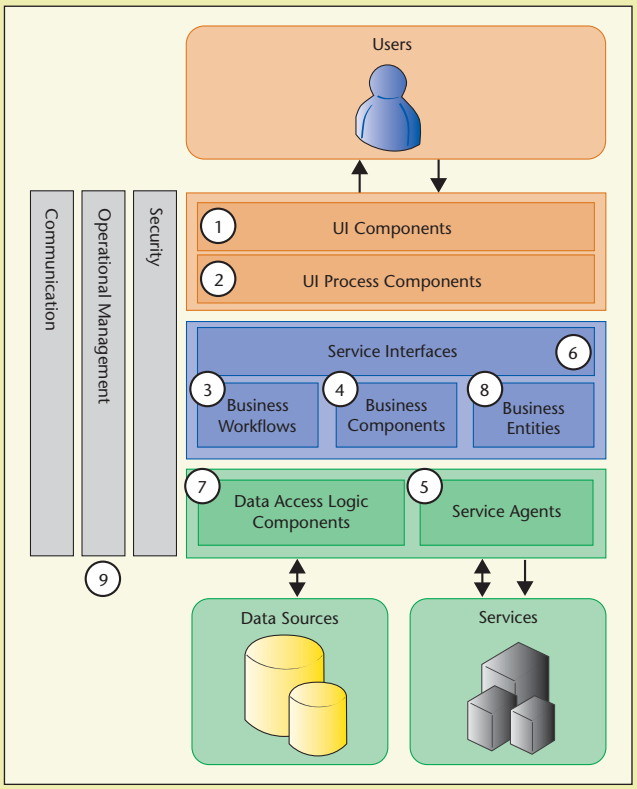
Architectuur en .NET technologie

Dit kader beschrijft de gehele structuur van het implementatiemodel, de decompositie van de software in lagen en subsystemen in het implementatiemodel en de architectuur-significante componenten op basis van Microsoft .NET. Hoewel er meer componenten onderscheiden kunnen worden, zijn de in het schema genoemde onderdelen de meest voorkomende in een gedistribueerde (.NET) applicatie.

Het hierna beschreven model vormt het technische begrippenkader van de oplossing. Bij het technisch ontwerp zullen de termen uit het model zoveel mogelijk worden overgenomen. Deze termen zijn niet leverancierspecifiek, maar wel gangbaar in de Microsoft-architectuurtechnologie.

Een korte beschrijving van elk van de genummerde componenten uit afbeelding 1:

1. User Interface (UI) componenten presenteren de applicatie aan de gebruikers. Mogelijkheden hiervoor zijn een webinterface of een windowsinterface. Overigens kan met deze gelaagde opdeling ook een Windows-client gemaakt worden naast een webapplicatie, waarbij alle overige lagen en componenten onaangepast hergebruikt kunnen worden. De gebruikte techniek is ASP.NET voor web en .NET WinForms voor Windows-applicaties.
2. UI Process componenten bevatten de samenhang tussen de schermen op basis van de geautomatiseerde bedrijfsprocessen. Denk hierbij aan de volgorde van schermen bij een 'Wizard' of de afhandeling van een bestelling. Deze laag bevat nadrukkelijk geen bedrijfsproceslogica. De gebruikte techniek is het .NET Framework.
3. Business Workflows bevatten de sturing van de logische bedrijfsprocessen die van kracht zijn in bepaalde scenario's. Na het verzamelen van de gegevens uit een UI Process, kan een workflow van start gaan. Denk hierbij bijvoorbeeld aan het toebedelen van taken op basis van de beoordeling van binnengekomen materiaal. De gebruikte techniek is het .NET Framework, of bij zwaardere business processen BizTalk Server.
4. Een Business-component behandelt een (of een set van gerelateerde) business rules. Denk hierbij aan het uitrekenen van een premie voor een verzekering, of de uitvoering van een complexe validatie op een gegevensset. De gebruikte techniek is het Microsoft .NET framework.
5. In sommige gevallen wordt een deel van de benodigde logica geleverd door een externe component. Denk bijvoorbeeld aan de creditcard validatie bij Interpay. Een ander voorbeeld is de integratie van een ander intern systeem. Een Service Agent bevat de benodigde communicatieprotocollen om met een dergelijke externe bron samen te werken. De gebruikte techniek is liefst een webservice, maar kan ook een applicatie-eigen protocol zijn.
6. Een Service Interface biedt een uniforme laag waarmee de aanwezige bedrijfsprocessen ontsloten worden. Hierdoor kunnen behalve de User Interface componenten, ook andere, eventueel externe (Service Agents), componenten gebruik maken van de businessprocessen. Daardoor wordt afgedwongen dat de toegang altijd op dezelfde manier plaatsvindt. Tevens wordt oneigenlijk of incorrect gebruik van de processen voorkomen. De gebruikte techniek is het Microsoft .NET framework, maar deze laag wordt feitelijk automatisch gevormd door een goed ontwerp. Als extra mogelijkheid kan ervoor gekozen worden om deze laag ook als een webservice aan te bieden, waardoor de integratie van de betreffende applicatie in andere omgevingen gefaciliteerd wordt.
7. In Data Access Logic componenten wordt vastgelegd HOE een databron (Data Source) benaderd moet worden. Het centraliseren van deze logica bevordert de uniformiteit en faciliteert heterogene databronnen.



AFBEELDING 1. Architectuur en .NET technologie

In ons geval zal de Data Source in elk geval bestaan uit een Oracle-database, maar wellicht zijn er andere databronnen van toepassing, bijvoorbeeld een configuratiebestand. De gebruikte techniek is ADO.NET in combinatie met een specifieke leveranciersdriver, zoals die van Oracle.

8. Om gegevens te tonen, manipuleren of anderszins te gebruiken, is een representatie nodig van de in een Data Source vastgelegde gegevens. Dit gebeurt door middel van Business Entities. De gebruikte technieken zijn DataSets, XML-representaties en custom classes.
9. Een applicatie kan in elke laag gebruik maken van algemene 'services'. Deze services worden geïmplementeerd in componenten. Denk hierbij aan faciliteiten voor Exception Handling, Autorisatie en Configuratie. De gebruikte techniek is het Microsoft .NET framework.

ARCHITECTUUR VAN DE OPLOSSING Bij het opzetten van de architectuur zijn de volgende belangrijkste uitgangspunten gebruikt:

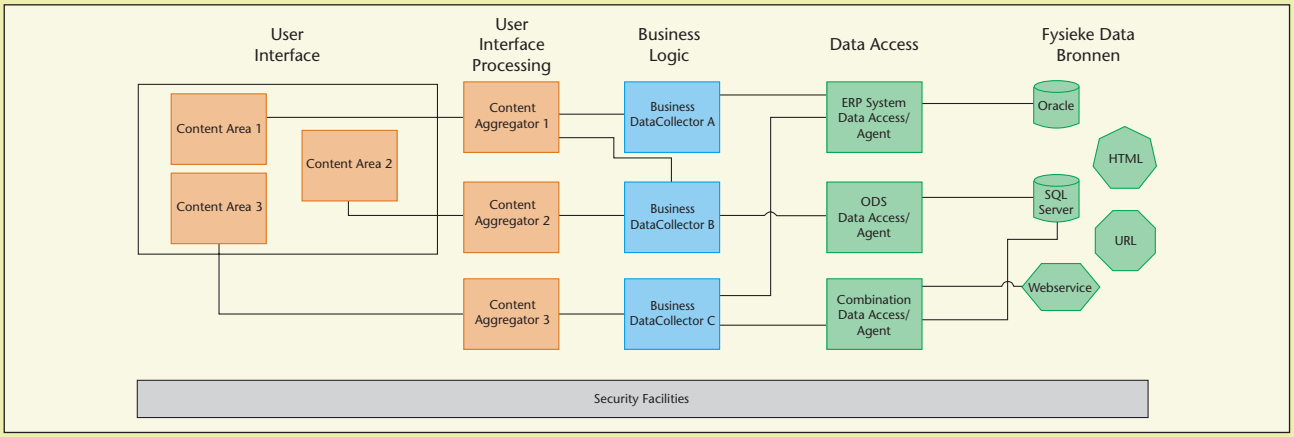
1. De architectuur faciliteert dat aan de 'achterkant' systemen vervangen en/of samengevoegd worden.
2. De architectuur heeft een 'loosely coupled' relatie tussen dat wat gepresenteerd wordt en de bron van de gegevens.

Deze uitgangspunten hebben tot doel om de flexibiliteit en onderhoudbaarheid van het portal te borgen. Hierdoor kan van (toekomstige) wijzigingen ingeschat worden op welke plaatsen ze gevolgen hebben voor de techniek (impact analyse). Daarnaast bestaat de situatie dat de achterliggende gegevens van systeem gaan wisselen.

ARCHITECTUUR VAN HET PORTAL De grafische interface van het portal bestaat uit verschillende functionele blokken, ofwel 'Content Areas', die samen het scherm vormen. Een voorbeeld van een Content Area is het invoergedeelte van een zoekscherm. Een tabblad kan uit meerdere Content Areas bestaan. De Content Areas vormen de UI Components. Elke Content Area kent een onzichtbare, gerelateerde logica component, de 'Content Aggregator'. De functie van de Content Aggregator is om uit één of meerdere logische bronnen gegevens te verzamelen en aan het bijbehorende Content Area aan te bieden. De Content Aggregators vormen de UI Processing Components.

Aan de 'achterkant' van het portal bestaan de diverse systemen die op verschillende manieren benaderd kunnen worden. Voor elk systeem bestaat een specifiek component dat precies weet welke gegevens op welke manier uit het systeem opgehaald kunnen worden. Dit zijn de 'Data Access Components/Agents'.

De koppeling tussen de systeemonafhankelijke Content Aggregators en de systeemspecifieke Data Access Components/Agents vindt plaats in de Business Data Collectors. Deze zijn in staat om de technische vertaalslag te maken tussen wat een Content Aggregator nodig heeft en wat de Data Access Components/Agents leveren. Het kan zo zijn dat meerdere Business Data Collectors gebruik maken van dezelfde Data Access Components/Agents. De fysieke data access hoeft hierbij slechts eenmaal gebouwd te worden. Een Business Data Collector levert de functionele groepering van gegevens. Op hun beurt kan dezelfde Business Data Collector gebruikt worden bij verschillende Content Aggregators. Afbeelding 2 maakt de architectuur inzichtelijk.



AFBEELDING 2. Architectuur van het portal

zorgen dat de verschillende applicaties met elkaar kunnen communiceren. Een deel van de functionaliteit (bijvoorbeeld autorisatie en een adressysteem) wordt via webservices ter beschikking gesteld aan alle applicaties.

Brongers: 'Het zijn allemaal intranet-applicaties, je gaat naar een URL en het systeem weet dus wie je bent omdat je op Windows bent ingelogd. De rechten die je hebt zijn echter niet vastgelegd in de active directory. Daar is een aparte database voor gemaakt omdat ze dusdanig specifiek zijn dat je ze lastig in active directory kwijt kan. Dat heeft te maken met het feit dat je in active directory bijvoorbeeld groepen, als je die gedefinieerd hebt, niet op twee plekken kunt gebruiken. Als je een groep applicaties hebt waar je een groep gebruikers aan wil toekennen én een groep gebruikers waar je een groep applicaties of functies van applicaties aan wil toekennen kom je in de knoop. Dan ontkom je er niet aan één van de twee dubbel uit te voeren. Bovendien is er voor veel applicaties data-segregatie van toepassing, waarvoor de rollen in de database nodig zijn. Voor het beheer het toekennen van de rollen is een webvoorkantje, dat werkt wat prettiger dan de standaard active directory tools. Autorisatie is in de eigen database geregeld, de authenticatie gebeurt gewoon met Windows.

DOMEINEN Brongers: 'Binnen een applicatie gaat het altijd over een bepaald onderwerp of domein, bijvoorbeeld het registreren van moeilijke huurders. Alle informatie binnen dat domein is in feite eigendom van die applicatie. Daar hoort dus een eigen database bij. Maar het kan best zo zijn dat je daar gebruik wil maken van adressen. Eigenlijk maak je altijd gebruik van de gegevens uit de centrale database om aan te geven om welk huurobject het gaat. Die koppelingen worden niet per applicatie gebouwd, dat gebeurt met webservices. Zo kun je elke applicatie zien als een domein, eigenaar van bepaalde gegevens en gebruiker van andere gegevens waarvoor hij andere domeinen aanspreekt. Het overkoepelende stuk is het feit dat elke applicatie op dezelfde manier is opgebouwd, dezelfde onderdelen kent, dezelfde *look & feel*. Omdat dat zo is, zijn er inmiddels ook GUI-elementen gemaakt die herbruikbaar zijn, bijvoorbeeld een gridlijstje waar standaard veel voorkomende sorteermogelijkheden op mogelijk zijn. Daarbij wordt altijd gekeken of je het koopt of zelf bouwt. Voor de mail-merge bijvoorbeeld is een component gekocht, voor het grid is zelf iets gebouwd omdat er specifieke eisen waren. Dat is mogelijk omdat je al die dingen op dezelfde manier insteekt. De huisstijl is daarbij ook heel belangrijk.'

UI Stolle: 'Alle applicaties zien er hetzelfde uit. Ik zorg er altijd voor dat er eenzelfde manier van toegankelijkheid is, dat mensen weten waar ze terecht komen wanneer ze een applicatie opstarten. Het zoeken is altijd de ingang, daarna volgt een detailscherm waarin verschillende *user-controls* zitten om allerlei informatie te ontsluiten. Deze opbouw maakt hergebruik heel gemakkelijk, bijvoorbeeld met een component voor het datagrid, zo kun je heel snel en steeds sneller een nieuwe applicatie bouwen. Ook wanneer we de applicatie uitbreiden met nieuwe functionaliteit of bestaande functionaliteit door voortschrijdend inzicht veranderen, zal de applicatie die daarop volgt altijd terugvallen op de principes van daarvoor. Twee applicaties terug hebben we bijvoorbeeld een soort context-sensitieve helpfunctionaliteit ingebouwd. Dat kunnen we dan vrijwel meteen implementeren, daar hoeven we niets meer voor te doen. Het enige wat we hoeven te zeggen tegen de opdrachtgever is: hier is de XML-file, dit zijn alle verschillende schermen en varianten die kunnen voorkomen, zet je tekstjes er maar bij en dat hangen we erin.'

Ir. Reinhard P. Brongers, architect van het project

VAN SCRATCH *Als er van scratch gebouwd zou zijn, zou het dan heel anders gegaan zijn?*

Brongers: 'Wanneer we ook de kernapplicatie zelf gebouwd zouden hebben, zou die waarschijnlijk al die modules bevatten voor de verschillende domeinen, omdat je dan vanuit één centraal systeem zou werken. Ik denk echter dat wanneer je het helemaal opnieuw zou moeten doen de overall architectuur gelijk zou zijn en dat je het alleen op details anders zou doen. Als je het nu

Robin C.J. Stolle, informatieanalist en teamleider van het project

zou gaan bouwen dan zou je wél features van .NET 2.0 gebruiken in plaats van die dingen die nu zelf zijn ingevuld of deels ontbreken. Het is echter niet noodzakelijk om op korte termijn over te gaan op ASP.NET 2.0 , daarvoor is er te weinig directe toegevoegde waarde.

Stolle: 'We zijn nu bezig met een applicatie die wat workflow-achtige kenmerken heeft en de bouw wordt nu wel in 1.1 gedaan en we gaan ook de huidige faciliteiten gebruiken. Maar we doen tevens een pilot in schaduw met Workflow Foundation om te kijken: heb je daar nu echt voordeel van, wat is dan dat voordeel en hoe snel kun je met die Windows workflow-faciliteiten werken. We zijn er dus wel mee bezig maar de noodzaak is er nog niet om te zeggen: we gaan nu over. Maar de klant realiseert zich ook wel dat ze mee moeten met de tijd dat er in toekomstige applicaties dingen zullen komen die met het nieuwste framework veel handiger opgelost zouden kunnen worden.'

Veel legacy is uiteindelijk verdwenen. Waar dat niet kon, is er gekozen voor een oplossing van één van de twee oorspronkelijke bedrijven, maar die applicaties waren onder te brengen in SQL-server, en waren dus relatief gemakkelijk te benaderen.

ONTWIKKELSTRAAT *De oorspronkelijke vraag naar de invoering van een ontwikkelstraat houdt in de praktijk meer in dan opdrachtgevers zich voorstellen.*

Brongers: 'Een ontwikkelstraat is niet een doos met een strik. Je kunt wel bepaalde principes overnemen, in dit geval vertaalt zich dat naar met name de programmeerstandaarden en het proces waaronder je werkt. Het uitgangspunt is in feite de architectuur zoals die voor die situatie bedacht is en de *assets* die in de ontwikkelstraat zitten, de design guidelines en dat soort zaken, alsmede een aantal technische componenten. Je ziet ook een belangrijk verschil als je kijkt naar een organisatie als VX-company, wij doen steeds verschillende projecten voor andere klanten, we hebben steeds te maken met een ander business-domein. Binnen een woningcorporatie heb je juist altijd te maken met jouw business domein, je kunt dus ook jouw domeinspecifieke dingen in de ontwikkelstraat stoppen. Wat je in de VX- ontwikkelstraat moeilijk in kunt bakken, is een standaardmanier om rechten en rollen toe te kennen. Het is wel mogelijk, maar de kans dat je dat iedere keer kunt gebruiken is vrij klein, dat hangt maar net van de situatie af. Bij een woningcorporatie is de manier waarop rollen en rechten toegekend worden onderdeel van de ontwikkelstraat, want dat gebeurt altijd op dezelfde manier. Dit soort aspecten worden *tailor made* geplaatst binnen de hoofdontwikkelstraat. Bij het invoeren van een ontwikkelstraat hoort overigens ook training over zaken als codeerstandaarden, databaseontwikkeling of reviews. De training was maatwerk: we keken naar de achtergrond van individuen om te bepalen welke kennis nodig is, maar ook om mensen wegwijs te maken in de ontwikkelstraat. Hoe werken we nu eigenlijk, hoe verloopt een project, welke fases zijn er, welke deliverables, portals, codestandaarden, architectuur, los van algemene skills een opleiding in hun eigen ontwikkelstraat.'

SPARRING PARTNER Tijdens het project is iteratief gewerkt, zij het niet met een vastgelegde methodologie of werkwijze. Er werd al in een vroeg stadium met prototyping gewerkt waarbij gebruikers al snel met (detail-)wijzigingen kwamen. Bij aanpassingen werd vrij flexibel gekeken óf en, zo ja, in welke versie van de software die nog konden worden doorgevoerd. Terugkijkend is met name Stolle daar dan ook tevreden over:

'Wat ik hier heel erg leuk vind, is dat je een heel nauwe samenwerking hebt met de projectleider van de opdrachtgever. Deze functioneerde ook als sparring partner, we zaten na gebruikerstesten vaak samen om de tafel om te zien hoe we dingen zouden kunnen veranderen. Zo hebben we veel, weliswaar kleine wijzigingen doorgevoerd waardoor de applicatie intuïtiever geworden is en sneller opgepikt wordt door de gebruikers.'

Tekst en fotografie: Dré de Man