# Drie architectuurpatronen voor applicatie-integratie

van Quick Win naar Flexibiliteit

*De roep om snelheid blijft onverminderd van kracht in applicatie-integratieprojecten. Dit geldt met name waar e-business in het geding is. Maar deze snelheid mag niet ten koste gaan van de kwaliteit van zowel het proces als het eindresultaat. Een belangrijk hulpmiddel om aan deze eisen te voldoen is een architectuurpatroon. Het begrip patroon is relatief nieuw in de wereld van IT, maar werd al in 1977 door Christopher Alexander geïntroduceerd in de context van gebouwen en steden. Na de adoptie van patronen binnen de IT door Erich Gamma in het kader van objectoriëntatie, beleven patronen de laatste tijd een hernieuwde belangstelling. En niet voor niets. Patronen werken versnellend, aangezien zij een vastlegging van opgedane ervaringen zijn, en kwaliteitsverbeterend omdat er wordt voortgebouwd op bewezen oplossingen. In dit artikel worden drie architectuurpatronen behandeld voor applicatie-integratie die zijn gebaseerd op de ervaringen van de auteurs. De drie patronen vormen tezamen een groeipad voor integratie.*

Ondanks de tegenvallende resultaten van de eerste golf van e-business is het Internet niet meer weg te denken uit de moderne manier van zaken doen. Deze eerste golf, *business-to-consumer*, bestond uit het koppelen van consumenten met de eigen organisatie. Dit stelde verzekeringsmaatschappijen bijvoorbeeld in staat om verzekeringen via het Internet te verkopen. De tweede golf, *business-to-business*, die nu volop in opkomst is, richt zich op het koppelen van in- en externe bedrijfsprocessen. In de praktijk stelt dit bedrijven bijvoorbeeld in staat om handmatige koppelingen tussen het verkoop- en leveringsproces te automatiseren of het eigen inkoopproces te koppelen met het verkoopproces van leveranciers.

De bedrijfseconomische drijfveren van de twee golven zijn compleet verschillend. Waar in de eerste golf een relatief goedkoop marktkanaal extra omzet diende te genereren, richt de tweede zich op het beheersen van de proceskosten. Immers, een geautomatiseerd proces maakt geen fouten, is nooit ziek en haalt ongelofelijke snelheden. Een constante factor in beide golven is de noodzaak om bedrijfsprocessen waar nodig te automatiseren, en om deze geautomatiseerde processen met elkaar te koppelen. Dit laatste is het gebied van applicatie-integratie.

## *De rol van architectuur*

Om als organisatie op eenduidige wijze inhoud te geven aan applicatie integratie is het noodzakelijk om op hoofdlijnen een flexibele architectuur op te zetten. In deze Enterprise architectuur, door de Gartner Group ‘cityplanning’ genoemd, wordt ten minste ingegaan op drie aspecten: de plaats van de geautomatiseerde bedrijfsprocessen binnen de organisatie, de onderlinge koppeling tussen deze processen en de koppeling met geautomatiseerde bedrijfsprocessen bij externe partijen. Het hoogste goed dat met een Enterprise architectuur wordt nagestreefd is een éénmalige implementatie van geautomatiseerde bedrijfsprocessen, die vervolgens vanuit andere bedrijfsprocessen aangeroepen kan worden [2].

Ter illustratie kan worden gekeken naar een bankverzekeraar die als intern geautomatiseerd bedrijfsproces een applicatie heeft die kredietaanvragen afhandelt. Deze applicatie zal een externe koppeling nodig hebben met het Bureau Krediet Registratie (BKR) om de kredietwaardigheid van de aanvragers te controleren. De wijze waarop deze systemen met elkaar communiceren kan op vele wijzen worden gerealiseerd. Door hier een keuze te maken die niet op zichzelf staat, maar in lijn is met de door de Enterprise Architectuur neergelegde communicatievoorschriften, wordt hergebruik van kennis en middelen bevorderd. Dit maakt de gekozen oplossing eenvoudig te begrijpen, te realiseren en te onderhouden. Daarnaast kan de bedrijfsvoering zoals deze door de leiding van het bedrijf is vastgesteld, eisen aan de koppeling stellen. Door deze uitgangspunten in de voorschriften te verankeren, wordt voorkomen dat wordt gekozen voor de technisch meest ideale oplossing, terwijl deze de bedrijfsvoering van de organisatie ondermijnt.

Cap Gemini Ernst & Young heeft een eigen architectuurmethodiek, het Integrated Architecture Framework (IAF). Het centrale punt in deze aanpak is dat ieder IT element in een organisatie direct te relateren moet zijn aan een uitgangspunt van de business. Hiertoe wordt ieder architectuurtraject in een viertal vaste fasen doorlopen. De eerste fase is de contextuele, waarin de scope van het proces en de

daaraan gerelateerde zakelijke domeinen in kaart worden gebracht. De tweede fase, de conceptuele, schetst op grond van de zakelijke uitgangspunten die binnen de relevante domeinen gelden de gewenste oplossing op hoofdlijnen. Gespreksgenoten bij de klant in deze eerste twee fasen zijn de zakelijke managers, die de opgeleverde architectuur toetsen aan de bedrijfsdoelstellingen. De volgende twee fasen zijn meer op automatisering gericht. De derde fase is de logische, waarin de conceptuele architectuur wordt vertaald naar services, zoals klantbeheer, polisadministratie, databases, authenticatie of workflow. De laatste fase is de fysieke. In deze fase wordt de logische architectuur vertaald naar producten, zowel in termen van hardware als software. Gespreksgenoten bij de klant in deze laatste twee fasen zijn IT managers.

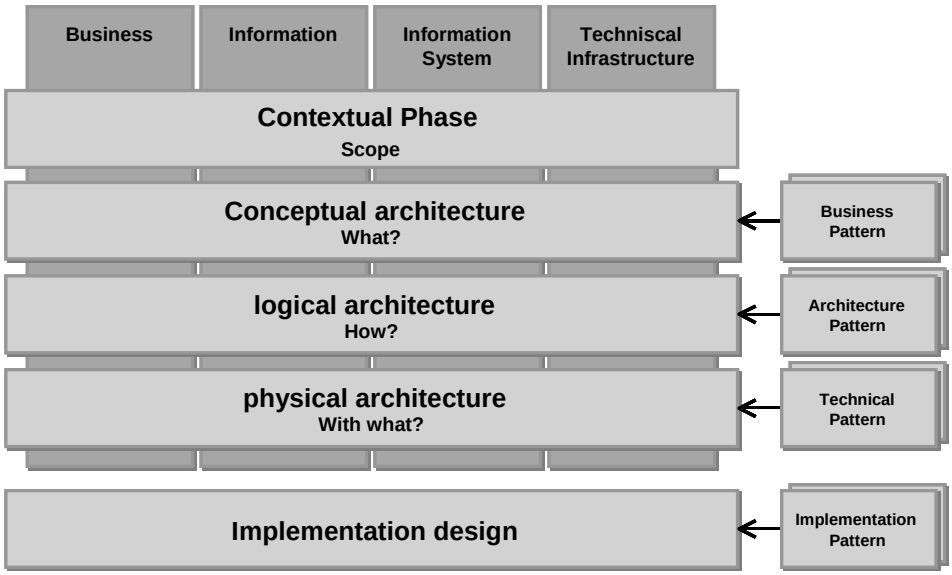
Deze generieke aanpak is toepasbaar op verschillende architectonische deelgebieden. Binnen het IAF wordt standaard al voorzien in een methodiek voor vier deelgebieden. Dit zijn de business, informatie, informatiesystemen en technische infrastructuur. Business richt zich op de bedrijfsvoering, terwijl informatie de informatiestromen binnen een organisatie in kaart brengt. Informatiesystemen is gericht op het ontwerp van veelal gedistribueerde informatiesystemen. Vaak wordt bij de invulling van deze systemen gebruik gemaakt van verscheidene applicaties die via een veelvoud aan IT mechanismen met elkaar gekoppeld zijn. De fysieke stroom, tenslotte, is puur gericht op het bepalen van de juiste hardware. De vier stromen moeten complementair gezien worden. In een groot, internationaal project zullen zij naast elkaar lopen waarbij de output van de ene stroom de input voor de andere vormt. Bij kleinschaligere projecten zal methodiek en gedachtegoed uit meerdere stromen in één aanpak worden verenigd.

## *De rol van architectuurpatronen*

Het begrip patroon is relatief nieuw in de wereld van IT, maar werd reeds in 1977 door Christopher Alexander geïntroduceerd in de context van gebouwen en steden [3]: *A pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use the solution a million times, without ever doing it the same twice*. Het grote voordeel van het gebruik van patronen betreft enerzijds het versnellende, doordat het wiel niet opnieuw uitgevonden hoeft te worden, en anderzijds het kwaliteitsverbeterende karakter, er wordt immers uitgegaan van een bewezen oplossing.

In de wereld van IT werd het begrip patroon rond 1994 voor het eerst geadopteerd door Erich Gamma [4] in het kader van object oriëntatie. De laatste tijd beleven patronen een hernieuwde belangstelling in het kader van IT-architectuur en IT-infrastructuur.

Het vastleggen en hergebruiken van architectuurpatronen in aanvulling op architectuur is één van de belangrijke pijlers om sneller projecten op te leveren waarbij een hoge kwaliteit gegarandeerd blijft. Patronen zijn immers een weerslag van '*best practices*', de documentatie van succesvolle projecten. Hiertoe is IAF aangevuld met een drietal patroontypes: business patronen, architectuurpatronen en technische patronen. In het onderstaande schema is te zien hoe deze patronen aansluiten op de fasen van het IAF.



Deze architectuurpatronen sluiten aan op de patronen voor object oriëntatie van Erich Gamma. Deze patronen geven een mogelijke oplossingsrichting in objecten voor de gedefinieerde services.

## *Drie architectuurpatronen voor applicatieintegratie*

Binnen de scope van een Enterprise Architectuur is een aantal architectuurpatronen te definiëren. Voor de definitie van de patronen zijn twee uitgangspunten genomen die een vast onderdeel dienen te vormen van ieder serieus architectuurproject.

Het eerste uitgangspunt is ***e-business is integratie***. Het meest elementaire kenmerk van ieder e-business traject is dat een geautomatiseerde koppeling wordt ontwikkeld tussen een externe partij en een intern bedrijfsproces. Of deze partij een persoon is die via het Internet een aankoop doet, of een applicatie bij een andere organisatie, is irrelevant. Van belang is dat de relevante processen volledig geautomatiseerd worden afgewerkt, en dat vereist integratie van de onderhavige systemen.

Het tweede uitgangspunt is ***integratie is ontkoppeling***. Dit lijkt een contradictio in terminis, maar is in feite een cruciale basisvoorwaarde die in ieder succesvol concept in de wereld is terug te vinden. Een auto, bijvoorbeeld, is niets anders dan een serie relatief los gekoppelde componenten. Dit stelt ons in staat ieder onderdeel met een eenvoudige serie handelingen uit de auto te verwijderen om te repareren of te vervangen door een fraaier exemplaar. Denk eens aan de problemen die zouden ontstaan als iedere auto één grote aan elkaar gelaste klomp onderdelen was. Toch moet bij applicatie-integratieprojecten de noodzaak tot loskoppeling steeds opnieuw worden aangetoond, en lijkt het zich blindstaren op de alledaagse werkdruk, samen met een gebrek aan visie over de verantwoordelijkheden van deelprojecten, steeds weer tot lasklomp-oplossingen in de automatisering te leiden.

Met deze uitgangspunten is een drietal architectuurpatronen te onderkennen waarin de mate van ontkoppeling steeds toeneemt. De patronen zijn opgesteld op basis van de praktijkervaringen van de auteurs met applicatie-integratieprojecten. Het eerste patroon, ***Quick Win***, biedt nog geen ontkoppeling, maar stelt een bedrijf slechts in staat om snel haar interne processen extern aan te bieden. Het tweede patroon, ***Backoffice Integratie***, voorziet al meer in ontkoppeling. Daarnaast biedt het ook echte integratie met de bestaande interne processen in de backoffice. Het laatste patroon, ***Flexibiliteit***, borduurt voort op de voordelen van het tweede patroon, maar voegt hier de benodigde flexibiliteit aan toe die een organisatie in staat stelt sneller interne functionaliteit extern aan te bieden.

## *Patroon 1: Quick Win*

Probleem

Een organisatie met een traditionele backoffice omgeving is niet zondermeer in staat haar bestaande interne bedrijfsprocessen aan te bieden aan externe partijen. De huidige applicaties zijn niet gebouwd op interactie met externe spelers, en de benodigde aanpassingen om dit te realiseren zijn kostbaar en tijdrovend.

Oplossing

Een eenvoudige manier om dit op te lossen, is door niet het hele proces geautomatiseerd aan de buitenwereld aan te bieden, maar alleen het punt van interactie: de frontoffice. Als de externe partij een privé persoon is, betekent dit dat alleen het verschaffen van informatie en het bestellen van producten geautomatiseerd wordt, maar niet noodzakelijkerwijs de afhandeling van die bestellingen.

Implementatie Richtlijnen

Op een applicatieserver wordt generieke functionaliteit gebouwd die de interactie met de externe partij kan afhandelen. Deze functionaliteit kan vervolgens via diverse technologische kanalen aan externe partijen worden aangeboden. Dit kan via een web- of wapserver op een privé persoon worden gericht, of via een webservice op een andere organisatie. Waar noodzakelijk worden de bestaande backoffice systemen één op één gekoppeld met de frontoffice, om een volledig geautomatiseerde afhandeling van het proces te realiseren. Hiervoor wordt standaardtechnologie gebruikt zoals deze op een applicatieserver voorhanden is. Hierbij kan gedacht worden aan de mogelijkheid om de backoffice functionaliteit op de applicatieserver beschikbaar te stellen als een COM+ component of een Enterprise Java Bean (EJB). Dit component staat via een synchrone koppeling in contact met het feitelijke component op het mainframe. Als het mogelijk is de applicatie via een bestaande interface te ontsluiten, heeft dat de voorkeur. Anders kan van technieken als screenscraping gebruik worden

gemaakt Een voorbeeld hiervan is Verastream Host Integrator van WRQ waarmee middels het concept "screens as components" mainframeschermen beschikbaar gemaakt kunnen worden als componenten.

Praktijkvoorbeelden
De eerste versie van veel beleggingssites maakte gebruik van dit patroon door klanten aan- en verkoopformulieren voor aandelen via het Internet beschikbaar te stellen. Op het moment dat een klant deze formulieren ingevuld heeft, worden ze bij het beleggingsinstituut via de al bestaande procedures verder afgehandeld. Dit kan volgens de hierboven beschreven methode volledig geautomatiseerd gebeuren, maar vaak betekent dit dat de formulieren handmatig opnieuw worden ingevoerd in de bestaande backoffice systemen. In dit voorbeeld vindt er zelfs geen geautomatiseerde, maar een handmatige integratie met de backoffice plaats. Op deze wijze was het mogelijk om in zeer korte tijd beleggen via het Internet aan te bieden zonder de bestaande bedrijfsvoering aan te passen. Dat deze oplossing zeer tijdelijk van aard dient te zijn, mag duidelijk zijn.

Consequenties
Door het gebruik van de binnen de applicatieserver voorhanden zijnde synchrone technologie voor de koppeling met de backoffice is het niet mogelijk om alle front- en backoffice functies te integreren. Immers, de frontoffice vraagt een rond de klok beschikbaarheid van functionaliteit, terwijl de meeste backoffice functies 's nachts niet beschikbaar zijn doordat batchprocessen en back-ups uitgevoerd worden. Een groot voordeel is echter dat heel snel en relatief eenvoudig een nieuwe hoeveelheid kanalen ontstaat voor bestaande processen.

## *Patroon 2: Backoffice Integratie*

Probleem
Integratie via *Quick Win* regelt weliswaar ontsluiting van de bestaande functionaliteit, maar er is een belangrijk nadeel aan backoffice systemen dat een negatieve uitstraling op de totaaloplossing heeft. Inherent aan klassieke mainframe technologie is het beperkte bioritme. Op gezette tijden moet het systeem worden stopgezet om batchverwerking te draaien en back-ups te maken. Kenmerkend aan de moderne tijd is juist beschikbaarheid van bedrijfsprocessen rond de klok.

Oplossing
De beste oplossing voor dit probleem is om alle bestaande mainframe systemen te ontmantelen, en te vervangen door moderne gedistribueerde applicaties. In de praktijk is dit uiteraard op korte termijn niet haalbaar. In plaats daarvan kan wel een mechanisme worden opgenomen tussen de mainframe applicaties en de moderne applicaties dat het verschil in bioritme oplost. De technologie daarvoor is *message oriented middleware* (MOM), ofwel integratie middels asynchrone communicatie. Applicaties spreken niet direct met elkaar, maar plaatsen hun boodschappen in een postbakje dat door de ontvangende partij kan worden uitgelezen op het moment dat deze beschikbaar is.

Implementatie Richtlijnen
Op een applicatieserver wordt, evenals in het architectuurpatroon *Quick Win* generieke functionaliteit gebouwd die de interactie met de externe partij kan afhandelen via diverse technologische kanalen. Waar noodzakelijk worden de bestaande backoffice systemen één op één gekoppeld met de frontoffice met asynchrone middleware. Hierbij kan gedacht worden producten zoals MSMQ van Microsoft, MessageQ van BEA Systems of MQSeries van marktleider IBM in dit segment.

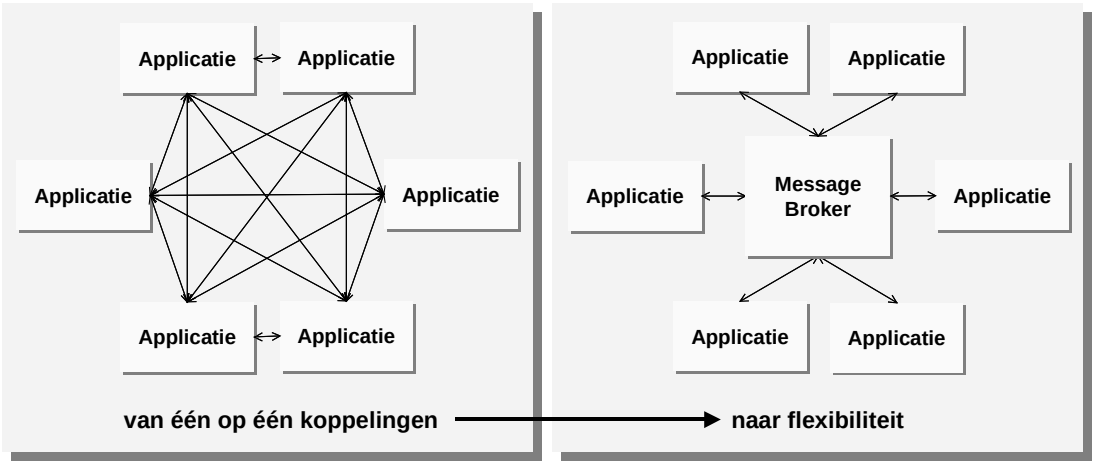
Praktijkvoorbeelden
In de strijd tussen de traditionele verzekeraars onderling en de aanstormende concurrentie van diverse start-ups zoals Independer en Well-O-Well zien veel verzekeraars zich gedwongen om zich te transformeren van hun traditionele vorm, *brick-and-mortar*, naar een gemengde vorm, *brick-and-click*, waarin ook ruimte voor nieuwe media is. Veel verzekeraars zijn al op het Internet te vinden voor statische activiteiten als het opvragen van informatie. Als gevolg van de transformatie naar *brick-and-click* worden zaken als het verkopen van polissen en het indienen en afhandelen van claims steeds vaker elektronisch afgehandeld. Hiervoor moet de frontoffice, waar de polisaanvraag of claimaanvraag plaatsvindt, worden gekoppeld met de backoffice, waar de polisacceptatie en de claimafhandeling plaatsvindt. Doordat deze backoffice systemen veelal niet rond de klok beschikbaar zijn, is de inzet van asynchrone middleware noodzakelijk om het verschil in bioritme te overbruggen. Dat wil zeggen dat polisaanvragen die binnen de beschikbare uren van de backoffice ingediend zijn, direct afgehandeld worden, en dat de polisaanvragen die buiten de beschikbare uren ingediend zijn tijdelijk bewaard worden in het elektronische postbakje.

Consequenties
Door het gebruik van een asynchrone koppeling tussen de diverse processen is het verschil in bioritme te overbruggen, maar vergt dit een investering in de nieuwe technologie asynchrone middleware. Daarnaast ontstaat bij veelvuldig gebruik van één op één koppelingsmechanismen al gauw een kluwen van koppelingen tussen de diverse systemen die moeilijk te beheren is. Immers, bij een hoeveelheid applicaties, gerepresenteerd door de variabele "n", leert ons de eenvoudige formule "aantal koppelingen = n*(n-1)" dat de potentiële hoeveelheid koppelingen al snel schrikbarende hoeveelheden aanneemt.

## *Patroon 3: Flexibiliteit*

Probleem
Zoals al eerder gezegd, leidt het één op één koppelen van applicaties tot een onbeheerbare kluwen. Deze kluwen ontstaat door de noodzaak om in een applicatie allerlei specifieke kennis op te nemen van de gekoppelde applicaties. Applicatieontwikkelaars moeten uitzoeken hoe andere applicaties technologisch werken, en via welke transportmechanismen en boodschapformaten zij communiceren. Hoe meer er geïntegreerd wordt, des te meer van dit soort 'infrastructurele informatie' opgenomen dient te worden. Als dit tijdens ontwikkeling al tot een sterk verhoogde complexiteit leidt, bij het in beheer nemen of aanpassen van applicaties leidt dit tot onwerkbare situaties. Zo ziet een grote verzekeraar in Nederland zich gedwongen te accepteren dat meermalen per jaar haar productiesystemen plat gaan als gevolg van het onbeheersbare karakter van de vele één op één koppelingen dat door de tijd heen is aangelegd.

Oplossing
De oplossing van dit probleem voorziet in een extra ontkoppeling. Alle specifieke informatie die voor het koppelen van applicaties noodzakelijk is wordt centraal opgeslagen in een *message broker*. Over het algemeen zijn dit drie typen informatie: *formatterings-, routerings en workflowinformatie*[1]. Hierdoor ontstaat een zogenaamde hub-and-spoke architectuur, een architectuur waarin alle onderdelen binnen de architectuur gebruik maken van de diensten van een centraal opgestelde service. In het *message broker* scenario worden de spokes gevormd door de applicaties en de hub door de message broker, waardoor het aantal potentiële koppelingen reduceert tot het minimaal benodigde aantal. Naast deze reductie van één op één koppelingen, biedt de message broker het voordeel dat syntactische en semantische verschillen relatief eenvoudig overbrugd kunnen worden. Hierbij wordt transformatieinformatie geïmplementeerd in de message broker zodat binnenkomende berichten vertaald kunnen worden naar het formaat dat de ontvangende applicatie begrijpt.




[1] Routering houdt in dat applicaties niet meer op detailniveau hoeven te weten waar en hoe ze andere functionaliteit kunnen aanspreken. Het volstaat om een bepaalde functionaliteit aan te roepen, waarna de message broker zorgt dat deze aanroep juist wordt afgeleverd. Workflow gaat een stap verder dan routering omdat hierbij niet alleen losse zender en ontvanger informatie, maar een keten van zender en ontvanger informatie opgeslagen wordt op bedrijfsprocesniveau. Het gaat echter niet zover als traditionele workflow, waarbij uitgebreide controle op taakvervulling en het verzamelen van managementinformatie mogelijk is.

Implementatie Richtlijnen

Op een applicatieserver wordt, evenals bij de architectuurpatronen *Quick Win* en *Backoffice Integratie*, generieke functionaliteit gebouwd die de interactie met de externe partij kan afhandelen via diverse technologische kanalen. De bestaande backoffice systemen worden via de message broker gekoppeld met de frontoffice middels asynchrone middleware.

Belangrijke leveranciers van message brokers zijn IBM met de combinatie van MQSeries Integrator en MQSeries Workflow, Neon's E-bizz Integrator, Integration Manager van TIbco, Mercator van het gelijknamige bedrijf, Level 8 met GenevaIntegrator, Seebeyond met E*Gate, Vitria met Businessware en Microsoft's Bizztalk.

Praktijkvoorbeelden

Naast de strijd waarbij veel verzekeraars zich gedwongen zien zich te transformeren van *brick-and-mortar* naar *brick-and-click* organisatie vindt een duidelijk overname- en fusiegolf plaats tussen verzekeraars onderling en met banken. Het gevolg hiervan is een imposante bruidsschat aan systemen.

Neem als voorbeeld een fictieve bankverzekeraar die polisadministraties voert voor leven, schade, zorg en banking. Zo is er een levensysteem aanwezig dat is gerealiseerd in CICS/COBOL op een OS/390 mainframe. Daarnaast is er een tweede levensysteem aanwezig met COBOL/OpenUTM op een Siemens BS2000. Dit tweede levensysteem is door een recente overname van een middelgrote levensverzekeraar bij dit bedrijf terechtgekomen. Het schadesysteem is een standaardpakket dat onlangs gekocht is en draait op een Sun Solaris machine onder Unix. De zorg- en bankingsystemen draaien beide op een Unisys A-Series mainframe met Linc.

Deze bankverzekeraar staat voor de uitdaging om via het internet extra marktomzet te genereren terwijl de verzekeringsproducten ook in de bankfilialen verkocht moeten gaan worden.

Door met het derde architectuurpatroon aan de slag te gaan kunnen de uitdagingen van deze organisatie worden gerealiseerd, ondanks de beperkingen die de bruidschat met zich meeneemt. In het frontoffice worden de benodigde functies voor zowel het internetkanaal als de bankfilialen generiek opgezet. Deze functies worden via asynchrone middleware aan de backoffice aangeboden om de verschillen in bioritme op te lossen. De backofficesystemen worden ook ontsloten met asynchrone middleware, om ook aan deze kant verschillen in bioritme op te lossen[2]. In het midden wordt de *message broker* ingezet om allerlei technologische verschillen tussen alle applicaties op te lossen. Verschillen in boodschap formaat worden getransformeerd en infrastructurele aspecten worden opgelost door de routeringsfunctionaliteit. Complexere vragen uit de frontoffice, tenslotte, worden in een workflow ondergebracht eer ze naar de diverse backoffice systemen worden gestuurd om te worden afgehandeld.

In bovenstaande voorbeeld levert de combinatie van asynchrone middleware en een message broker de bankverzekeraar de flexibiliteit om zijn uitdagingen met een relatief eenvoudige infrastructuur te realiseren.

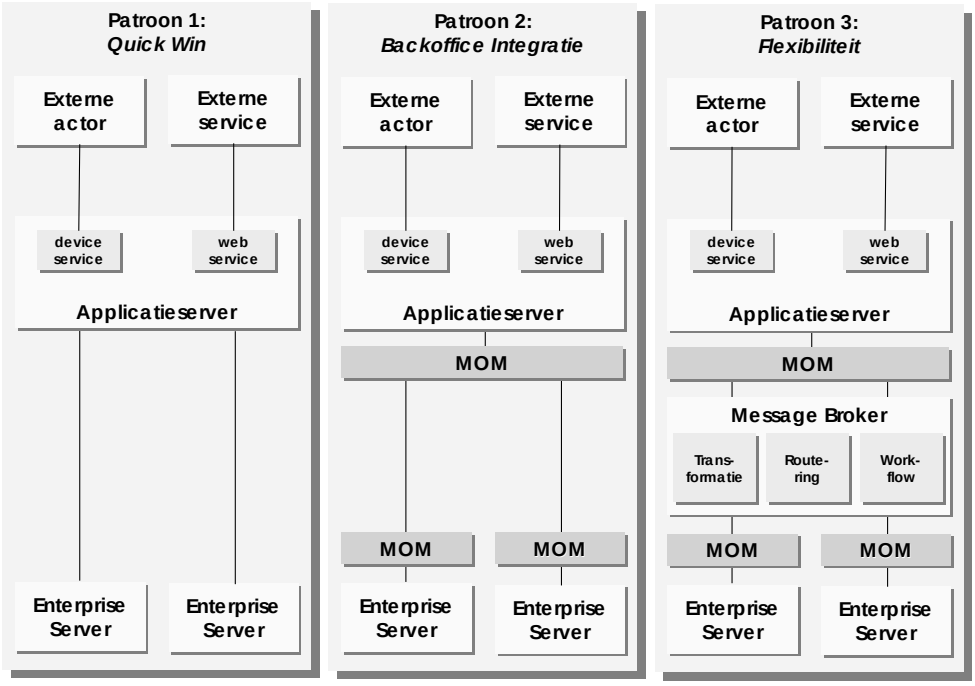
Consequenties

Door het gebruik van een asynchrone koppeling in combinatie met een message broker tussen de diverse processen is het verschil in bioritme te overbruggen en de nodige flexibele te realiseren. Waar het tweede patroon, B*ackoffice Integratie*, echter al investering in de relatief nieuwe technologie voor asynchrone middleware vereist, is voor dit patroon ook investering in de nog nieuwere technologie van de *message broker* noodzakelijk. En waar de markt voor asynchrone middleware al redelijk is uitgekristalliseerd, geldt voor de markt voor *message brokers* dat deze nog volop in beweging is. Er zal de komende tijd een shake-out gaan plaatsvinden, waarbij een duidelijk trend naar consolidatie tussen de vaak nog diverse technologische oplossingen zal plaatsvinden. Het is nu nog moeilijk vast te stellen welke leverancier een grote kanshebber is deze shake-out te overleven.

[2] In werkelijkheid zal naast de ontsluiting naar asynchrone communicatie ook nog andere middleware nodig zijn om platform- of ontwikkelingsomgeving specifieke aspecten op te lossen. In dit patroon wordt deze noodzaak om redenen van overzichtelijkheid achterwege gelaten.

## *Groeipad*

Het onderstaande schema geeft de drie patronen schematisch weer. Wat direct opvalt is het groeipatroon dat vanaf *Quick Win* naar *Flexibiliteit* mogelijk is.




Bij dit groeipad hoort wel het bewustzijn dat in een oplossing gebaseerd op het gebruik van een message broker, het een belangrijke voorwaarde is dat binnen de gekoppelde applicaties een goede componentisering bestaat. Dat wil zeggen dat niet langer kan worden gedacht in termen van *stovepipes of* mainframe-applicaties waar alle code één grote lap tekst is, maar dat een scheiding moet worden aangebracht tussen presentatie, besturing, materielogica, databenadering en data. Alleen dan kan optimaal hergebruik van bestaande functionaliteit worden gemaakt. Maar dit bewustzijn is slechts de eerste stap, er moet nog een volgende worden genomen: hetzelfde ontkoppelingsprincipe dat binnen applicaties geldt, dient ook voor de applicatie-integratie oplossing als geheel te gelden.

Als een organisatie verder gevorderd raakt in applicatie integratie zal op een gegeven moment niet alleen meer sprake zijn van koppeling van een aantal individuele applicaties, maar van vele functieblokken. Een functieblok kan dan deel een applicatie zijn, maar ook een enkel component op een applicatieserver, zoals in het eerste patroon, *Quick Win*, wordt getoond. Uiteindelijk kan dan het bedrijf Sun Microsystems worden geparafraseerd door te stellen "*the network is the Information System*". Als in dat geval niet is gezorgd voor ontkoppeling tussen zowel technische als functionele brokken, is opnieuw een *stovepipe* gecreëerd, alleen nu een gedistribueerde *stovepipe* in plaats van een gecentraliseerde.

## *Conclusie*

Het vastleggen en hergebruiken van patronen, een weerslag van *'best practices'* van succesvolle projecten, vormt een belangrijke pijler om sneller projecten op te leveren waarbij een hoge kwaliteit gegarandeerd blijft. Ook binnen een integratieproject kunnen zij op deze wijze een waardevolle bijdrage leveren. Enige voorzichtigheid is echter wel geboden. Het ondoordacht gebruik van patronen als componenten om een architectuur vorm te geven kan er toe leiden dat de totaaloplossing op een hoger niveau leidt aan de kwaal die op een lager niveau door het patroon diende te worden opgelost. In het geval van applicatie integratie kan dit het geval zijn als ontkoppeling op integratieniveau niet serieus genoeg wordt genomen, en alleen op deelniveau wordt ingevuld.

**Auteurs**
Drs. P.J. Koning en Drs. B.A. Groenewoud adviseren in hun rol van Technology Consultant bij Cap Gemini Ernst & Young klanten uit diverse branches bij de realisatie van een E-Business en/of Enterprise Application Integration strategie middels het gebruik van state-of-the-art technologie.

**Literatuur**

[1] De koortsige hartslag van de 'e-bizz' - Vijf ijkpunten om van hype tot echt elektronisch zakendoen te komen, *Drs. B.A. Groenewoud en Drs. P.J. Koning*, 13 oktober 2000, Computable.

[2] Haarlemmerolie voor B2B en B2C - Centrale rol voor XML in een multi-tier client/server-architectuur, *Drs. P.J. Koning en Drs. B.A. Groenewoud*, 7 juli 2000, Computable.

[3] A pattern language, *Christopher Alexander, Sara Ishikawa, Murray Silverstein, Max Jacobson, Ingrid Fiksdahl-King, en Shlomo Angel*, 1977, Oxford University Press.

[4] Design Patterns – Elements of Reusable Object-Oriented Software, *Erich Gamma, Richard Helm, Ralph Johnsom en John Vlissides*, 1994, Addison-Wesley.