De Zeehavenpolitie Rotterdam is onder meer verantwoordelijk voor het controleren van personen die de Europese Unie in- en uitreizen op met name vrachtschepen. De nieuwe intranetapplicatie ZUIS, die in juni van dit jaar in gebruik is genomen, is de spin in het web van de gehele grensbewaking. Dit systeem ondersteunt de planning en de werkverdeling en bewaakt realtime de actuele voortgang van alle werkzaamheden, 24 x 7 uur per week. De hoge eisen gesteld door politie en justitie, hebben geleid tot een webapplicatie met een zeer innovatief karakter. Dit artikel geeft een indruk van de complexe combinatie van technologieën die is ingezet.

ZUIS: Java-applicatie voor grensbewaking

Technologische uitdagingen voor een intensief gebruikte intranetapplicatie

Bij de grensovergangen naar Duitsland en België wordt niet meer gecontroleerd, maar des te meer aandacht gaat uit naar onze zee- en luchthavens als buitengrenzen van de EU. Het werkterrein van de Zeehavenpolitie (de nieuwe naam voor de Rivierpolitie) strekt zich uit over zo'n 55 kilometer, vanaf Krimpen aan den IJssel tot op de Noordzee, voorbij Europoort. Zo'n dertigduizend maal per jaar vaart er een schip 's werelds grootste haven binnen. ZUIS is een anagram voor Zeescheepvaart Uitbreidbaar Integraal Systeem. Het verzamelt en levert een schat aan actuele informatie over onder meer schepen, scheepsbewegingen, personen en bedrijven in de Rotterdamse haven. Daarnaast worden grofweg de volgende werkprocessen ondersteund:

Planning. Hierin staan de diensten van de surveillance-eenheden centraal: de medewerkers die per auto of boot de schepen bezoeken om controles uit te voeren. Ook de diensten van chefs en piketmedewerkers kunnen worden ingepland.

Uitvoering. Aan de hand van instelbare beslissingsregels wordt automatisch bepaald welke controles er uitgevoerd dienen te worden en welke eenheden daarvoor ingezet worden. Er wordt voor elke inzet bijgehouden in welk stadium die verkeert en wat de volgende stap in het proces is. Hiermee samenhangend wordt ook voor elke dienstdoende eenheid bijgehouden waar deze eenheid mee bezig is, ook betreffende activiteiten die niet met de grensbewaking van doen hebben. Met name in de meldkamer, waar het werk gecoördineerd wordt, heeft men continu een overzicht op het scherm en kan men ingrijpen waar nodig.

Documentatie en administratie. Binnen ZUIS worden talrijke documenten beheerd. Het invullen van formulieren wordt vergemakkelijkt en ook andere administratieve handelingen worden ondersteund.

FIGUUR 1. Een webpagina van ZUIS

INFRASTRUCTUUR Er is zoveel mogelijk gebruik gemaakt van de bestaande infrastructuur. Elke gebruiker heeft een PC met Windows NT en Microsoft Office. De applicatie wordt benaderd met Internet Explorer 5.5, met een verbeterde XML parser. Ook op de servers draait Windows NT en verder de Microsoft Internet Information Server en de IBM WebSphere Application Server 3.5. De databases staan op UNIX servers. Het gebruikte DBMS is dat van Sybase. ZUIS kan ook mobiel gebruikt worden, in auto's en boten, via een beveiligde GPRS-verbinding. Voor de auto's is speciale hardware geselecteerd, om op een comfortabele wijze vanuit de bijrijdersstoel de applicatie te gebruiken.

> Het programmeerwerk vindt plaats op een vrij primitief niveau

ARCHITECTUUR De basis van de ZUIS-architectuur wordt gevormd door het bekende drie-lagenmodel: presentatielaag, logicalaag en gegevenslaag.

- De *presentatielaag* verzorgt de interface naar de gebruiker in de vorm van webpagina's.
- De *logicalaag* bevat alle bedrijfsfuncties, van een eenvoudige adreswijziging tot en met het automatisch verdelen van de uit te voeren controles over de dienstdoende eenheden. Als de gebruiker op een knop drukt, bijvoorbeeld "Opslaan", dan zal de presentatielaag de juiste bedrijfsfunctie van de logicalaag aanroepen om deze actie uit te voeren.
- De *gegevenslaag* staat gelijk aan de database. Met behulp van SQL kan de logicalaag de database benaderen.

Wat betreft de technische invulling hiervan volgt ZUIS de J2EE-standaard. In de rest van dit artikel worden de hiervoor beschreven elementen van deze architectuur verder uitgediept. Daarbij presenteer ik geen revolutionaire ideeën, wel geef ik een indruk van de gemaakte keuzes en de "design patterns" die zijn toegepast.

PRESENTATIELAAG

Webpagina's De webpagina's die de ZUIS-gebruiker te zien krijgt, zijn uiteindelijk HTML-pagina's, gekoppeld aan een cascading style sheet (CSS). Met behulp van JavaScript zijn er dynamische elementen aan toegevoegd, bijvoorbeeld om het formaat van datumvelden te controleren en om een veld onder bepaalde voorwaarden wijzigbaar of niet-wijzigbaar te maken.

Java Server Pages Voor het ontwikkelen van de webpagina's is gebruik gemaakt van JSP's (Java Server Pages). Een JSP is een dynamische Webpagina: alle vaste onderdelen van de pagina, HTML en JavaScript, staan er gewoon in, maar de variabele onderdelen, met name de bedrijfsgegevens, worden met behulp van Java uit de logicalaag opgevraagd.

Tag library's In ZUIS is gebruik gemaakt van zogenaamde tag libraries. Hiermee is HTML uitgebreid met specifieke ZUIS-tags, die in elke JSP gebruikt kunnen worden. Voordat een pagina naar de client gestuurd wordt, worden deze speciale tags eerst vertaald naar gewone HTML tags en JavaScript. Deze vertaling is geschreven in Java. Het voordeel van deze speciale tags is, dat veel voorkomende HTML/JavaScript-constructies op een centrale plek onderhouden kunnen worden. Op deze manier is bijvoorbeeld de autorisatie op invoervelden geregeld. Ook de inhoud van selectievelden wordt dynamisch opgehaald uit de database met behulp van tags uit de tag library.

PRESENTATIELAAG: XML/XSL Voor het weergeven van tabellen worden geen JSP's gebruikt. In plaats daarvan wordt een combinatie van XML en XSL gebruikt. Bij deze techniek worden eerst de bedrijfsgegevens, dat is het dynamische deel van de webpagina, naar de client verstuurd (in XML-formaat) en daarna pas de pagina-opmaak (in XSL-formaat). Om bijvoorbeeld een tabel van schepen te tonen, wordt eerst een lijst van scheepsinformatie in XML-formaat naar de client gestuurd. Hierbij wordt tevens de naam van een XSL-bestand doorgegeven. Met behulp van het XSL-bestand is de webbrowser in staat om van de kale lijst van schepen een webpagina in HTML-formaat te construeren. Wat betreft lay-out is er voor de gebruiker geen verschil te zien met een JSP.

Het grote voordeel van deze technologie is, dat er een optimale scheiding van presentatie en logica plaatsvindt. Dit maakt de applicatie in technisch opzicht flexibeler, wat gedurende de verdere evolutie van het product geld kan besparen. Helaas is er geen ontwikkeltool voorhanden dat XML/XSL voldoende ondersteunt, waardoor het opsporen van fouten zeer tijdrovend is. Dat laatste is de reden dat deze technologie voor ZUIS niet integraal is toegepast.

Een ander voordeel van XSL is, dat de opmaak bestuurd kan worden met behulp van JavaScript. Op deze wijze kan bijvoorbeeld een tabel op de client gesorteerd worden. Als een ZUIS-gebruiker op een kolomtitel klikt, wordt de tabel gesorteerd op die kolom, zonder een nieuwe webpagina van de server op te vragen. Het sorteren is in een oogwenk geschied en de server wordt

niet belast met het opnieuw opvragen van dezelfde tabel. Vandaar dat voor tabellen toch de voorkeur is gegeven voor XML/XSL boven JSP.

PRESENTATIELAAG: CONTROLLER SERVLET Een 'servlet' is een onderdeel van de presentatielaag, geschreven in Java, dat op de server draait en dat verantwoordelijk is voor het verwerken van berichten (HTTP requests) die vanaf de client naar de server worden gestuurd. De meeste webapplicaties die in Java geschreven zijn, hebben één servlet voor elke webpagina; deze servlet handelt alle acties af die de gebruiker op deze pagina uitvoert. Voor ZUIS is echter gekozen voor één centrale servlet, de zogenaamde controller servlet (figuur 2). De wijze waarop de diverse acties worden afgehandeld is niet ingeprogrammeerd in de servlet, maar is vastgelegd in een configuratiebestand in XML-formaat. De controller servlet leest dit bestand en bepaalt aan de hand van de regels die daarin zijn gespecificeerd hoe hij moet reageren op de input van de gebruiker. De 'flow' van de applicatie, dat is met name de volgorde waarin de webpagina's worden getoond, is dus eenvoudig te wijzigen door het configuratiebestand aan te passen.

Tevens biedt de controller servlet een oplossing voor een belangrijk dilemma waar elke webapplicatie mee te kampen heeft. Met een webbrowser heeft de gebruiker namelijk de mogelijkheid om terug te gaan naar reeds eerder bezochte pagina's. Het is voor een webapplicatie vaak echter moeilijk, zo niet onmogelijk, om een willekeurige situatie uit het verleden weer op te roepen. Voor ZUIS is de volgende oplossing gekozen. De 'page history' van de webbrowser wordt onzichtbaar gemaakt voor de gebruiker. In plaats daarvan wordt door de controller servlet een eigen page history bijgehouden. De laatste drie bezochte pagina's worden rechtsboven op elke webpagina getoond, in de vorm van drie hyperlinks (figuur 1). Als de gebruiker nu op zo'n hyperlink klikt, zal de controller servlet er vervolgens voor zorgen dat de betreffende pagina weer wordt getoond, met actuele gegevens uit de database. Bij het verwijderen van gegevens uit de database, wordt de webpagina niet in de page history opgenomen, om te voorkomen dat de gebruiker tracht terug te keren naar reeds verwijderde gegevens.

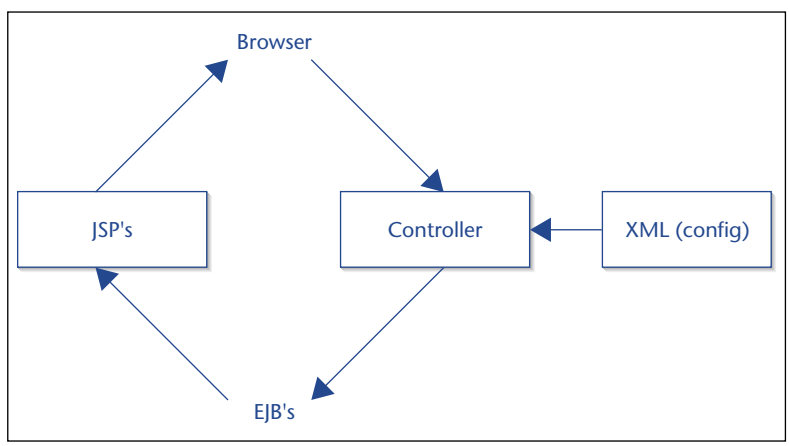
COMPONENTENARCHITECTUUR De logicalaag is geheel gebaseerd op componenten: functioneel afgebakende software-onderdelen die zo min mogelijk van elkaar afhankelijk zijn. De onderlinge afhankelijkheden tussen deze componenten vormen een hiërarchie, met andere woorden: als component A afhankelijk is van B, dan is B niet afhankelijk van A. Met deze componentenarchitectuur is de complexiteit van ZUIS opgesplitst in overzichtelijke onderdelen. Bovendien biedt het mogelijkheden om delen van ZUIS in de toekomst te integreren met andere applicaties. Het voert te ver om de complete componentenhiërarchie te bespreken, vandaar dat we ons beperken tot een vereenvoudigd schema van groepen componenten, weergegeven in figuur 3.

De basis van de hiërarchie wordt gevormd door de zogenaamde "utilities": componenten die niet afhankelijk zijn van andere componenten. Deze componenten vormen een infrastructuur van algemeen bruikbare software, onder meer. voor het schonen van de database, voor het autorisatiemechanisme en voor het bijhouden van een logboek van gewijzigde gegevens.

> Het is te duur om allerlei DBMS-taken in de componenten te programmeren, terwijl het DBMS dat beter zelf kan

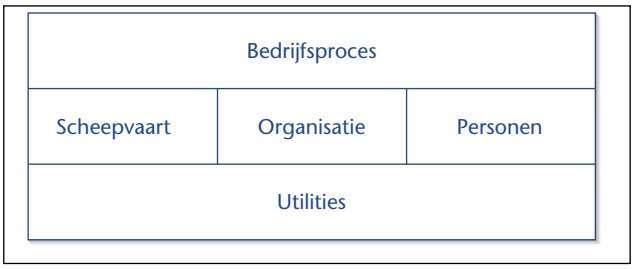
GEGEVENSLAAG In de volgende laag bevindt zich een aantal gegevensgerichte componenten, onderverdeeld in de groepen "scheepvaart", "organisatie" en "personen". Elk van deze componenten beheert een aantal databasetabellen en implementeert de bijbehorende bedrijfsregels. Zo is de component "Schip" (die uiteraard tot de scheepvaartcomponenten behoort) verantwoordelijk voor de tabellen "Schip" en "TypeSchip". Andere componenten mogen deze tabellen uitsluitend benaderen via component "Schip".

De procescomponenten vormen een overkoepelende groep. Er is bijvoorbeeld een component die de voortgang van een grenscontrole bijhoudt en een andere component die een risicoanalyse uitvoert op elk schip dat de haven binnenloopt. Sommige van deze componenten beheren bepaalde tabellen, net als de gegevensgerichte componenten; dit zijn dan tabellen waarin de



FIGUUR 2. Informatiestromen in de presentatielaag (vereenvoudigd)

Adv.
Compuware



FIGUUR 3. Groepen van componenten in ZUIS

procesgegevens en de bedrijfsregels zijn opgeslagen. Zo beheert de risicoanalyse-component een beslissingsmatrix. In de presentatielaag bevinden zich webpagina's waarmee deze beslissingsmatrix kan worden ingevuld. Op deze wijze kan een daartoe bevoegde gebruiker te allen tijde bepalen op basis van welke criteria de risicoanalyse moet plaatsvinden (bijvoorbeeld bepaalde kenmerken van het schip).

EVENTS Het komt vaak voor, dat er in een scheepvaartcomponent iets gebeurt, dat van belang is voor een procescomponent. Als er bijvoorbeeld een schip de haven binnenvaart, dan moet er een controle uitgevoerd worden. Het is niet de bedoeling dat een scheepvaartcomponent rechtstreeks een procescomponent benadert. Dat zou tegen de hierboven geschetste architectuur ingaan, waardoor de overzichtelijke opbouw van de applicatie zou worden geschaad en waardoor de scheepvaartcomponent niet meer los van de procescomponent te hergebruiken zou zijn.

Om dit te voorkomen, is er een algemeen eventmechanisme gebruikt (een utility, zie figuur 3). Een willekeurige component kan zich bij het eventregister aanmelden als zijnde geïnteresseerd in een bepaald soort event (gebeurtenis). We noemen dit een abonnement. Een procescomponent kan bijvoorbeeld geïnteresseerd zijn in de aankomst van schepen en daarom een abonnement nemen op deze gebeurtenis. Het eventregister houdt alle abonnementen bij in de database. Zodra er nu een schip binnenkomt, zal de betreffende scheepvaartcomponent hiervan melding maken bij het eventregister, dat op zijn beurt alle abonnees op de hoogte brengt. Het eventregister weet niets van de abonnees, behalve dan dat ze event-notificaties kunnen ontvangen. Hierdoor is het eventregister niet afhankelijk van de aanwezigheid van specifieke procescomponenten en blijft de applicatie overzichtelijk en herbruikbaar.

LOGICALAAG: ENTERPRISE JAVA BEANS Elke component is gebouwd volgens de Enterprise Java Beans (EJB) standaard. Deze standaard maakt onderscheid tussen entity beans en session beans. Een entity bean kan gebruikt worden als representatie van een bedrijfsobject, bijvoorbeeld een schip, terwijl een session bean bedoeld is om een taak uit te voeren voor een gebruiker. Voor ZUIS is ervoor gekozen om uitsluitend session beans te gebruiken, om de volgende redenen:

- Entity beans zijn te traag, in de gegeven omstandigheden. In ZUIS wordt met duizenden objecten tegelijk gewerkt; evenzovele entity beans zouden teveel geheugen en processortijd vergen.
- Entity beans ondersteunen een 1:1-verhouding tussen bedrijfsobjecten en databasetabellen. In ZUIS is dit echter niet het geval. Een bedrijfsobject kan verdeeld zijn over meer dan één tabel en sommige tabellen vertegenwoordigen een relatie tussen twee bedrijfsobjecten.

Voor de session beans is een speciaal "persistency framework" gebouwd, om de database op een eenduidige manier te benaderen. Ook dit framework behoort tot de utilities van de logicalaag (figuur 3). Het vormt als het ware een Java-schil om de database heen en het is de enige plaats waar SQL-commando's worden uitgevoerd.

BLACK-BOXPRINCIPE Elke component (EJB) heeft zijn eigen tabellen in de relationele database. Deze tabellen mogen uitsluitend door die component gebruikt worden. De rest van de applicatie kan de inhoud van de tabel via de interface van deze component opvragen. Op deze manier wordt de fysieke bestandsstructuur van een bedrijfsobject afgeschermd, zodat de bestandsstructuur naderhand geoptimaliseerd kan worden, zonder de hele applicatie te hoeven herschrijven. Wie dit black-boxprincipe zo zuiver mogelijk wil toepassen, zal volhouden dat het niet is toegestaan om bijvoorbeeld de referentiële integriteit door het DBMS te laten regelen (althans niet de integriteit van referenties tussen tabellen die tot *verschillende* componenten behoren). Als er een afhankelijkheid is tussen twee componenten, dan moeten ze via hun interfaces communiceren, niet via het DBMS, omdat ze geen kennis mogen hebben van elkaars interne gegevensstructuur.

> In de presentatielaag bevinden zich webpagina's waarmee deze beslissingsmatrix kan worden ingevuld

Helaas is dit niet vol te houden. Het is in diverse opzichten te duur om allerlei DBMS-taken in de componenten te programmeren, terwijl het DBMS die veel beter zelf kan uitvoeren. Voor ZUIS is van tevoren afgesproken wat wel en niet is toegestaan. Zaken zoals refe-

Adv.
Precise Software

rentiële integriteit en 'cascading delete' mogen wel degelijk in de gegevenslaag worden vastgelegd, ook omdat de impact van wijzigingen op dit gebied beperkt blijft tot de gegevenslaag. Het is daarentegen in principe verboden om vanuit een component een zoekvraag te stellen waarin naar een tabel van een andere component wordt gerefereerd, dus een "join" is alleen toegestaan op de eigen tabellen. Niettemin zijn er een aantal uitzonderingen gemaakt voor gevallen waarin het vasthouden aan dit principe de applicatie te traag zou maken.

Een concessie van geheel andere aard kwam voort uit de noodzaak om de database te laten bevragen door externe toepassingen, zoals een report writer. Deze applicaties hebben een SQL-interface nodig en geen Java-interface. Om hieraan tegemoet te komen, zijn er zogenaamde "views" gedefinieerd in het DBMS. Een view kan worden bevraagd alsof het een tabel is, maar in feite is het slechts een afbeelding van één of meer tabellen. Het DBMS vertaalt alle acties die op de views worden uitgevoerd naar acties op de eigenlijke tabellen. Op deze manier blijft de fysieke bestandsstructuur verborgen voor externe applicaties. De views worden beschouwd als onderdeel van de interfaces van de betreffende componenten.

KOPPELINGEN ZUIS is gekoppeld aan een aantal externe systemen. Voor de gegevensoverdracht is zoveel mogelijk gebruik gemaakt van XML. Hieronder volgt een kort overzicht van de gekoppelde externe systemen.

- Actuele scheepsbewegingen worden elke 10 minuten aangeleverd vanuit Datadirk, een systeem van het havenbedrijf (KSD). Gebruikers kunnen dit volgen op een overzichtsscherm, dat zichzelf automatisch ververst.
- Bemanningslijsten en passagierslijsten in EDIFACT-formaat die via e-mail binnenkomen worden automatisch in ZUIS verwerkt. Personen op deze lijsten die nog niet in ZUIS bekend zijn, worden toegevoegd aan het personenbestand.
- Bij elke wijziging in het personenbestand wordt er automatisch gecontroleerd of de betreffende persoon nationaal of internationaal gesignaleerd staat.
- Er is een koppeling met het algemene politiesysteem X-pol, waarmee onder meer processen verbaal worden opgemaakt. ZUIS en X-pol wisselen onderling informatie uit.
- Ingescande documenten worden bewaard in Zy-Image. Deze documenten kunnen eenvoudig worden gekoppeld aan ZUIS-gegevens, door ZUIS een sticker te laten printen, waarop door middel van een streepjescode een persoon of schip wordt geïdentificeerd. Vóór het inscannen van het document moet deze kleine sticker erop geplakt worden. Vanuit ZUIS kan men even later het ingescande document op de monitor laten verschijnen.
- Door middel van een koppeling met Word kan men bepaalde formulieren (Word-sjablonen) laten invullen. Gegevens die bij ZUIS bekend zijn, hoeft de gebruiker dus niet opnieuw in te typen in Word.
  Later dit jaar volgen nog meer koppelingen.

EVALUATIE Ondanks de kinderziektes van het nieuwe informatiesysteem, zijn de gebruikers zeer tevreden over ZUIS. De applicatie geeft een adequate ondersteuning van het grensbewakingsproces, attendeert de gebruiker op het juiste moment op relevante informatie en daarnaast zijn alle gegevens gemakkelijk te vinden. Het heeft er onder meer toe geleid dat de zwaarte van de personencontroles beter is afgestemd op de werkelijke grensveiligheidsrisico's, waardoor de pakkans is vergroot. Wat opvalt is dat voor een grote intranetapplicatie als ZUIS een heel scala aan technologieën ingezet moet worden en dat er dus veel verschillende expertises in het team aanwezig moeten zijn. Hierdoor is ook een duidelijke visie op de technische architectuur vereist.

Het programmeerwerk vindt plaats op een vrij primitief niveau. Java is een goed gestructureerde taal waarvoor prima ontwikkeltools beschikbaar zijn, maar dit geldt duidelijk niet voor JavaScript en XSL. Standaardcomponenten zoals pull-down menu's en pop-ups zijn wel te krijgen, maar hieraan moet vaak nog gesleuteld worden om ze in te passen in de applicatie. Een goede kennis van beschikbare open source software is geen overbodige luxe. Al met al zijn de kosten per functiepunt veel hoger dan voor een met 4GL-tools gebouwde applicatie. Sinds de aanvang van het ZUIS-project zijn de tools voor het bouwen van intranetapplicaties wel verbeterd en deze trend zal zich ongetwijfeld doorzetten. Steeds meer bedrijven en instellingen ontdekken immers dat het intranet een uniek medium vormt voor het geïntegreerd ontsluiten en verwerken van al hun bedrijfskritische informatie en de IT-wereld zal hierop inspelen.

Hans Admiraal
Dhr. Admiraal is software architect en projectleider bij Ordina
(hans.admiraal@ordina.nl)