Uitbreiding functionaliteiten BookDB-XML

Naam student: Christian Jansen
Nummer: 20013115

Bedrijf: Sabern Computer Software BV
Mentor: K.Lelieveld

Examinatoren: P.C.M. Borsboom
J. van Peski

Referaat

C.W. Jansen, afstudeerverslag afstuderen, Leiderdorp, Sabern Computer Software BV, februari 2005 - juni 2005

De auteur van dit afstudeerverslag studeert Informatica en Informatiekunde aan de Haagse Hogeschool. In dit afstudeerverslag zijn de activiteiten beschreven die de auteur tijdens zijn afstudeerperiode heeft uitgevoerd bij Sabern Computer Software BV te Leiderdorp om de functionaliteiten van BookDB-XML uit te breiden.
Dit vond plaats in de periode van februari 2005 tot en met juni 2005.

Trefwoorden:

- BookDB-XML
- XML
- Java
- IAD
- Adobe FrameMaker

Voorwoord

Voor u ligt het afstudeerverslag waarin precies staat beschreven hoe het proces van mijn afstuderen bij Sabern BV is verlopen.

Hierbij wil ik graag Karen Lelieveld bedanken voor het ter beschikking stellen van een afstudeerproject. Ook bedank ik hierbij Gerben van der Laan voor de hulp die hij mij gaf als ik ergens niet uit kwam.

Ik wens u evenveel plezier toe met het lezen van dit verslag als dat ik had tijdens het werken aan het project.

Leiderdorp, 3 juni 2005

Christian Jansen

Inhoudsopgave

1. Inleiding 6
2. De opdrachtgever: Sabern BV 7
   - 2.1 Geschiedenis 7
   - 2.2 Bedrijfsactiviteiten 7
   - 2.3 Bedrijfsstructuur 7
   - 2.4 Plaats opdrachtnemer 8
   - 2.5 Concurrentie 8
   - 2.6 Toekomstvisie 8
3. De opdracht 9
   - 3.1 Titel afstudeeropdracht 9
   - 3.2 Omgevingsschets 9
   - 3.3 Omschrijving BookDB-XML 9
   - 3.4 Probleembeschrijving 9
   - 3.5 Doelstelling 9
   - 3.6 Benodigde software 9
   - 3.7 Werkzaamheden 10
   - 3.8 Methoden 10
   - 3.9 Technieken 10
   - 3.10 Op te leveren produkten 10
4. Oriëntatie 11
   - 4.1 Verdediging gekozen systeemontwikkelingsmethode 11
   - 4.2 Verdediging gekozen modelleertaal 11
   - 4.3 Aanvangssituatie 12
5. BookDB-XML 13
   - 5.1 Wat is BookDB-XML en wat kan het? 13
   - 5.2 De opties 13
   - 5.3 Het proces van het maken van een eindprodukt 13
   - 5.4 De architectuur van BookDB-XML 14
   - 5.5 Voorbeeld template 15
6. Extensible Markup Language 16
   - 6.1 Wat is XML? 16
   - 6.2 Het gebruik van XML in BookDB-XML 16
   - 6.3 Hoe werkt XML? 16
   - 6.4 Een klein voorbeeld 16
7. Het proces 18
   - 7.1 De aanpak van het proces 18
   - 7.2 Definitiestudies 19
   - 7.3 Pilotontwikkelingen 21
   - 7.4 Invoeringen 22
8. Opgeleverde produkten 23
   - 8.1 Pilot I 23
   - 8.2 Pilot II 24
   - 8.3 Pilot IIIa 26
   - 8.4 Pilot IIIb 28
   - 8.5 Pilot IV 32
   - 8.6 Pilot V 33
   - 8.7 Pilot VI 34
   - 8.8 Pilot VII 35
   - 8.9 Pilot VIII 39
9. Conclusie 45
10. Evaluatie 46
    - 10.1 Het proces 46
    - 10.2 Het produkt 46

10.3 Problemen 46
10.4 Verbeterpunten 46

Bijlagen

Bijlage A - Definitieve opdrachtsomschrijving
Bijlage B - Literatuurlijst / Bronvermelding
Bijlage C - Contactinformatie
Bijlage D - Opgeleverde produkten
Bijlage E - Buglijst
Bijlage F - Visiedocument

1. Inleiding

In dit afstudeerverslag is mijn gehele afstudeerproject gedetailleerd beschreven. Dit heeft als doelstelling de lezer op de hoogte te stellen van de activiteiten die ik heb uitgevoerd, de keuzes die ik hierbij heb gemaakt en een voorstelling te geven van de produkten die ik geproduceerd heb.

Dit eindverslag begint met een klein bedrijfsprofiel van mijn gastbedrijf Sabern BV, gevolgd door de omschrijving van de opdracht. Hierna volgt de beschrijving van mijn oriëntatie, gevolgd door een uitgebreide beschrijving van BookDB-XML en XML. Vervolgens is een apart hoofdstuk gewijd aan het proces, waarna er een uitgebreide omschrijving wordt gegeven van de produkten die ik heb gemaakt. Tenslotte volgt, na een conclusie, een uitgebreide evaluatie van het proces waarin ik mijn visie en mening weergeef over aspecten van het afstudeerproject, zoals het proces, de produkten, problemen en eventuele verbeterpunten.

2. De opdrachtgever: Sabern BV

Dit hoofdstuk geeft een beschrijving weer van de organisatie waarin ik mijn afstudeerperiode heb doorgebracht: Sabern Computer Software BV.

2.1 Geschiedenis

Sabern Computer Software BV is een softwarebedrijf dat in 1990 is opgericht door K. Lelieveld en L. Scholtes. Het bedrijf is gevestigd in Leiderdorp. Sabern specialiseerde zich in de beginjaren van haar bestaan in de verkoop van grafische hardware (zoals Silicon Graphics systemen, professionele kleurenprinters en scanners) en standaard software. Langzamerhand specialiseerde het bedrijf zich meer in de hieronder genoemde bedrijfsactiviteiten, omdat de directie vond dat Sabern hier meer mogelijkheden op de markt zou hebben. Een andere reden voor deze specialisatie is dat de vraag hiernaar steeds groter werd.

2.2 Bedrijfsactiviteiten

Sabern BV houdt zich hoofdzakelijk bezig met de volgende vier bedrijfsactiviteiten:

- Het ontwikkelen en leveren van standaard software
  Dit betreft zowel produkten als Adobe als produkten van Sabern BV, zoals BookDB, BookDB-XML en DocumentDB.

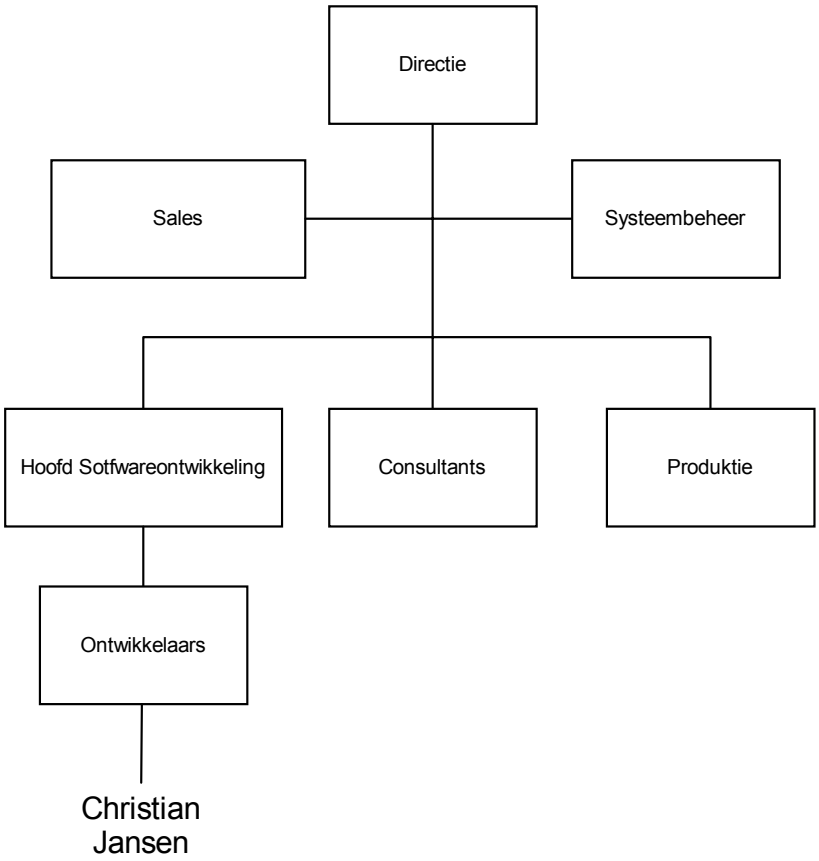
- Het ontwikkelen en leveren van maatwerk toepassingen
  Deze toepassingen verschillen per klant.

- Het geven van cursussen
  Dit betreft cursussen over Adobe-produkten en produkten van Sabern BV. Ook worden er cursussen gegeven over database publishing, document publishing, content management en XML.

- Het verlenen van consultancy
  Sabern BV adviseert klanten op het gebied van database publishing, document publishing en content management.

2.3 Bedrijfsstructuur

Om een bedrijf zo goed mogelijk te laten functioneren, is structuur in de opbouw van het bedrijf nodig. In het volgende figuur is de bedrijfsstructuur (organigram) van Sabern BV te zien.



Bovenaan staat de directie. Deze wordt bijgestaan door de afdeling Sales en Systeembeheer. Het doel van de afdeling Sales is het werven van klanten. Deze afdeling bestaat pas sinds kort. De afdeling Systeembeheer zorgt ervoor dat de technische infrastructuur bij Sabern werkt en blijft werken, zodat de werknemers hun werk kunnen doen. Het Hoofd Softwareontwikkeling is tevens de projectleider van BookDB-XML. Deze is verantwoordelijk voor alle door Sabern ontwikkelde software. De mensen van de afdeling Consultants leveren ondersteuning aan klanten. Bij de afdeling Produktie worden er produkten voor de klanten geproduceerd, zoals bijvoorbeeld catalogi. Momenteel zijn er tien mensen werkzaam bij Sabern BV. Dit is exclusief stagiairs en freelancers.

2.4 Plaats opdrachtnemer

De opdrachtnemer (Christian Jansen) werkt bij de “afdeling” Ontwikkelaars.

2.5 Concurrentie

Sabern ondervindt vrijwel geen concurrentie. Er is echter wel één concurrent die soortgelijke produkten maakt als Sabern, maar meestal gaat de voorkeur van de klant uit naar de produkten van Sabern en niet die van deze concurrent.

2.6 Toekomstvisie

Er zijn plannen om Sabern uit te breiden. Zo is er sinds kort een afdeling Sales waar potentiële klanten worden gezocht voor produkten en diensten van Sabern. De verwachting is dat er dan meer klanten bijkomen. Indien dit zal gaan gebeuren, zullen de afdelingen Ontwikkelaars en Consultants ook uitgebreid worden.

3. De opdracht

In dit hoofdstuk is de opdracht waaraan ik heb gewerkt tijdens mijn afstudeerperiode uitvoerig weergegeven. Dit hoofdstuk bevat de definitieve opdrachtsomschrijving (tevens als apart document te vinden in Bijlage A).

3.1 Titel afstudeeropdracht

De titel van de afstudeeropdracht luidt:

“Uitbreiding van functionaliteiten van BookDB-XML bij Sabern Computer Software”

3.2 Omgevingsschets

Sabern Computer Software B.V., opgericht in 1990, assisteert organisaties bij het opzetten van database publishing, content management, XML-applicaties, dynamische Websites en documentenbeheer. Sabern ontwikkelt en levert standaard software en maatwerk toepassingen. Verder geeft Sabern cursussen en verleent consultancy.

3.3 Omschrijving BookDB-XML

BookDB-XML is database publishing software voor het maken van teksten, tabellen, plaatjes en grafieken. Het systeem maakt gebruik van XML voor de documentopmaak. Met BookDB-XML kan een XML document worden gemaakt voor papieren of digitale uitvoer. Met BookDB-XML kunnen de gegevens afkomstig van database management systemen worden gebruikt als basis voor opgemaakte documenten in XML. In het template kunnen de vaste elementen en de opmaakinformatie worden gedefinieerd. In XSLT’s worden de selecties op de database gedefinieerd en wordt de uitvoer opgebouwd.

3.4 Probleembeschrijving

Sabern Computer Software wenst dat de functionaliteit van BookDB-XML uitgebreid wordt. Er ontbreekt nog een aantal functionaliteiten. Deze ontbrekende functionaliteiten dienen te worden geanalyseerd, ontworpen en geïmplementeerd. Dit betreft de volgende punten:

- Finetunen van bestaande functionaliteiten.
- Ontwerpen en implementeren van algemene nieuwe functionaliteiten.
  Denk hierbij aan seriële executie van taken en aan de mogelijkheid om in een template iteratie en selectie op te nemen.
- Het ontwerpen en eventueel implementeren van een grafische user-interface voor het samenstellen van templates.

3.5 Doelstelling

De doelstelling van de feitelijke opdracht is de hierboven genoemde functionaliteiten te realiseren, zodat de functionaliteit van BookDB-XML uitgebreid wordt.

3.6 Benodigde software

De volgende software zal gebruikt worden:

- Adobe FrameMaker voor het maken van documenten
- Software voor het maken van ontwerpen (Paraben’s FlowCharter)
- Een texteditor voor het schrijven en bekijken van de source-code (EditPlus 2)
- Een Java-compiler en runtime-environment
- BookDB-XML

Er is documentatie over BookDB-XML, bestaande uit gebruikershandleidingen en systeemdocumentatie in de source-code beschikbaar.

3.7 Werkzaamheden

In het kader van de afstudeeropdracht zullen de volgende activiteiten verricht worden:

1. Verplichte functionaliteiten:
    - Definitiestudie
    - Pilotontwikkeling en invoering pilot I, II, III, IV en V

2. Overige functionaliteiten:
    - Definitiestudie
    - Pilotontwikkeling en invoering pilot VI, VII en VIII

3.8 Methoden

Bij de uitvoering van de opdracht zal IAD gehanteerd worden. De iteratiestrategie die gebruikt zal worden is een combinatie van incrementeel ontwikkelen en incrementeel opleveren. Voor de door Sabern verplicht gestelde functionaliteiten wordt eenmalig de definitiestudie uitgevoerd, gevolgd door een pilotontwikkeling voor elke pilot. Hierna wordt eventueel voor elke pilot of meerdere pilots tegelijk de invoering uitgevoerd. Voor de overige functionaliteiten wordt nogmaals een definitiestudie uitgevoerd, gevolgd door een pilotontwikkeling voor elke pilot met daarna de invoering voor elke pilot of voor meerdere pilots tegelijk.

3.9 Technieken

De volgende technieken zullen gebruikt worden:

- Communicatie met collega's door middel van kleine workshops
- Timeboxing
- Juicy Bits First
- Java
- Extensible Markup Language (XML)
- Extensible Stylesheet Language Transformations (XSLT)
- Unified Modelling Language (UML)

3.10 Op te leveren produkten

- Plan van aanpak
- Ontwikkelscenario
- Systeemeisen
- Systeemconcept
- Pilotplan
- Pilotontwikkelplannen
- Ontwerpen van de software-bouweenheden
- Pilots
- Geactualiseerde handleidingen

4. Oriëntatie

Om structuur aan te brengen in het project en om het project met succes te kunnen realiseren, is een systeemontwikkelingsmethode nodig. Voor het maken van duidelijke ontwerpen is een modelleertaal nodig. In dit hoofstuk heb ik beschreven voor welke systeemontwikkelingsmethode en modelleertaal ik heb gekozen en waarom.

4.1 Verdediging gekozen systeemontwikkelingsmethode

Voor het oplossen van de probleemstelling heb ik gekozen voor IAD en wel om de volgende redenen:

- IAD stelt mij en de opdrachtgever in staat om met behulp van kleine workshops goed met elkaar te communiceren. Hierdoor is de kans op misverstanden minimaal. Dit komt het project ten goede. Mocht het toch de verkeerde kant op gaan, dan kan het makkelijk bijgesteld worden.
- De systeemeisen van BookDB-XML waren nogal wijzigingsgevoelig. Dit blijkt ook uit het feit dat de opdrachtsomschrijving aan het begin van de afstudeerperiode is veranderd. IAD biedt de mogelijkheid om met dit soort situaties flexibel om te gaan door het gebruik van iteraties en evaluaties.
- De meest cruciale delen van het systeem worden al in een vroeg stadium opgeleverd. Dit is handig als de opdrachtgever dat deel al gelijk wil gebruiken. Dat is in dit geval ook zo. Elke nieuwe functie van BookDB-XML kon immers gelijk worden gebruikt binnen het bedrijf zelf.
- Met behulp van IAD kan ook de haalbaarheid van het project beter ingeschat worden. Hiermee wordt voorkomen dat ik te veel of te weinig hooi op mijn vork neem.
- Ik ben erg bekend met IAD, omdat het mij op de Haagse Hogeschool met de paplepel is ingegoten. Hierdoor ben ik goed op te hoogte van wat er allemaal mogelijk is met IAD en wat er niet mogelijk is met IAD.

Alle fasen van IAD kunnen op veel verschillende manieren doorlopen worden. Hoe dit precies is gebeurd tijdens dit project, is omschreven in het ontwikkelscenario wat verwerkt is in het plan van aanpak. Voor alle informatie die ik heb gebruikt omtrent IAD heb ik het boek van R.J.H. Tolido gebruikt (zie Bijlage B - Literatuurlijst).

4.2 Verdediging gekozen modelleertaal

Voor het analyseren en ontwerpen van de functionaliteiten heb ik gekozen om Unified Modeling Language (UML) te gebruiken, omdat het een zeer uitgebreide object georiënteerde modelleertaal is waarmee je alle kanten op kunt. UML is de enige modelleertaal die ik goed beheers, bovendien had ik er al veel ervaring mee opgedaan op school.

Bij Sabern BV maakt men standaard gebruik van Paraben's FlowCharter. Dit is software dat met name bedoeld is om flowcharts mee te maken. De eigenschappen van flowcharts komen sterk overeen met de eigenschappen van het activiteitendiagram van UML. Beide zijn bedoeld om een stroom van activiteiten en de daarbij behorende keuzes en volgorde te beschrijven. Het enige verschil is dat de gebruikte iconen in Paraben's FlowCharter hierbij ietswat verschillen. Het pakket kan ook gebruikt worden om andere diagrammen mee te maken, alleen voldoen deze vaak niet aan de exacte regels van UML, omdat het pakket sommige componenten niet ondersteunt. Zo is het bijvoorbeeld niet mogelijk om aggregatiepijltjes te tekenen in een klassendiagram en ook het tekenen van losse pijltjes om de richting van een associatie mee aan te geven wordt niet ondersteund. Het maken van een klassendiagram is niet mogelijk, omdat operaties en attributen niet kunnen worden omschreven. Verder is er geen mogelijkheid om de use-cases af te bakenen in een systeem door er een vierkantje eromheen te tekenen.

Omdat ik tijdens dit project veel bezig was met het ontwerpen en aanpassen van met name methoden en processen, was de flowchart een ideaal middel om dit weer te geven.

Bij de functionaliteiten waar de gebruiker interactie voert met een GUI (Pilot IIIb en VIII), heb ik gebruik gemaakt van use-cases bij de analyse in het systeemconcept. Omdat bij de overige pilots

de acties van de gebruiker niet centraal staan of onveranderd blijven, maar de nadruk meer ligt op de werking van het systeem zelf, heb ik daar geen gebruik gemaakt van use-cases.

Van klassendiagrammen heb ik alleen gebruik gemaakt om de structuur van bepaalde XML-elementen mee aan te geven. Alleen hebben de klassen hier geen attributen en operaties, omdat het slechts XML-elementen betreft. BookDB-XML bestaat slechts uit drie hoofdklassen met elk tientallen operaties. Het enige dat er steeds aan deze klassen veranderde gedurende het project, waren nieuwe operaties of operaties die aangepast werden. Het is daarom ook niet interessant om een klassendiagram van BookDB-XML zelf te maken. Omdat BookDB-XML tevens gebruik maakt van verschillende bestanden en bibliotheekbestanden, kan er geen klassendiagram van gemaakt worden. Daarom heb ik ervoor gekozen om de architectuur van BookDB-XML gewoon weer te geven in een componentendiagram. Een componentendiagram beschrijft de softwarecomponenten en hun afhankelijkheden ten opzichte van elkaar, zodat de structuur van de code wordt uitgebeeld.

Omdat ik met behulp van use-cases, componentdiagrammen, flowcharts (activiteitendiagrammen) en overige schema's met de daarbij behorende ondersteundende tekst mijn analyses en ontwerpen goed duidelijk kon maken voor zowel de projectleider als voor mijzelf en deze technieken een goede ondergrond vormden om met de implementatie te beginnen, heb ik ervoor gekozen deze technieken te gebruiken bij mijn analyses en ontwerpen. Voor alle informatie die ik heb gebruikt omtrent UML, heb ik het boek van Hans-Erik Eriksson & Magnus Penker gebruikt (zie Bijlage B - Literatuurlijst).

4.3 Aanvangssituatie

Voor de aanvang van mijn afstudeerproject, heb ik een tweedaagse basiscursus FrameMaker gevolgd op het kantoor van Sabern BV. Na het volgen van deze cursus was ik in staat uitgebreide documenten te maken in Adobe FrameMaker. Tijdens het werken met Adobe FrameMaker heb ik de boeken "Classroom in a book" en "The Complete Reference FrameMaker 7" gebruikt (zie Bijlage B - Literatuurlijst).

Op de eerste dag van mijn afstudeerperiode bleek dat ik mijn opdrachtsomschrijving moest gaan veranderen. Het was de bedoeling dat mijn afstudeerstage zou aansluiten op de bevindingen van een stagiair die vóór mij stage had gelopen. Het is deze stagiair echter niet gelukt zijn doelstellingen te behalen, waardoor een aantal punten in mijn afstudeeropdracht (nog) niet door kon gaan. Dit betrof de grafische user-interface voor het maken van templates, de koppeling met het vertaalgeheugen en de uitbreiding van de functies van het content management systeem. Omdat deze punten eerst door Sabern BV zelf uitgewerkt dienden te worden, kon hier nog niet aan gewerkt worden en kregen deze functionaliteiten een lagere prioriteit. Het finetunen van bestaande functionaliteiten en het ontwerpen en implementeren van nieuwe functionaliteiten kregen de hoogste prioriteit. Sabern BV had al een lijst gemaakt met daarin de globale systeemeisen die gerealiseerd dienden te worden. Zodra deze gerealiseerd zouden zijn, kon pas verder worden gegaan met de grafische user-interface voor het maken van templates, de koppeling met het vertaalgeheugen en de uitbreiding van de functies van het content management systeem, mits de projecttijd dat toe zou laten. Hier kon eventueel ook nog een licentienummer-systeem worden gerealiseerd en een workaround worden ontwikkeld voor een Java-bug dat BookDB-XML teisterde.
Pas nadat de functionaliteiten met de hoogste prioriteit waren gerealiseerd, werd er een selectie gemaakt voor de functionaliteiten die hierna gerealiseerd dienden te worden. Deze staan omschreven in de opdrachtsomschrijving.

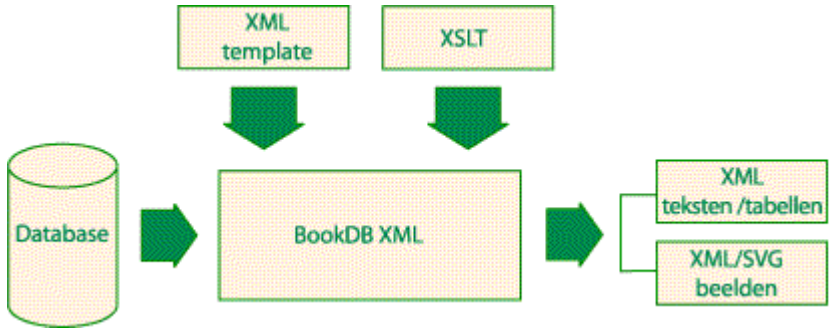
Van BookDB-XML was alleen de gebruikershandleiding en documentatie in de source-code beschikbaar, geen ontwerpen. Het project is door medewerkers van Sabern gemaakt zonder daarbij gebruik te maken van ontwerpen. Dit was voor mij nadelig, omdat ik nu zelf de broncode van BookDB-XML uitvoerig moest gaan bekijken om de precieze werking van het pakket te snappen.

5. BookDB-XML

In dit hoofdstuk is een uitgebreide beschrijving van BookDB-XML gegeven, zodat een beeld kan worden gevormd van hoe het precies werkt en wat het doet. Hiermee kan een beter beeld worden gevormd van de afstudeeropdracht. Dit hoofdstuk beschrijft de versie BookDB-XML waar ik aan het begin van het afstuderen mee begon.

5.1 Wat is BookDB-XML en wat kan het?

BookDB-XML is database publishing software. Het kan worden gebruikt om bijvoorbeeld produktcatalogi, prijslijsten, telefoonboeken, handboeken in verschillende talen, bus- en spoorboekjes en plaatjes of kaarten mee te genereren. Deze kunnen dan worden samengesteld aan de hand van data die in één of meerdere databases zijn opgeslagen.



In de bovenstaande figuur is de invoer en uitvoer van BookDB-XML te zien. Als invoer gebruikt BookDB-XML één of meerdere databases, een door de gebruiker opgestelde XML template en eventueel één of meerdere XSLT's om voor de opmaak te zorgen. Als uitvoer geeft BookDB-XML valide XML-bestanden, zoals teksten, tabellen en XML/SVG plaatjes.

De gebruiker van BookDB-XML kan zowel een klant als een medewerker van Sabern BV zijn. Het gebruik van BookDB-XML is niet altijd makkelijk voor de gebruiker, omdat de gebruiker zelf een template moet maken. Om dit goed te doen is kennis vereist van XML en SQL. Aangezien sommige klanten van Sabern BV deze kennis niet bezitten, wordt het maken van een eindprodukt in sommige gevallen aan Sabern BV zelf overgelaten. Ook worden er cursussen gegeven over het gebruik van BookDB-XML.

5.2 De opties

Zodra de gebruiker BookDB-XML opstart, verschijnt er een grafische user-interface met een aantal opties. Zo kan de gebruiker de template selecteren welke gebruikt dient te worden. Ook kan de bestandsnaam van het eindresultaat opgegeven worden. Indien de gebruiker het wenst, kan het eindresultaat worden opgesplitst in meerdere bestanden. Additionele XSLT's kunnen worden toegevoegd. Een XSLT is een soort uitbreiding van de template, je kan er je eigen elementen in definiëren en wat er moet gebeuren als zo'n element in de template voorkomt. Ook kunnen er parameters worden meegegeven welke gebruikt kunnen worden om bepaalde waarden mee te geven aan de additionele XSLT's. In dit verslag wordt een XSLT ook wel als een XSL aangeduid.

5.3 Het proces van het maken van een eindprodukt

Het maken van een eindprodukt gebeurt in twee stappen. De eerste wordt uitgevoerd met behulp van BookDB-XML, de tweede met een opmaakprogramma zoals Adobe FrameMaker.

5.3.1 Het genereren van een XML-bestand

Het opstellen van een template

BookDB-XML is (nog) niet erg gebruiksvriendelijk, omdat de gebruiker zelf moet programmeren om het gewenste resultaat te krijgen. Dit gebeurt door middel van het opstellen en

uitvoeren van templates. Een template is een XML-document waarin precies staat gedefinieerd hoe het eindresultaat moet worden opgebouwd (de structuur), de SQL-queries om data uit de database te halen en welke databases waarop deze queries uitgevoerd dienen te worden. Met behulp van verschillende soorten loops kan een bepaalde query herhaald worden, eventueel met andere parameters. Het maken van een template geschiedt in een willekeurige text-editor. Een voorbeeld van een template is te vinden in paragraaf 5.5.

Het executeren van de template
Zodra de template klaar is, kan deze worden geëxecuteerd. Er zijn vaste XSLT-bestanden die BookDB-XML gebruikt. Deze bestanden worden gebruikt om aan de hand van de template het gewenste resultaat te genereren. In deze bestanden staan de elementen gedefinieerd die in de templates voor mogen komen. Hierbij staan ook de mee te geven parameters (in de vorm van attributen) welke meegegeven dienen te worden en de methode die uitgevoerd dient te worden zodra het element voorkomt. Zodra de template geëxecuteerd wordt, zal het element herkend worden door Apache Xalan en zal de bijbehorende methode worden aanroepen in de klasse BookDB.java. Deze methoden kunnen bijvoorbeeld een connectie met de database maken of een SQL-query uitvoeren en de resultaten gestructureerd in het eindresultaat plaatsen. Ook is het mogelijk om bepaalde elementen te verwijderen. Het eindresultaat is een valide XML-bestand waarin gestructureerd alle data staat.

5.3.2 Het maken van de opmaak

Met behulp van Adobe FrameMaker kan het eindresultaat (het valide XML-bestand dat door BookDB-XML is gegenereerd) worden omgevormd tot het gewenste produkt. Zodra er bijvoorbeeld een nieuwe versie van hetzelfde produkt gegenereerd dient te worden (zoals bijvoorbeeld een nieuwe jaarcatalogus), kan dezelfde template worden uitgevoerd op de database. Het XML-bestand wordt gegenereerd door BookDB-XML en Adobe FrameMaker zal het XML-bestand herkennen en er een catalogus van maken. Deze ziet er dan hetzelfde uit als de vorige, maar nu met de huidige waarden uit de database.

5.4 De architectuur van BookDB-XML

BookDB-XML bestaat uit de volgende onderdelen:

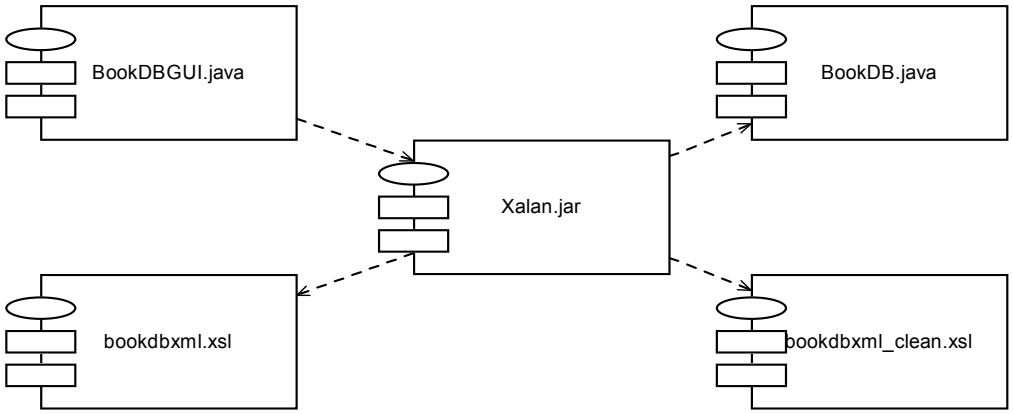
- De klasse BookDBGUI
  Deze klasse initieert BookDB-XML. Het zorgt voor het gedeelte van BookDB-XML dat voor de gebruiker zichtbaar is: de GUI. Er staat ook precies in wat er moet gebeuren als de gebruiker een actie uitvoert in de GUI. Van deze klasse wordt één instantie gemaakt.

- De Java-library Xalan
  Deze library bevat een processor die XML-bestanden kan transformeren in andere XML-bestanden, HTML of andere tekst-bestanden. Deze library is ontwikkeld door de Apache groep.

- De klasse BookDB
  In deze klasse staan de methoden die uitgevoerd worden als er in de XML template een bepaalde functie voorkomt.

- Standaard XSLT's
  In deze XSL-bestanden staan de functies die gebruikt kunnen worden in BookDB-XML gedefinieerd en de bijbehorende methode(s) die hierbij horen. De XSL-bestanden zijn dus eigenlijk de link tussen de template en BookDB-XML. Zodra er een bepaalde functie wordt aangeroepen in de template, zal Apache Xalan kijken of deze functie staat gedefinieerd in een van de XSL-bestanden. Als dit het geval is, zal Apache Xalan de bijbehorende methode uitvoeren in BookDB.java en hieraan als parameter het gehele element dat in de template staat, meegeven. De methode in BookDB.java kan vervolgens berekeningen uitvoeren op deze data. Dit betreft onder andere het uitvoeren van SQL-statements. Indien de functie niet gedefinieerd staat in de XSL-bestanden, zullen deze worden overgenomen in het eindresultaat.

In het onderstaande componentdiagram zijn de belangrijkste componenten van BookDB-XML en hun onderlinge relaties weergegeven.



Zodra BookDB-XML wordt opgestart, wordt er één instantie gestart van de klasse BookDBGUI. Deze roept Xalan.jar aan voor het transformeren van de tempate. Het reeds bestaande binaire component Xalan gebruik op zijn beurt de klasse BookDB en de bestanden bookdbxml.xsl en bookdbxml_clean.xsl. Alle componenten zijn in principe afhankelijk van Xalan.jar, maar kunnen allemaal (behalve Xalan) los gecompileerd of aangepast worden.

5.5 Voorbeeld template

Om duidelijk te maken hoe een template is opgebouwd, volgt hier een klein voorbeeld:

```
<?xml version="1.0" encoding="UTF-8"?>
<LandenEnPlaatsen>
<DBINFO>
    <dbdriver>sun.jdbc.odbc.JdbcOdbcDriver</dbdriver>
    <dburl>jdbc:odbc:Cursus</dburl>
    <dbname>Cursus</dbname>
    <user/>
    <password/>
</DBINFO>
<SQLLoop resultset="Landgegevens" query="SELECT * FROM Landen ORDER BY ID">
  <Land>
    <Landnaam><SQLField resultset="Landgegevens" name="Landnaam"/></Landnaam>
      <Plaatsen>
        <SQLLoop resultset="Plaatsen">
        <SQLLoopQuery>SELECT * FROM Plaatsen WHERE LandID =
              <SQLField resultset="Landgegevens" name="ID"/> ORDER BY ID</SQLLoopQuery>
          <Plaats>
            <Plaatsnaam><SQLField resultset="Plaatsen" name="Plaatsnaam"/></Plaatsnaam>
            <Plaatsinfo><SQLField resultset="Plaatsen" name="Info"/></Plaatsinfo>
          </Plaats>
        </SQLLoop>
      </Plaatsen>
  </Land>
</SQLLoop>
</LandenEnPlaatsen>
```

Zodra deze template is uitgevoerd zal het resultaat een valide XML-document zijn. Hierin is voor elk land in de tabel “Landen”, de landnaam plus de plaatsnaam en plaatsinformatie die bij het land horen, weergegeven. De uitkomst hiervan is in het volgende hoofdstuk weergegeven. De elementen <Land>, <Landnaam>, <Plaatsen>, <Plaats>, <Plaatsnaam> en <Plaatsinfo> zijn hier gedefinieerd om het resultaat zo gestructureerd mogelijk te maken. Er wordt hierbij gebruik gemaakt van de database “Cursus”. De link met deze database staat gedefinieerd in het DBINFO-element.

6. Extensible Markup Language

In dit hoofdstuk is een beschrijving gegeven van eXtensible Markup Language, ofwel XML. Omdat zowel de templates als de uitvoer-bestanden van BookDB-XML in XML-formaat zijn, wijd ik hier een hoofdstuk aan om wat achtergrondinformatie te geven over XML.

6.1 Wat is XML?

XML is een universele opmaaktaal (ontwikkeld door het Wold Wide Web Consortium) dat gebruikt kan worden om data gestructureerd te definiëren en te beschrijven. Het stelt ontwikkelaars in staat zelf tags te specificeren. Hiermee wordt het mogelijk zelf definities te schrijven en protocollen en talen te ontwikkelen om communicatie tussen programma's organisaties mee te realiseren.

6.2 Het gebruik van XML in BookDB-XML

BookDB-XML maakt gebruik van XML voor de input (template) en output (eindresultaat). XSLT wordt gebruikt om deze XML documenten (in dit geval de templates) om te zetten met behulp van Apache Xalan naar het eindresultaat. XSLT is een in XML ontworpen taal dat gebruikt wordt om XML-bestanden om te zetten naar XHTML, HTML of andere XML-bestanden.

6.3 Hoe werkt XML?

Een XML-bestand is een tekstbestand. Dit bestand zelf doet niets, je kunt het dus niet uitvoeren. Aan het begin van een XML-bestand staat altijd de declaratie. Hieraan kan worden herkend dat het een XML-bestand betreft. Hierna volgt het verplichte root-element, dat alle informatie bevat. De informatie staat tussen tags waarvan de gebruiker vrij is om deze zelf de definiëren. Een element genaamd “Persoon” wordt geopend met <Persoon>, gevolgd door de informatie, weergegeven als tekst. Ook binnen dit element mogen andere elementen gedefinieerd worden. Vervolgens moet de tag weer gesloten worden met </Persoon>. Een tag kan ook attributen hebben om een eigenschap weer te geven: <Persoon naam=”Christian”>. Ook bij attributen is de gebruiker vrij deze zelf te definiëren. Toch gaat de voorkeur uit naar het gebruik van tags en niet van attributen, omdat met het gebruik van attributen de essentie van XML wordt vermeden. Tags kunnen net zo diep genest worden als gewenst. Hiermee kan de data zo gestructureerd mogelijk worden opgeslagen.

6.4 Een klein voorbeeld

Om een indruk te geven van hoe een XML-bestand er uit zou kunnen zien, heb ik hier een voorbeeld opgenomen. Dit voorbeeld geeft de uitkomst weer van de template uit Hoofdstuk 5. Hier is goed de structuur van het XML-document te zien. Omdat alle data netjes geordend is, kan er in het opmaakprogramma (bijvoorbeeld FrameMaker) precies opgegeven worden wat er met welk gedeelte van de data moet gebeuren qua opmaak.

```
<?xml version="1.0" encoding="UTF-8"?>
<LandenEnPlaatsen>
  <Land>
    <Landnaam>Nederland</Landnaam>
    <Plaatsen>
      <Plaats>
        <Plaatsnaam>Amsterdam</Plaatsnaam>
        <Plaatsinfo>Hoofdstad</Plaatsinfo>
      </Plaats>
      <Plaats>
        <Plaatsnaam>Rotterdam</Plaatsnaam>
        <Plaatsinfo>Arbeidersstad aan de maas</Plaatsinfo>
      </Plaats>
      <Plaats>
        <Plaatsnaam>Leiderdorp</Plaatsnaam>
        <Plaatsinfo>Vestigingsplaats van Sabern</Plaatsinfo>
      </Plaats>
    </Plaatsen>
  </Land>
```

```
  <Land>
    <Landnaam>Belgie</Landnaam>
    <Plaatsen>
      <Plaats>
        <Plaatsnaam>Antwerpen</Plaatsnaam>
        <Plaatsinfo>Veel bezocht door Nederlanders</Plaatsinfo>
      </Plaats>
      <Plaats>
        <Plaatsnaam>Brussel</Plaatsnaam>
        <Plaatsinfo>Hoofdstad, Europese bolwerk</Plaatsinfo>
      </Plaats>
    </Plaatsen>
  </Land>
  <Land>
    <Landnaam>Duitsland</Landnaam>
    <Plaatsen>
      <Plaats>
        <Plaatsnaam>Dusseldorf</Plaatsnaam>
        <Plaatsinfo>Vertrekpunt voor de Malediven</Plaatsinfo>
      </Plaats>
      <Plaats>
        <Plaatsnaam>Bonn</Plaatsnaam>
        <Plaatsinfo>Hoofdstad</Plaatsinfo>
      </Plaats>
      <Plaats>
        <Plaatsnaam>Berlijn</Plaatsnaam>
        <Plaatsinfo>Ooit in Oost en West verdeeld door een muur</Plaatsinfo>
      </Plaats>
    </Plaatsen>
  </Land>
</LandenEnPlaatsen>
```

Alle informatie over XML en XSLT heb ik vandaan bij http://www.w3schools.com (zie Bijlage B - Literatuurlijst / Bronvermelding).

7. Het proces

In dit hoofdstuk zijn de activiteiten beschreven die voor het realiseren van de afstudeeropdracht zijn uitgevoerd. Dit hoofdstuk begint met een uitleg van de aanpak van het proces. Hierin staat beschreven hoe ik het gehele proces heb ingedeeld en waarom ik dat op die manier heb gedaan. Hierna volgt voor elke hoofdfase van IAD een beschrijving van hoe deze zijn ingedeeld en zijn verlopen.

7.1 De aanpak van het proces

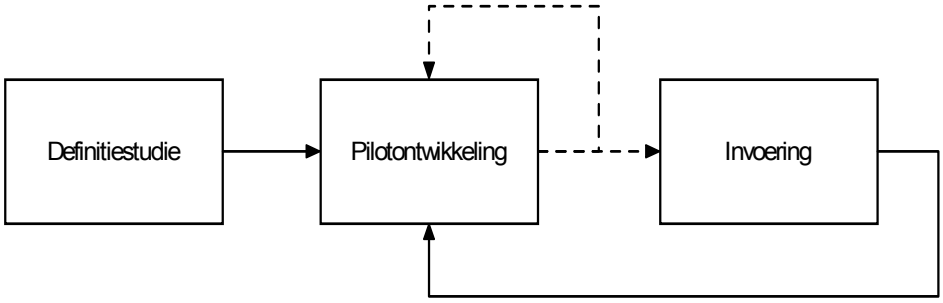
Mijn afstudeeropdracht bestond globaal gezien uit twee delen:

- Verplichte functionaliteiten
- Optionele functionaliteiten

Aan het begin van de afstudeerperiode waren alleen de verplichte functionaliteiten bekend. Dit waren punten die in ieder geval tijdens mijn afstudeerperiode moesten kunnen worden gerealiseerd. Mocht ik daarna eventueel nog tijd hebben, dan werd er pas beslist wat de overige functionaliteiten zouden zijn. Dit is de reden waarom ik aan het begin van mijn afstudeerperiode alle projecttijd heb ingedeeld voor deze verplichte functionaliteiten, deze moesten immers gerealiseerd worden binnen de projecttijd van 15 weken. Omdat de systeemeisen van deze functionaliteiten al waren vastgesteld, hoefde de definitiestudie maar éénmalig te worden doorlopen. Vervolgens kon dan voor elke pilot apart een pilotontwikkeling worden doorlopen. Mocht de projectleider hierna de pilot gelijk willen gebruiken, dan kon er eventueel een invoering plaatsvinden. Deze ontwikkelingsstrategie is een combinatie van “incrementeel ontwikkelen” en “incrementeel opleveren”.

Omdat aan het begin van de afstudeerperiode nog niet duidelijk was wat de optionele functionaliteiten zouden inhouden, wist ik dat als ik klaar zou zijn met de verplichte functionaliteiten weer éénmalig de definitiestudie doorlopen zou moeten worden. Hierna zou weer voor elke pilot een pilotontwikkeling worden doorlopen en indien nodig een invoering.

Het verloop van het afstudeerproces kan als volgt schematisch weergegeven worden:



Voor zowel de verplichte als optionele functionaliteiten gold het volgende:
Eénmalig de definitiestudie, gevolgd door voor elke pilot een pilotontwikkeling. Indien de projectleider een invoering wenst, zal die na de pilotontwikkeling worden uitgevoerd. Zo niet, dan werd er doorgegaan met de volgende pilotontwikkeling.

7.2 Definitiestudies

Het doel van de definitiestudie is de doelen van het systeem te analyseren.
Tijdens mijn afstudeerproject heb ik twee keer een definitiestudie uitgevoerd, een keer voor de verplichte functionaliteiten en een keer voor de optionele functionaliteiten. Hieronder volgt een uitleg van de activiteiten die ik heb doorlopen en de produkten die ik tijdens deze definitiestudies heb opgeleverd.

7.2.1 Plan van aanpak

Aan het begin van de eerste definitiestudie heb ik een plan van aanpak opgesteld. Het plan van aanpak biedt houvast tijdens het project. Hierin heb ik ook gelijk het ontwikkelscenario beschreven zoals ik vond dat het project doorlopen moest worden. Het plan van aanpak was dynamisch, dat wil zeggen dat ik het steeds heb aangepast wanneer dit nodig was bij de aanvang van elke hoofdfase, dus ook bij de tweede definitiestudie. Volgens IAD is dit ook nodig. De laatste versie van dit plan van aanpak is te vinden in Bijlage D.

7.2.2 Bekend raken met BookDB-XML

Voordat ik aan BookDB-XML zou gaan werken, leek het de opdrachtgever en mij erg verstandig om eerst wat ervaring op te doen met het pakket. Het idee was om zelf het gehele traject een keer te doorlopen waarbij met behulp van een zelf geschreven template bepaalde data uit de database wordt gehaald en omgezet in een XML-bestand. Ik heb hierbij gebruik gemaakt van de gegevens van een klant van Sabern BV. Met het maken van een template en daarmee de benodigde gegevens uit de database halen met behulp van BookDB-XML had ik geen moeite. Het resultaat was een valide XML-bestand met daarin de waarden uit de database gestructureerd weergegeven.

Tijdens de hiervoor ingeplande drie weken heb ik ook geprobeerd voor mezelf duidelijk te maken hoe BookDB-XML precies werkt onder de motorkap. Dit was vrij lastig, omdat er niets over de ontwikkeling van BookDB-XML was gedocumenteerd. Er was alleen een incomplete gebruikershandleiding en documentatie in de source-code beschikbaar. Om duidelijk te krijgen hoe BookDB-XML precies werkt heb ik de source-code doorlopen en voor mezelf op papier globaal in beeld gebracht wat er precies gebeurde binnen de source-code. Het zou te lang duren om alle details van de precieze werking te weten te komen, het ging mij erom om een globale indruk te krijgen van de werking van BookDB-XML. Eventuele details zouden later komen tijdens de pilotontwikkeling.

Dit alles geschiedde tijdens de eerste definitiestudie.

7.2.3 Ontwikkelscenario-workshop

De ontwikkelscenario-workshop bestond uit een kort gesprek met de projectleider. Tijdens deze workshop hebben wij het ontwikkelscenario doorgelopen. De projectleider ging akkoord met de manier waarop ik het project aan wilde pakken. Ik heb tijdens dit gesprek nog enkele onduidelijkheden opgehelderd, zoals of er na elke pilotontwikkeling ook daadwerkelijk een invoering plaats zou vinden. Na het gesprek heb ik, zoals vereist volgens IAD, het plan van aanpak aangepast.

Tijdens de tweede definitiestudie heb ik geen ontwikkelscenarioworkshop gehouden, omdat dezelfde ontwikkelingsstrategie zou worden gebruikt.

7.2.4 Pilotplan-workshop

De pilotplan-workshop, welke direct na de ontwikkelscenario-workshop volgde, bestond uit een drie uur durend gesprek met de projectleider. Tijdens dit gesprek hebben we het volgende gedaan:

- De door Sabern vastgestelde systeemeisen één voor één nauwkeurig doorgenomen
- Indeling aantal pilots gemaakt met bijbehorende prioriteiten, deadlines en planning

Ik heb ervoor gekozen alle systeemeisen (die de opdrachtgever van te voren al had opgesteld) in één keer te bespreken, omdat dat het maken van een gedetailleerde planning en indeling van het aantal pilots makkelijker zou maken omdat er dan meer duidelijkheid zou zijn. Niet alle systeemeisen waren voor mij in één keer duidelijk, omdat ik nog geen kennis had van de precieze werking van BookDB-XML onder de motorkap. Mijn verwachting was dat ik, naarmate ik meer met het project bezig was, meer inzicht zou krijgen in de werking van het systeem.

Tijdens de tweede definitiestudie werd nogmaals een pilotplan-workshop gehouden om de optionele functionaliteiten allemaal een één keer te bespreken.

Na de pilotplan-workshop heb ik, zoals vereist volgens IAD, de volgende produkten opgeleverd:

Systeemeisen
Na de pilotplan-workshop heb ik de systeemeisen in eigen woorden weergegeven in een lijst met systeemeisen. Deze lijst met systeemeisen is te vinden in Bijlage D. De systeemeisen zijn in dit document al geprioriteerd en ingedeeld in pilots. Het belang van de systeemeisen is dat deze kunnen worden gebruikt om de pilots mee te toetsen.
De projectleider heeft deze lijst met systeemeisen goedgekeurd.

Tijdens de tweede definitiestudie heb ik de lijst met systeemeisen aangevuld met de systeemeisen behorend bij de overige functionaliteiten.

Systeemconcept
Op basis van de lijst met systeemeisen heb ik een systeemconcept opgesteld waarin te zien is hoe de functionaliteiten moesten gaan werken. Dit heb ik beschreven vanuit het oogpunt van de gebruiker. De functionaliteiten zijn in het systeemconcept in volgorde van prioriteit weergegeven. Het belang van het systeemconcept is dat het alle betrokkenen van het project (inclusief mijzelf) inzicht geeft in de werking van het te realiseren systeem. Tevens kan je eraan meten of de gerealiseerde pilots ook daadwerklijk wel aan het systeemconcept voldoen. Het systeemconcept is goedgekeurd door de projectleider en is terug te vinden in Bijlage D. Omdat het systeemconcept al een beoogde oplossing weergeeft van de te ontwerpen en implementeren functionaliteit, kan het beschouwd worden als een analyse.

Tijdens de tweede definitiestudie heb ik het systeemconcept bij de aanvang van elke afzonderlijke pilotontwikkelingsfase aangevuld.

Pilotplan
Na het maken van het systeemconcept heb ik een pilotplan gemaakt waarin per pilot wordt weergegeven op welk deel van het systeemconcept het betrekking heeft. Ook is in dit document de geschatte tijdsindeling weergegeven die ik met de projectleider heb afgesproken. Ook eventuele risico's komen hier aan bod. Het pilotplan is te vinden in Bijlage D.

Omdat de definitiestudie voor het eerste gedeelte van het project slechts éénmaal doorlopen werd, en het pilotplan na elke iteratie soms toch aangepast diende te worden, heb ik besloten om steeds aan het begin van elke iteratie het pilotplan te toetsen en eventueel te herzien. Dit is toegestaan bij IAD. Het pilotplan fungeert als leidraad van de ontwikkeling van BookDB-XML.

Aan het eind van de definitiestudies werden de verplichte functionaliteiten als volgt verdeeld over de volgende pilots in de volgende mijlpaalprodukten:

Eerste definitiestudie:
- Pilot Ia - Variabelen bij naam benoemen
- Pilot Ib - Database-selectie SQLSingleResult en SQLAggregatedResult

- Pilot II - Alternatieve oplossing SQLLoops
- Pilot IIIa - (Fout)meldingen beter afvangen en tonen
- Pilot IIIb - Seriële executie van taken
- Pilot IV - Mechanisme voor algemene loop
- Pilot V - Mechanisme voor headers en footers voor gebalanceerde tabellen

Tweede definitiestudie:
- Pilot VI - Workaround/Oplossing voor Java-bug
- Pilot VII - IF/THEN/ELSE-statement voor in template
- Pilot VIII - Grafical User Interface voor genereren van templates

De probleembeschrijving wordt door de volgende pilots gedekt:

- **Finetunen van bestaande functionaliteiten:**
  Pilot Ia, Ib, II, IIIa, V, VI

- **Ontwerpen en implementeren van algemene nieuwe functionaliteiten. Denk hierbij aan seriële executie van taken en aan de mogelijkheid om in een template iteratie en selectie op te nemen.**
  Pilot IIIb (Seriële executie van taken)
  Pilot IV (Iteratie)
  Pilot VII (Selectie)

- **Het ontwerpen en eventueel implementeren van een grafische user-interface voor het samenstellen van templates:**
  Pilot VIII

7.3 Pilotontwikkelingen

Voor elke pilot werd éénmalig de pilotontwikkelingsfase doorlopen. Aan het begin van deze fase heb ik een pilotontwikkelplan opgesteld. Hierin staat beschreven hoe de pilot zou worden gebouwd binnen de begrenzingen die hiervoor zijn gesteld met behulp van timeboxes. Hiervoor zijn de pilots (waar mogelijk) ingedeeld in zo klein mogelijke eenheden die geprioriteerd zijn neergezet in een tabel. Per eenheid is een geschatte tijd neergezet waarbinnen deze eenheid gerealiseerd diende te worden. Tevens is ook de relevantie van de eenheden weergegeven. Ik heb ervoor gekozen de pilotontwikkelplannen van alle pilots in één document onder te brengen. Hierdoor blijft het aantal documenten beperkt. De pilotontwikkelplannen van alle pilots zijn terug te vinden in Bijlage D. Aan het begin van elke pilotontwikkeling heb ik, indien dit nodig was, de systeemeisen, systeemconcept en pilotplan herzien. Dit is toegestaan bij IAD. Hierna heb ik voor elke pilot of pilotdeel een ontwerp gemaakt waarna het ontwerp werd gerealiseerd. Zoals vereist heb ik na de realisatie van de pilots ook de gebruikersdocumentatie aangepast wanneer dit nodig was. Het testen van de pilotdelen vond grotendeels plaats tijdens de implementatie en erna. Bij de wat grotere pilots zoals Pilot IIIb, IV, VII en VIII heb ik wat uitgebreidere tests uitgevoerd. Steeds wanneer een pilot eerder klaar was dan gepland heb ik de timebox van de volgende pilot naar voren opgeschoven, zodat ik gelijk met deze pilot kon beginnen.
Tijdens de pilotontwikkelingen heb ik de projectleider goed op de hoogte gehouden van de vorderingen. Hierdoor was het niet nodig om een uitgebreide beoordelings- en test-workshop te houden. Het testen deed ik immers zelf.

7.4 Invoeringen

Het was niet mogelijk om van te voren vast te stellen wanneer de invoeringsfase plaats zou vinden. Dit komt omdat de projectleider op onverwachte tijdstippen pas liet weten wanneer hij een pilot of een groep pilots in ontvangst wou nemen. Uiteindelijk vonden er vijf invoeringen plaats, bij welke de volgende pilots per invoering werden opgeleverd:

- Pilot I, II, III
- Pilot IV, V
- Pilot VI
- Pilot VII
- Pilot VIII

De invoeringsfasen bestonden uit twee delen:

Acceptatieworkshop
Tijdens de acceptatieworkshops gaf ik een demonstratie van de opgeleverde pilots aan de projectleider. Deze was hier in alle gevallen positief over, hoewel er in het geval van Pilot VIII nog kleine puntjes waren die hij er graag nog bij wou hebben. Dit betrof per pilot de volgende punten:

- Aanduiding van de precieze fout waar het mis gaat als een template niet valide is aan de gebruiker.
- Het testen van reeds bestaande <DBINFO>-elementen
- Voor elke aangemaakte SQLLoop, moet men met behulp van een menuutje een database kunnen kiezen.
- Zodra een SQLField wordt geselecteerd, moet hierbij ook de juiste bijbehorende database als attribuut worden ingevuld.

Deze punten werden daarna ook zo snel mogelijk aangebracht in de pilot, waarna de acceptatietest opnieuw werd uitgevoerd. Dit is toegestaan bij IAD.

Pilot overdragen
Na de acceptatieworkshops, werden de pilot(s) opgeleverd aan de projectleider. Aangezien de projectleider de opgeleverde pilot ook daadwerkelijk in gebruik nam, was hij ook in staat eventueel feedback te geven over eventuele bugs. Dit kwam alleen voor bij Pilot IIIb. De bugs die hierbij gevonden werden door de projectleider heb ik genoteerd in een buglist, welke te vinden is in Bijlage E. Aangezien ik van de overige pilots geen negatieve kritiek heb ontvangen, ga ik er van uit dat hier tot nu toe nog geen bugs in zijn gevonden.

8. Opgeleverde produkten

In dit hoofdstuk is van elke opgeleverde pilot een uitvoerige beschrijving gegeven.
Hierbij wordt er per pilot inhoudelijk ingegaan waarbij de nadruk ligt op het opgeleverde produkt. Omdat niet alle pilots even interessant zijn, heb ik ervoor gekozen om vier pilots (IIIb, IV, VII en VIII) wat uitvoeriger te bespreken dan de overige pilots. Dit is ook de reden dat beide pilotdelen van Pilot III in aparte paragrafen besproken worden.

8.1 Pilot I

Pilot I bestond uit twee afzonderlijke pilotdelen, welke hieronder afzonderlijk worden beschreven. Beide pilotdelen zijn afzonderlijk van elkaar ontworpen en geïmplementeerd.

8.1.1 Variabelen bij naam benoemen

Voorheen was het voor de gebruiker alleen mogelijk een variabele te kunnen benoemen en deze aan te duiden met een nummer. Dit geschiedde door in de template de volgende elementen te zetten:

- Variable aanmaken: *<SetVar index="X" value="Y"/>*
- Variable oproepen: *<GetVar index="X"/>*

Hierin is X de aanduiding van de variable en Y de variabele zelf. Aangezien X hier alleen uit een nummer kon bestaan, kon de gebruiker dit als verwarrend ervaren. Alle variabelen werden in een Array geplaatst. Het nadeel van een Array is dat de variabelen alleen maar kunnen worden aangeduid met een nummer. Om het voor de gebruiker mogelijk te maken de variabelen ook aan te duiden met een String, was het nodig om een andere manier van opslag te gebruiken. Ik heb hiervoor gekozen een HashTable te gebruiken, omdat een HashTable wél de mogelijkheid biedt een variabele aan te duiden met een String. Het heeft verder dezelfde eigenschappen als een Array, met als enige verschil dat de index een String is in plaats van een integer. In de onderstaande tabel is dit verschil goed te zien.

Array & HashTable

index (int)	value
1	Aap
2	Noot
3	Mies
4	Wim
5	Zus

index (String)	value
Dier	Aap
Vrucht	Noot
3	Mies
4	Wim
Familielid	Zus

Na de realisatie van deze pilot was het mogelijk voor de gebruiker om tijdens het benoemen van de variable, deze aan te duiden met zowel een naam als een nummer.

8.1.2 Database-selectie SQLSingleResult en SQLAggregatedResult

Tevens was het alleen mogelijk om de functies SQLSingleResult en SQLAggregatedResult toe te passen op de standaard database. Beide functies leveren (indien uitgevoerd) één resultaat op. Beide functies konden als volgt worden aangeroepen:

- Een enkel resultaat: *<SQLSingleResult query="X" field="Y"/>*
- Het aantal resultaten van een query: *<SQLAggregatedResult query="X" field="Y"/>*

Hierin is X de SQL-query die uitgevoerd dient te worden. De uitkomst levert in beide gevallen één resultaat op. In het geval van SQLSingleResult wordt, indien er meer dan uit resultaat is, alleen het eerste resultaat genomen. Y is hier de naam van de kolom van de tabel die opge-

vraagd dient te worden.

Deze functies gebruikten oorspronkelijk de standaard database die gedefinieerd is in het <DBINFO>-element. Indien de gebruiker een andere database wilde gebruiken, kon hij of zij dus geen SQLSingleResult of SQLAggregatedResult gebruiken. In plaats daarvan moest de gebruiker een SQLLoop opstellen en hieraan de naam van de te gebruiken database meegeven en vervolgens de query zo specificeren dat deze hetzelfde resultaat zou geven als die van een SQLSingleResult of SQLAggregatedResult.

Door een extra attribuut toe te voegen aan deze functies, is het voor de gebruiker mogelijk een database naar keuze te selecteren waarop de query moet worden uitgevoerd. Hierdoor was het ook niet meer nodig een SQLLoop om de functie heen te zetten. Een voorwaarde is wel dat de te gebruiken database (“Z”) is een <DBINFO>-element is gedefinieerd. De opbouw ziet er nu als volgt uit:

- *<SQLSingleResult query=”X” field=”Y” usedb=”Z”/>*
- *<SQLAggregatedResult query=”X” field=”Y” usedb=”Z”/>*

Het is overigens niet verplicht om het “usedb”-attribuut mee te geven, maar als deze wordt weggelaten zal gewoon dat standaard database worden gebruikt.

Er waren eigenlijk geen alternatieve mogelijkheden om deze functionaliteiten te realiseren. Het gebruik van een HashTable is simpelweg gewoon de enige en verstandigste oplossing om dit probleem snel en effectief op te lossen. Ditzelfde geldt voor het gebruik van een “usedb”-attribuut.

Tussenconclusie
Het is nu wél mogelijk om variabelen bij naam te kunnen benoemen en om een database naar keuze te selecteren bij het gebruik van een SQLSingleResult of SQLAggregatedResult. Hiermee is voldaan aan de systeemeisen en systeemconcept behorende bij deze pilot.

8.2 Pilot II

Pilot II betrof het optimaliseren van de transformatie van de template, zodat gedeelten van de template niet meerdere keren onterecht zouden worden uitgevoerd.

Oorspronkelijk was het zo dat BookDB-XML de volgende twee XSL’s gebruikte in de volgende volgorde:

- bookdbxml.xsl
  Eerst werd dit bestand net zo lang uitgevoerd met behulp van Apache Xalan totdat alle SQLLoops in de template waren uitgewerkt. Aan dit bestand werden oorspronkelijk ook de additionele XSL’s gekoppeld met daarin definities van elementen die niet standaard bij BookDB-XML zaten.

  Bijvoobeeld:
  In het additionele XSL-bestand “extra.xsl” kon bijvoorbeeld het element <AO/> gedefinieerd staan. Zodra dit element voor zou komen, zou er in het tussenresultaat de tekst “Artikelomschrijving: ” in het element <AO/> komen te staan.

- bookdbxml_clean.xsl
  Tenslotte werd dit bestand éénmalig uitgevoerd met behulp van Apache Xalan om tijdelijke elementen te verwijderen uit het tussenresultaat.

  Het tijdelijke element <AO/> werd bijvoorbeeld weer weggehaald, zodat alleen de tekst “Artikelomschrijving: ” over zou blijven.

Het probleem was dat tijdens het meermalig uitvoeren van bookdbxml.xsl (indien er dus meerdere

geneste SQLLoops voorkwamen in de template) niet alleen steeds de SQLLoops werden uitgewerkt, maar ook de elementen die beschreven staan in de additionele XSL's net zo vaak werden herhaald als de diepte van het geneste SQLLoop waarin het zich bevond. In het onderstaande voorbeeld is dit probleem duidelijk gemaakt.

```
<SQLLoop>
    <SQLLoop>
        <SQLLoop>
            <AO/><SQLField/>
        </SQLLoop>
    </SQLLoop>
</SQLLoop>
```

De XSL bookdbxml.xsl zou in dit geval drie keer worden uitgevoerd, voor elke geneste SQLLoop één keer, beginnend bij de bovenste SQLLoop. Omdat ook de additionele XSL's gelinkt waren aan bookdbxml.xsl, werd ook deze additionele XSL's ("extra.xsl") drie keer uitgevoerd. De XSL bookdbxml_clean.xsl is op dit moment nog niet uitgevoerd, daardoor blijft het <AO/> element dus steeds in het tussenresultaat voorkomen. Hierdoor werd er dus in het eindresultaat drie keer de tekst "Artikelomschrijving: " in het <AO/> element geplaatst. Dit is uiteraard niet de bedoeling. Stel dat de artikelomschrijving uit de database op de plaats van het <SQLField>-element zou komen te staan, zou het resultaat kunnen zijn:

"Artikelomschrijving: Artikelomschrijving: Artikelomschrijving: Een blauw speelgoedautootje"

Om dit probleem op te lossen heb ik de klasse BookDB dusdanig aangepast dat de volgende XSL's in de volgende volgorde worden verwerkt:

- bookdbxml_preprocessor.xsl
  Als eerste wordt dit bestand net zo lang uitgevoerd totdat alle elementen die queries uitvoeren (dus ook SQLLoops) op de database zijn uitgewerkt. Zodra dit is gedaan, zijn alle loops uitgewerkt en is het niet meer nodig bepaalde elementen meerdere malen uit te voeren.

- bookdbxml.xsl
  Dit is een aangepaste versie van de originele bookdbxml.xsl, alleen staan hier nu de elementen in die slechts éénmalig uitgevoerd dienen te worden. Dit bestand wordt daarom ook éénmalig uitgevoerd tijdens het transformatieproces. Aan dit bestand worden ook de additionele XSL's gekoppeld met daarin elementen die niet standaard bij BookDB-XML zitten. (Zoals bijvoorbeeld de additionele XSL "extra.xsl" in het genoemde voorbeeld.)

- bookdbxml_clean.xsl
  Dit bestand wordt éénmalig uitgevoerd om tijdelijke elementen te verwijderen uit het tussenresultaat. (Bijvoorbeeld het <AO/> element.)

Hierdoor worden dus eerst alle loops uitgewerkt, waarna de additionele XSL's éénmalig worden verwerkt. Tenslotte worden alle tijdelijke elementen verwijderd.

Een alternatieve manier om dit probleem op te lossen was het recursief behandelen van SQLLoops. Hierbij zouden alle SQLLoops inclusief de geneste SQLLoops in één stap worden uitgewerkt. Hiervoor zou dus een methode in de klasse BookDB moeten worden aangemaakt die zichzelf steeds aanroept om zo de SQLLoops uit te werken. Dit is nadelig, omdat daarvoor de source-code voor een groot deel zou moeten worden aangepast. Hierbij zou de originele bookdbxml.xsl dus maar eenmalig verwerkt hoeven worden. Bovendien zouden dan alleen de SQLLoops recursief behandeld worden en niet de eventueel toekomstige elementen die meerdere keren herhaald zouden moeten worden, want dan zou daar ook steeds een recursieve methode voor moeten worden gemaakt.

De manier waarop ik dit probleem heb opgelost is de simpelste en effectiefste manier, omdat in de source-code alleen de afhandeling van de XSL's gewijzigd diende te worden. De rest kon in

de XSL's zelf geregeld worden door hier de elementen in te definiëren die meermalig en éénmalig behandeld dienden te worden. Ook voor eventuele toekomstige elementen zou dit handig kunnen zijn, zoals bij Pilot IV.

De ontwerpen van deze pilot zijn te vinden in Bijlage D. Omdat deze pilot het veranderen van een proces betrof, heb ik in de ontwerpen gebruik gemaakt van flowcharts.

Tussenconclusie
Omdat BookDB-XML gedeelten van de template niet meer meerdere keren onterecht uitvoert op plaatsen waar dat niet nodig is, is voldaan aan de systeemeis en het systeemconcept behorende bij deze pilot.

8.3 Pilot IIIa

Deze pilot bestond uit twee afzonderlijke pilotdelen, welke afzonderlijk van elkaar zijn ontworpen en geïmplementeerd:

- Het moest mogelijk worden om foutmeldingen op een goede manier op te vangen en aan de gebruiker te tonen. (Pilot IIIa)
- Het moest mogelijk zijn voor de gebruiker om meerdere templates serieel uit te voeren. (Pilot IIIb)

Om tot een goed ontwerp te komen, ben ik op zoek gegaan naar eventuele oplossingen. Ik kwam tot de conclusie dat een apart log-tabblad met daarin een tekst-veldje waarin tekst kan worden weergegeven de beste manier was om een de gebruiker op de hoogte te houden van eventuele meldingen. Tevens was het mogelijk om in dit log-tabblad een log te laten zien van het gehele proces. Een ander goed alternatief was er simpelweg niet. Een aparte klasse LogKeeper (welke tevens een Thread is) zou er dan voor zorgen dat (fout)meldingen die voor zouden komen in alle Java-klassen van BookDB-XML in het log-tabblad konden worden bijgeschreven. Zo zou ook precies te zien zijn waar en in welke fase van het transformatieproces de fout optreedt.

Hierbij kwam ik echter een probleem tegen: indien er een foutmelding op zou treden in de klasse BookDB, kon in geen geval de foutmelding naar de klasse LogKeeper worden gestuurd. Omdat het voorgecompileerde component Xalan (een JAR-bestand gemaakt door de Apache groep) de klasse BookDB initieert, herkent deze instantie van BookDB niet de instantie van LogKeeper, welke door de klasse BookDBGUI geïnitieerd wordt.

Hiervoor heb ik een (ietswat ruwe) oplossing gevonden: Zodra er in de klasse BookDB een (fout)melding plaats zou vinden, zou deze worden weggeschreven in een apart tekst-bestandje (log.txt). Hiervoor heb ik binnen de klasse BookDB een aparte methode gemaakt waaraan de foutmelding kan worden meegegeven. De klasse LogKeeper zou dit bestand vervolgens lezen en de melding uit het tekst-bestand in het log-tabblad zetten.

De klasse LogKeeper heeft onder andere de volgende belangrijke methodes:

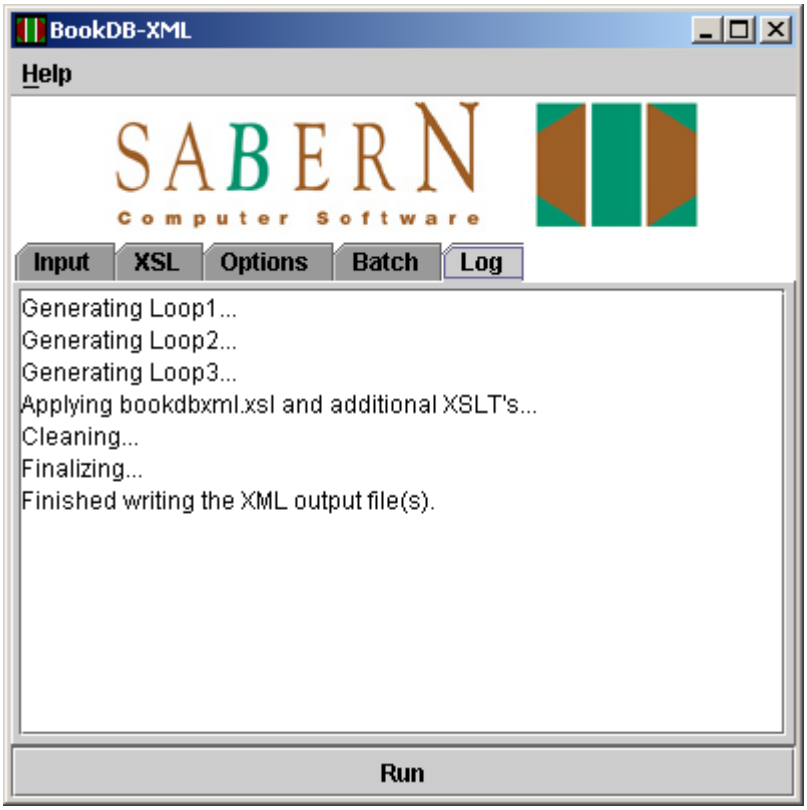
- readLog()
  Deze methode leest het bestand log.txt waarin de (fout)meldingen staan die zijn ontstaan in de klasse BookDB en toont deze meldingen in het log-tabblad.
- logMe(String message)
  Door deze methode aan te roepen en een (fout)melding mee te geven, wordt deze melding in het log-tabblad geplaatst.
- cleanMe()
  Door deze methode aan te roepen wordt het tijdelijke tekst-bestand leeggemaakt.

Mogen er in de toekomst nog meer klassen bijkomen, dan kunnen foutmeldingen die daarin voorkomen, ook weer weggeschreven worden naar log.txt.

Het aanpassen van Xalan om dit probleem te verhelpen leek mij geen goed idee, omdat dit al een

kant-en-klaar bestand is dat niet door Sabern is ontwikkeld. Omdat de meeste foutmeldingen bestonden uit Java-exceptions, was het lastig om een goede omschrijving te vinden van de foutmelding waar de gebruiker iets mee kan. Zo'n exception kan meerdere oorzaken hebben, daarom kon alleen een globale oplossing worden gegeven indien zo'n exception op zou treden. De gedetailleerde foutmelding, gegenereerd door Java zelf, werd ook in het logtabblad in beeld gebracht, zodat de gebruiker toch kan zien waar het precies fout gaat. Naast de foutmeldingen worden ook meldingen in het logtabblad weergegeven die de gebruiker op de hoogte houden van de voortgang van het proces van het genereren van het XML-bestand. Hierbij kan worden gedacht aan de melding dat het genereren van het XML-bestand is gelukt of de melding met welke loop BookDB-XML op het moment bezig is. Al deze meldingen worden "real time" en op chronologische volgorde in het log-tabblad geplaatst.

In de onderstaande figuur is een screenshot te zien van het Log-tabblad met daarin de meldingen van een met succes uitgevoerde template.



Dit log-tabblad springt automatisch op de voorgrond zodra de gebruiker een template executeert. De gebruiker wordt hier op de hoogte gehouden van de status van de transformatie en eventuele fouten.

Het ontwerp van dit pilotdeel is tevens te vinden in Bijlage D.

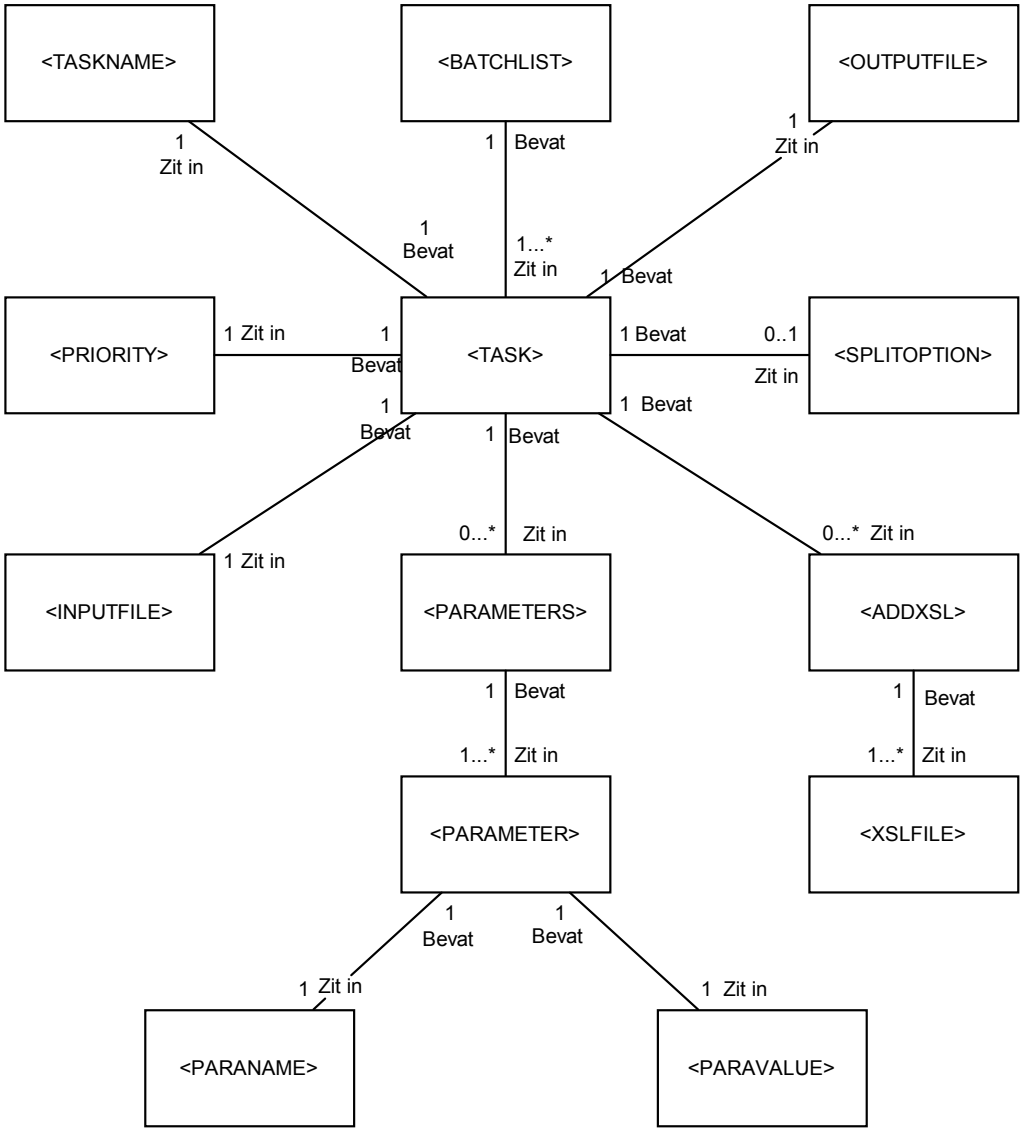
Tussenconclusie
Omdat de gebruiker nu op een gebruiksvriendelijke manier op de hoogte gebracht van foutmeldingen, voldoet deze pilot aan de gestelde systeemeisen en het systeemconcept. Een bijkomstigheid is dat de gebruiker ook op de hoogste wordt gesteld van het verloop van het transformatieproces.

8.4 Pilot IIIb

In deze paragraaf staat het tweede pilotdeel van Pilot III beschreven. Dit betrof de mogelijkheid voor de gebruiker om meerdere templates serieel uit te laten voeren.

8.4.1 Ontwerp

Omdat het mogelijk moest zijn voor de gebruiker om een batchlist op te slaan, leek het mij het verstandigst om de mogelijkheid te bieden om deze op te slaan als XML-bestand. Hierin kon immers alle informatie van de batchlist op een makkelijke en gestructureerde manier worden opgeslagen. Java biedt tevens de mogelijkheid om makkelijk XML-bestanden te genereren en manipuleren. Het gebruik van XML was dus voor de hand liggend. Ik heb in het ontwerp een schema gemaakt waarin staat weergegeven uit welke elementen een Batchlist moet bestaan. Dit schema heeft de eigenschappen van een klassendiagram, maar is het niet, omdat de “klassen” hier simpelweg bestaan uit elementen en geen attributen en operaties hebben. Dit schema is hieronder te zien. Ik had ook een DTD (Document Type Definition) kunnen maken om dit te verduidelijken, maar dit is niet overzichtelijk voor de lezer, omdat een DTD een XML-bestand is. Het schema is hieronder te zien.



Zoals elk XML-bestand, is het vereist dat alle informatie zich bevindt in één root-element, in dit geval genaamd “batchlist”. Aangezien de gebruiker zelf mag beslissen hoeveel tasks hij of zij wil toevoegen aan de batchlist (minimaal één), is het aantal “tasks” ongelimiteerd. Een “task” bestaat hier uit één template met de bijbehorende eigenschappen (naam, prioriteit, input-file, output-file, parameters, split-optie en additionele XSL-bestanden). De naam, prioriteit, input-file en output-file zijn verplichte eigenschappen en moeten dus worden gespecificeerd. In het schema hebben zij daarom ook een 1 op 1 relatie met het TASK-element. In het “priority”-element komt voor elke task een uniek nummer te staan (vanaf 1) die de prioriteit van de task weergeeft. Dit nummer wordt door BookDB-XML zelf gegenereerd. De overige eigenschappen zijn optioneel, daarom hebben deze een 1 op 0/1 of 1 op 0/* relatie met het TASK-element. Indien de gebruiker ervoor kiest parameters of additionele XSL’s te koppelen aan de task, dan hebben deze elementen verplichte kind-elementen. Als de gebruiker beslist om één of meer parameters toe te voegen, dan moet de parameter op zijn of haar beurt precies één parameternaam en één parameter-waarde hebben. De elementen zelf bevatten tekst. Indien de splitoptie wordt gebruikt, wordt een element “splitoption” aangemaakt met daarin de tekst waarbij het resultaat gesplit zal worden. Een opgeslagen batchlist zou er als volgt uit kunnen zien:

```
<batchlist>
  <task>
    <taskname>Taak1</taskname>
    <priority>1</priority>
    <inputfile>C:\BookDB-XML\templates\Task1.xml</inputfile>
    <outputfile>C:\BookDB-XML\output\Task1.xml</outputfile>
    <splitoption>Paragraaf</splitoption>
  </task>
  <task>
    <taskname>Taak2</taskname>
    <priority>2</priority>
    <inputfile>C:\BookDB-XML\templates\Task2.xml</inputfile>
    <outputfile>C:\BookDB-XML\output\Task2.xml</outputfile>
    <parameters>
      <parameter>
        <paraname>Para1</paraname>
        <paravalue>A</paravalue>
      </parameter>
      <parameter>
        <paraname>Para2</paraname>
        <paravalue>B</paravalue>
      </parameter>
    </parameters>
    <addxsl>
      <xslfile>C:\BookDB-XML\XSLT\extra.xsl</xslfile>
      <xslfile>C:\BookDB-XML\XSLT\plus.xsl</xslfile>
    </addxsl>
  </task>
</batchlist>
```

8.4.2 Implementatie

Omdat deze nieuwe functionaliteit de bestaande GUI uitbreidt met nieuwe componenten, is alle code opgenomen in de klasse BookDBGUI.

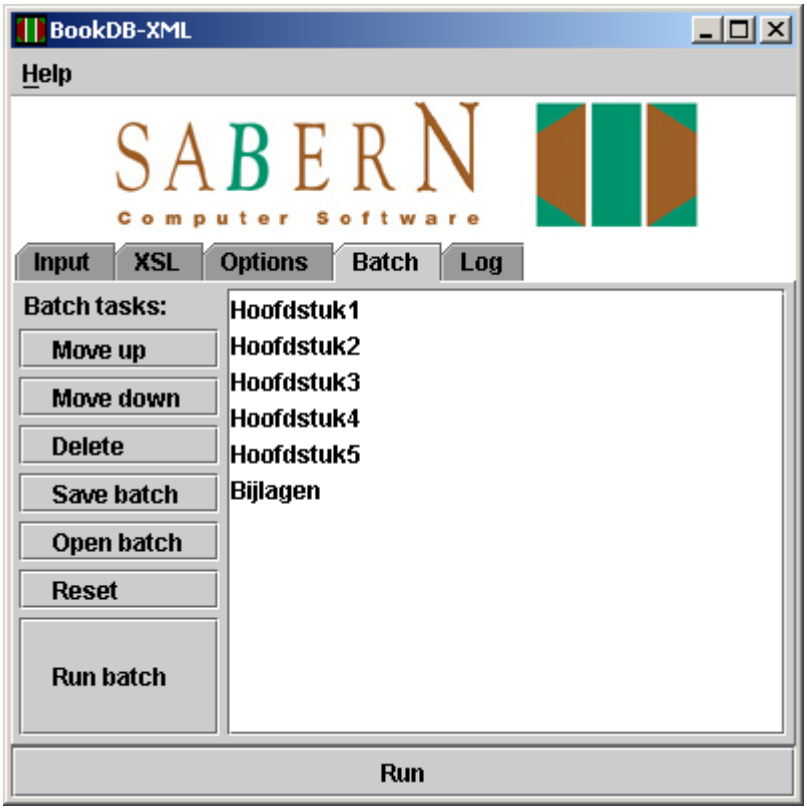
Het maken van source-code achter de “Move up”-, “Move down”- en “Delete”-buttons was vrij lastig, omdat er rekening moest worden gehouden met de prioriteit van de taken. De prioriteit van een taak wordt bepaald door een getal in het XML-bestand. Deze moet uniek zijn, tevens dienen zij opeenvolgend te zijn. Zodra er een taak wordt verwijderd of verplaatst wordt in de lijst, heeft dit invloed op de prioriteit van de overige taken en de taak zelf.

Zodra er een bestaande batchlist wordt geopend of er wordt een nieuwe gegenereerd, dan wordt deze batchlist als XML-bestand (opgebouwd volgens het schema) in het geheugen geplaatst. Zodra een prioriteit verandert, wordt dit direct doorgevoerd in het XML-bestand dat in het geheugen staat. Stel dat de gebruiker een task selecteert en op “Move up” klikt, wordt de prioriteit van deze task één eenheid minder en de task die boven de geselecteerde task stond, één eenheid meer. De prioriteiten worden dus verwisseld. Uiteraard is het niet mogelijk een

task omhoog te verplaatsen die al bovenaan staat of omlaag te verplaatsen die al onderaan staat. Indien een task wordt verwijderd, moet het “gat” wat nu in de prioriteiten zit, worden gedicht. Er wordt eerst gekeken waar het “gat” zit, om vervolgens alle prioriteiten die na het “gat” komen, met één eenheid te verlagen. Stel dat er bijvoorbeeld 6 tasks zijn, waarvan de task met prioriteit 4 wordt verwijderd, onstaat er een gat tussen prioriteit 3 en 5, want prioriteit 4 komt niet meer voor. 1, 2, 3, 5, 6 wordt dus 1, 2, 3, 4, 5.

Het log-tabblad werkte goed samen met deze functionaliteit. Als er een batch werd uitgevoerd, werd van alle tasks het transformatieproces (met eventuele foutmeldingen) weergegeven in het log-tabblad.

In de onderstaande figuur is een screenshot te zien van het Batch-tabblad. Hierin is te zien dat de batch bestaat uit zes aparte taken die serieel uitgevoerd gaan worden. Tevens zijn de opties te zien waar de gebruiker toegang toe heeft.



8.4.3 Testen

Het testen van deze functionaliteit vond grotendeels plaats tijdens het bouwen ervan. Tijdens deze tests kwamen kleine details naar boven waar ik een oplossing voor moest bedenken, zoals bijvoorbeeld het feit dat de “move up”-button niet zou mogen werken op de bovenste taak in de batchlist en de “move down”-button niet op de onderste. Door deze kleine “gaatjes” te dichten, kreeg het systeem een steeds stabieler karakter; de gebruiker kon immers steeds minder fout doen. Zaken zoals bijvoorbeeld het openen en opslaan van een batchlist werkte goed. Omdat bij deze functionaliteit (in tegenstelling tot de voorgaande pilots) de gebruiker erg veel mogelijkheden heeft, is de kans groot dat er toch ergens een fout optreedt. Het leek mij daarom verstandig uitgebreidere tests uit te voeren dan voorheen. Om er zeker van te zijn of de executie van de batchlist correct verliep, heb ik een uitgebreide test-batchlist gemaakt met verschillende tasks en verschillende bijbehorende opties. Dit is te zien in de volgende tabel.

Testresultaten batchlist-executie Pilot IIIb

Naam	Prio	Input	Output	Split	XSL	Para's	**Resultaat**
Test1	1	Test1.xml	Out1.xml	Nee	Nee	Nee	*Goed*
Test2	2	Test2.xml	Out2.xml	Ja	Nee	Nee	*Goed*
Test3	3	Test3.xml	Out3.xml	Nee	Ja	Nee	*Goed*
Test4	4	Test4.xml	Out4.xml	Ja	Ja	Nee	*Goed*
Test5	5	Test5.xml	Out5.xml	Nee	Nee	Ja	*Goed*

Deze tests had ik oorspronkelijk op een kladblok bijgehouden. Het gehele proces van het genereren, manipuleren en executeren van een batchlist werd getest. Als input werd steeds dezelfde template gebruikt, maar met verschillende bestandsnamen. De batchlist uit de tabel werd meerdere malen uitgevoerd op verschillende stadia in het implementatieproces, met als verschil een veranderende prioriteit van de tasks en af en toe een task verwijderen, toevoegen en de eigenschappen ervan te veranderen. De tests verliepen uiteraard in de wat vroegere stadia van het implementatieproces niet geheel vlekkeloos. Zo kwam het bijvoorbeeld voor dat de prioriteit van de tasks niet altijd correct waren; er kwamen soms dubbele prioriteiten voor en in sommige gevallen zat er zelfs een "gat" in de prioriteitenlijst. Deze bugs werden er tijdens de implementatie uitgehaald. Op het moment dat deze tests vlekkeloos verliepen, was de implementatie ten einde.

8.4.4 De werking van de Batch-functie

De gebruiker is nu in staat de volgende acties te ondernemen:

- Een template met alle bijbehorende opties (naam, input-file, output-file, prioriteit, additionele XSL's, splitoptie en parameters) toevoegen aan de batchlist in de vorm van een task door op de "Add to batchlist" button te klikken in het Input-tabblad.
- De volgorde van executie van de tasks bepalen door de prioriteit van de tasks te wijzigen. Dit kan door een task in de batchlist te selecteren en met behulp van de "Move up" en "Move down" button de positie in de batchlist te wijzigen.
- Een task uit de batchlist verwijderen door de task te selecteren en op de "Delete" button te klikken.
- De volledige batchlist wissen door op de "Reset" button te klikken.
- Een reeds bestaande batchlist openen.
- De huidige batchlist opslaan in de vorm van een XML-bestand.
- De batchlist executeren.
- Een task in de batchlist selecteren, waarna de bijbehorende opties kunnen worden aangepast door op de "Modify" button te klikken in het Input-tabblad. De GUI wordt op dat moment "gevuld" met de opties van de geselecteerde task. Deze kunnen vervolgens worden veranderd.

De volgende punten zijn niet geïmplementeerd, omdat deze een vrij lage prioriteit hadden en als "nice to have" werden gezien door de projectleider. Het leek mij verstandig daarom deze punten niet te implementeren en gewoon verder te gaan met de volgende pilot.

- Tijdstip executie bepalen
- Een template uitsluiten van executie

Tussenconclusie

Omdat het nu voor de gebruiker mogelijk is om meerdere template serieel uit te voeren, voldoet deze pilot aan de gestelde systeemeisen en het systeemconcept.

8.5 Pilot IV

Deze pilot betrof de mogelijkheid tot het gebruik van een FOR-loop in de template waarbij de gebruiker zelf kan bepalen hoe vaak deze loop uitgevoerd dient te worden en welke variabelen er in de loop gebruikt kunnen worden.

Dit moest mogelijk worden gemaakt door hier een speciaal element voor aan te maken waaraan variabelen kunnen worden meegegeven die binnen de loop gebruikt kunnen worden voor bijvoorbeeld het opstellen van een SQL-query of gewoon als output.

Een voorbeeld:

```
<FORLoop ID=”aap, noot, mies, wim, zus, jet”>
        <FORLoopVar/>
</FORLoop>
```

Door dit element in de template te zetten, wordt deze loop zes keer herhaald. In de eerste keer dat de loop doorlopen wordt, wordt het element <FORLoopVar/> vervangen door de eerste variabele. In dit geval is dat “aap”. De tweede keer dat de loop doorlopen wordt, gebeurt hetzelfde, maar dan met de variable “noot”, enzovoorts. De gebruiker is vrij om zoveel variabelen mee te geven als hij of zij wil. De FORLoop kan gebruikt worden binnen een SQLLoop en om een SQLLoop. Ook kunnen er meerdere FORLoops genest worden. Hierbij moet wel in gedachten worden gehouden dat het element <FORLoopVar/> alleen vervangen zal worden door de variabele die gedefinieerd is in de FORLoop waarin het <FORLoopVar/> element geplaatst is.

Vanwege het feit dat FORLoops genest kunnen worden (net als SQLLoops), moest de definitie van dit element in bookdbxml_preprocessor.xsl komen te staan (zie Pilot II). Dit bestand wordt immers net zo lang uitgevoerd totdat alle geneste FORLoops (en SQLLoops) zijn uitgewerkt.

Het had overigens ook gekund om de FORLoops recursief te behandelen, dus dat een FORLoop inclusief de geneste FORLoops in één stap behandeld zouden worden met behulp van een methode in de klasse BookDB die zichzelf steeds aanroept. De reden waarom ik dit niet heb gedaan is omdat ik in Pilot II er al voor heb gekozen om bookdbxml_preprocessor.xml net zo lang toe te passen om SQLLoops uit te werken. Door in dit bestand ook de definitie van de FORLoop te zetten, wordt deze ook net zo lang uitgevoerd totdat alle FORLoops zijn verwerkt.

Als ik ervoor had gekozen om FORLoops recursief te behandelen, had ik een extra XSL moeten maken voor het uitwerken van FORLoops. Deze XSL zou dan nog vóór bookdbxml_preprocessor.xsl uitgevoerd moeten worden. Hierdoor waren dan eerst de FORLoops en daarna pas de SQLLoops uitgewerkt. Hiervoor zou ik dus de functionaliteit die ik in Pilot II heb gerealiseerd weer moeten aanpassen. De definitie van de FORLoop had overigens dan niet in bookdbxml.xsl (die eenmalig wordt uitgevoerd) kunnen staan, omdat er in een FORLoop gedefinieerd zou kunnen staan hoe vaak een SQLLoop herhaald moet worden. Aangezien deze dan al door bookdbxml_preprocessor uitgewerkt zouden moeten zijn, zouden deze SQLLoops dus niet meer uitgevoerd worden en zou er een gewone SQL-query in het eindresultaat komen te staan.

Het uitwerken van de FORLoops werkt als volgt:

Eerst worden de variabelen die meegegeven worden aan het FORLoop-element weggeschreven in een Vector. Voor elke waarde in de Vector gebeurd er het volgende: De inhoud van het <FORLoop>-element wordt gecopieerd in het tussenresultaat. Indien hier er een <FORLoopVar/> element tussen zit, wordt deze vervangen met de waarde uit de Vector. Zodra er binnen het <FORLoop>-element nog een <FORLoop>-element genest zit, wordt dit volledige element (dus inclusief de hierbij behorende <FORLoopVar/>-elementen) gecopieerd naar het tussenresultaat. Deze zal dan pas uitgewerkt worden op het moment dat bookdbxml_preprocessor.xsl opnieuw wordt uitgevoerd. Dan begint dit proces weer van voor af aan. Uiteindelijk zijn dan alle FORLoops uitgewerkt.

Het ontwerp van deze pilot is te vinden in Bijlage D. Omdat deze pilot de ontwikkeling van een proces betrof, heb ik gebruik gemaakt van flowcharts om de verwerking van de FORLoop in beeld te brengen.

Tussenconclusie
Het nu mogelijk voor de gebruiker om zelf te bepalen hoe vaak een bepaald gedeelte in de template kan worden herhaald. Hierbij heeft de gebruiker de keuze om per herhaling een variabele naar keuze te gebruiken. Dit voldoet aan de systeemeisen zoals genoemd in de lijst met systeemeisen en het systeemconcept.

8.6 Pilot V

Deze pilot betrof het mechanisme voor het maken van headers en footers voor gebalanceerde tabellen.

Een header is een stuk tekst dat bovenaan een tabel staat, dit kan bijvoorbeeld de naam van de tabel zijn. Een footer is een stuk text dat onderaan een tabel staat, dit kan bijvoorbeeld additionele informatie over de tabel bevatten. Omdat de header en footer pas tijdens de opmaak (in bijvoorbeeld Adobe FrameMaker) in de tabel zelf worden gezet, is het alleen nodig om in het XML-bestand een extra element te genereren met daarin de text van de header of footer. Dit element kan aan het begin en/of aan het eind van de tabel worden toegevoegd. Het opmaakprogramma zal deze dan herkennen en er een header en/of footer van maken en in de tabel zetten. Door aan de methode waarmee de tabel kan worden gegenereerd, twee extra attributen mee te geven waarmee de header en footer kunnen worden aangemaakt, krijgt de gebruiker de mogelijkheid zelf de header en/of footer te definiëren.

Voor het genereren van een tabel wordt de volgende functie aangeroepen (de attributen “header” en “footer” zijn ontworpen in deze pilot):

```
<BalancedTable columns=”2” tablename=”Letterlijst” rowname=”rij” cellname=”Letter”
        header=”Letters van het alfabet” footer=”De letters staan op alfabetische volgorde”>
        <FORLoop ID=”A, B, C, D, E, F, G”>
          <Letter><FORLoopVar/></Letter>
        </FORLoop>
</BalancedTable>
```

Dit voorbeeld genereert in het eindresultaat een tabel met de naam “Letterlijst”. Deze tabel heeft twee kolommen. De rijen zullen de naam “rij” dragen en de cellen in de tabel zullen gevuld worden met het element <Letter> en alles wat daar tussen staat. Als header heeft deze tabel de tekst “Letters van het alfabet” en als footer de tekst “De letters staan op alfabetische volgorde”. In het eindresultaat ziet de uitkomst er als volgt uit:

```
<Letterlijst cols=”2”>
  <header>Letters van het alfabet</header>
    <rij><Letter>A</Letter><Letter>E</Letter></rij>
    <rij><Letter>B</Letter><Letter>F</Letter></rij>
    <rij><Letter>C</Letter><Letter>G</Letter></rij>
    <rij><Letter>D</Letter></rij>
  <footer>De letters staan op alfabetische volgorde</footer>
</Letterlijst>
```

De tabel is gebalanceerd, wat wil zeggen dat de kolommen even lang zullen zijn. Hierbij kan het zijn dat de laatste kolom één cel korter is dan de overige kolommen, zoals in dit voorbeeld het geval is. Er zal geen echte tabel te zien zijn in het eindresultaat, omdat alles nog steeds is opgebouwd in XML. Het opmaakprogramma zal verder voor de opmaak zorgen wat zal resulteren in een echte tabel.

Het ontwerp van deze pilot is te zien in Bijlage D.

Tussenconclusie
Omdat het nu voor de gebruiker mogelijk is om headers en/of footers te genereren voor gebalanceerde tabellen, voldoet deze pilot aan de gestelde systeemeisen en het systeemconcept.

8.7 Pilot VI

Deze pilot bestond uit een onderzoek naar een bug. Voor deze pilot was een timebox van precies vijf werkdagen ingepland.

8.7.1 De bug

De source-code van BookDB-XML maakt gebruik van het object ResultSet om de uitkomst van een SQL-query in op te slaan. Deze ResultSet heeft de eigenschap “scrollable”, dat wil zeggen dat je elk record in de ResultSet meerdere keren kan bekijken. Dit is nodig voor eventueel toekomstige functies in BookDB-XML. Een ResultSet is standaard niet “scrollable”, in dat geval kan een record slechts éénmalig gelezen worden.

Zodra er een SQL-query werd uitgevoerd met daarin een DISTINCT- of GROUP BY-keyword, dan klopte het aantal records in de “scrollable” ResultSet niet. Een DISTINCT-keyword zorgt ervoor dat records die meerdere malen voorkomen slechts één keer in de ResultSet worden geplaatst. Stel dat er een query met daarin een DISTINCT-keyword werd uitgevoerd. De ResultSet kreeg dan de grootte van het aantal records wat er uit dezelfde query, maar dan zonder DISTINCT-keyword zou komen. Deze records werden echter wel worden gevuld met de juiste waarden, maar omdat de ResultSet dus te groot was werden de overige records leeg (NULL) gelaten. Hierdoor werden deze lege records opgenomen in de transformatie van het eindresultaat. Dit kon resulteren in lege elementen en loops die te vaak werden uitgevoerd.

8.7.2 De oplossing

Er waren twee potentiële mogelijkheden om deze bug op te lossen of te ontlopen:

- Een bug-fix of patch
- Een workaround

Omdat er geen bestaande bug-fix of patch beschikbaar was (ook in JDK 1.5.0 was de bug nog niet opgelost), heb ik een omschrijving van de bug opgenomen in de Sun Java Bug Database. Hierdoor zou de bug misschien in de volgende Java-release kunnen worden opgelost. Nu was het nodig een workaround te vinden om de oorzaak van de bug te omzeilen. Hiervoor had ik verschillende opties:

- Een gewone ResultSet gebruiken (niet “scrollable”)
  Deze optie viel vrijwel gelijk af, omdat de projectleider per se een scrollable ResultSet wou gebruiken omdat dit nodig was voor eventuele toekomstige functies in BookDB.XML

- De NULL-waarden onderaan de ResultSet weghalen en de ResultSet “inkorten”
  Deze optie zou wel werken indien de query wordt uitgevoerd op een enkele tabel. In dat geval zou er een “echte” NULL-waarde bovenaan de ResultSet komen te staan. Het probleem hierbij is dat als de query betrekking heeft op meerdere tabellen dat er dan ook “echte” NULL-waarden in komen te staan en wel op verschillende plaatsen in de ResultSet. Het verschil tussen een “echte” en een onterechte NULL-waarde is dan niet meer te zien. Het gevaar is dan dat een “echte” NULL-waarde misschien wordt verwijderd en daarmee het resultaat niet meer klopt.

- Het gebruik van een PreparedStatement in plaats van een Statement
  Voordat een query wordt uitgevoerd, wordt deze eerst in het object Statement geplaatst en daarna uitgevoerd. Een alternatief was om deze query in een PreparedStatement te zetten en daarna uit te voeren. Er was immers een kans dat hiermee de bug niet meer voor zou komen. Na dit te hebben geprobeerd bleek deze derde optie

geen effect te hebben.

- Een methode maken die zelf een “DISTINCT” of “GROUP BY” uitvoert op de query zonder DISTINCT- of GROUP BY-keyword
  Deze optie zou erg moeilijk worden om te realiseren, dit zou niet lukken binnen vijf werkdagen. Indien deze oplossing toegepast zou worden zou hij (indien goed ontworpen en geïmplementeerd) wel goed werken.

- Een andere JDBC-driver gebruiken
  Deze optie was het gebruik van een alternatieve JDBC-driver. Deze driver zorgt voor de communicatie tussen de Java-applicatie en de database. Het zou immers kunnen dat de driver het resultaat verkeerd in de ResultSet plaatst.) Ik heb twee alternatieve drivers getest, waarvan er eentje bleek te werken en de scrollable ResultSet correct vulde met waarden uit de database. Op basis hiervan heb ik een conclusie getrokken. Deze conclusie en de bevindingen van deze pilot zijn te vinden in Bijlage D. Hierin staat beschreven dat ik de alternatieve JDBC-driver JadoZoom van InfoZoom prefereer om als tijdelijke workaround te gebruiken om de bug te omzeilen.

Tussenconclusie
Omdat er een workaround is gevonden waarmee de bug kan worden ontweken is voldaan aan de systeemeisen en het systeemconcept.

8.8 Pilot VII

Deze pilot betreft het gebruik van een IF/THEN/ELSE statement in de template. Hiermee wordt het voor de gebruiker mogelijk om keuzes te maken in de template aan de hand van waarden uit de database.

8.8.1 Ontwerp

Een IF/THEN/ELSE-statement ziet er globaal als volgt uit:

```
<BDB_IF>
    <BDB_CONDITION>
    </BDB_CONDITION>
    <BDB_THEN>
    </BDB_THEN>
    <BDB_ELSE>
    </BDB_ELSE>
</BDB_IF>
```

Hierbij kan de gebruiker in het <BDB_CONDITION> element de condities voor de selectie definieren. Aan de hand van de uitkomst hiervan wordt óf het <BDB_THEN> óf het <BDB_ELSE> element overgenomen in het tussenresultaat. Indien het <BDB_ELSE> element is weggelaten, wordt er niets overgenomen in het tussenresultaat als de conditie niet waar is. De opbouw voor het <BDB_IF> element is vrij logisch, er zijn eigenlijk geen alternatieven om op een betere manier selectie uit te voeren in de template. Het is overigens niet mogelijk om een IF/ELSE IF statement hiermee uit te voeren, hoewel men wel hetzelfde resultaat kan krijgen door in het <BDB_ELSE>-element weer een <BDB_IF>-element op te nemen.

Het <BDB_CONDITION> element kan uiteindelijk alleen waar (true) of niet waar (false) zijn. Dit wordt gepaald aan de hand van de selectie die de gebruiker in dit element heeft gedefinieerd. Dit kan de gebruiker doen door operators te plaatsen en deze op waarden uit te laten voeren. Deze waarden kunnen zowel door de gebruiker zelf ingevuld zijn, maar het kunnen ook waarden uit de database zijn. De gebruiker heeft de keuze uit de volgende operators:

- <BDB_EQUAL>
- <BDB_NOT_EQUAL>
- <BDB_GREATER>
- <BDB_SMALLER>

- <BDB_GREATER_EQUAL>
- <BDB_SMALLER_EQUAL>
- <BDB_AND>
- <BDB_OR>
- <BDB_NOT>

Deze operators kunnen met elkaar gecombineerd worden, zodat de gebruiker volledig vrij is en alle denkbare soorten selecties kan samenstellen. De operators moeten gescheiden geplaatst worden in <BDB_PART> elementen. Hierdoor kan BookDB-XML makkelijk de conditie opdelen in kleinere stukken en deze apart uitvoeren.

Voordat de conditie kan worden uitgevoerd, moeten eerst de waarden uit de database in de conditie worden geplaatst. Dit gebeurt al op het moment dat de bookdbxml_preprocessor.xml is uitgevoerd. Het uitvoeren van de conditie gaat van binnen naar buiten. Dat wil zeggen dat eerst de diepst gelegen operators worden uitgevoerd. Die diepst gelegen operator heeft altijd betrekking op de waarde uit een database of een door de gebruiker ingevulde waarde. Aan de hand hiervan kan de uitkomst van de operator worden bepaald. De operator met inhoud wordt vervolgens vernietigd (deze is dan niet meer nodig) en het <BDB_PART> element waarin de operator zich bevindt krijgt een element met de naam TRUE of FALSE. Nu de uitkomst van deze operator bekend is kan pas de bovenliggende operator worden uitgevoerd. Dit proces gaat net zo lang door totdat de bovenste operator is berekend. Deze geeft uiteindelijk aan of de conditie waar of niet waar is. Ook in het <BDB_CONDITION> element wordt dan een elementje geplaatst met de naam TRUE of FALSE.

Het kan zo zijn, dat er zich in een <BDB_THEN> of <BDB_ELSE> element weer een <BDB_IF> element genest is. De verwerking van geneste <BDB_IF> elementen kon ik op twee manieren aanpakken:

- De definitie van het <BDB_IF> element in de bookdbxml_preprocessor.xml zetten. (Deze wordt immers net zo lang herhaald totdat alle geneste <BDB_IF> elementen zouden zijn verwerkt.)

- Alle geneste <BDB_IF> elementen in één keer (en dus recursief) verwerken. (Hierbij komt de definitie van het <BDB_IF> element in bookdbxml.xsl te staan en wordt de rest in de klasse BookDB afgehandeld.)

Ik heb gekozen voor de tweede optie en wel om de volgende reden:

De eerste optie zou niet werken, omdat er tijdens het uitvoeren van de bookdbxml_preprocessor.xml alle waarden uit de database in het tussenresultaat worden gezet. Hier worden immers alle SQL-queries al uitgevoerd. Als op ditzelfde moment wordt geprobeerd een <BDB_IF> element uit te voeren, gaat dit mis. Dit komt omdat het <BDB_CONDITION> element niet kan worden verwerkt als de waarden uit de database op dat moment nog niet bekend zijn. Dit probleem zou kunnen worden opgelost door een extra XSL-bestand te maken die speciaal voor de verwerking van het <BDB_IF> element is bedoeld. Deze zou dan pas na de executie van de bookdbxml_preprocessor.xsl moeten worden uitgevoerd. Hiervoor zou ik de functionaliteiten uit Pilot II weer aan moeten passen. Dit is hetzelfde probleem als ook bij Pilot IV ter sprake is gekomen. Wederom heb ik ervoor gekozen dit niet te doen, omdat dit veel omslachtiger zou zijn dan de tweede optie: alle geneste <BDB_IF> elementen in één keer verwerken. Dit werkt als volgt:

Voordat het <BDB_THEN> of <BDB_ELSE> element in het tussenresultaat wordt geplaatst, wordt eerst gecontroleerd of er zich hierin een <BDB_IF> element bevindt. Als dat het geval is, wordt eerst de inhoud van het <BDB_THEN> of <BDB_ELSE> element in het tussenresultaat geplaatst (behalve het geneste <BDB_IF> element), waarna dit <BDB_IF> statement wordt uitgevoerd. De uitkomst hiervan wordt dan weer in het tussenresultaat geplaatst. Dit gaat net zo lang door totdat alle geneste <BDB_IF> elementen verwerkt zijn.

Het is ook mogelijk om attributen aan operators mee te geven om bepaalde opties mee te geven betreffende het vergelijken van waarden. Zo kan er bijvoorbeeld worden gespecificeerd of er case-sensitive moet worden vergeleken of juist niet. Ook is het mogelijk om met NULL-waarden te vergelijken. Met de <BDB_EQUAL> operator kunnen bijvoorbeeld zowel teksten als getallen worden vergeleken. Het is zelfs mogelijk om met bijvoorbeeld de <BDB_GREATER>, <BDB_SMALLER>, <BDB_SMALLER_EQUAL> en <BDB_GREATER_EQUAL> operators (naast getallen) ook stukken tekst alfabetisch te vergelijken. Met het attribuut “focus_on_digits” is het zelfs mogelijk waarden te vergelijken die beginnen of eindigen met cijfers. Hierbij wordt dan alleen het cijfergedeelte van de waarde vergeleken.

De ontwerpen van deze pilot zijn te vinden in Bijlage D. Omdat het hier een proces betrof dat ontworpen diende te worden, heb ik hier wederom gebruik gemaakt van flowcharts.

8.8.2 Implementatie

De volledige afhandeling van het IF/THEN/ELSE-statement vond plaats met behulp van deze vier belangrijkste methodes in de klasse BookDB:

- *public DocumentFragment IfThenElse(XSLProcessorContext context, org.w3c.dom.Element elem)*
  Aan deze methode wordt het volledige <BDB_IF> element meegegeven zodra Xalan het element in de template tegenkomt. Er wordt dan direct de methode “*private boolean prepareIfThenElse(Node node)*” aangeroepen om te controleren of het <BDB_IF> element en eventueel geneste <BDB_IF>-elementen op de correcte manier zijn opgebouwd. Verder verwerkt deze methode het IF-statement en alle hierin geneste <BDB_IF> elementen. Dit gebeurd door de volgende methode aan te roepen voor elk <BDB_IF> element:

- *private Node runIfThenElse(Node node, Document doc)*
  Deze methode laat het <BDB_CONDITION> element uitvoeren en zet aan de hand daarvan het <BDB_THEN> element, <BDB_ELSE> element, of niets in het tussenresultaat. Het <BDB_CONDITION> element wordt verwerkt met de volgende methode:

- *private void IfIterator(Node node)*
  Deze methode verwerkt het gehele <BDB_CONDITION> element van binnen naar buiten uit, totdat het <BDB_CONDITION> element uiteindelijk TRUE of FALSE is. Voor elke operator binnen het <BDB_CONDITION> element, wordt de volgende methode aangeroepen:

- *private boolean processOperator(Node executableNode)*
  Aan deze methode wordt een operator met bijbehorende waarden meegegeven. De uitkomst hiervan is altijd TRUE of FALSE.

Zoals hierboven genoemd wordt het <BDB_CONDITION> elementen van binnen naar buiten verwerkt. Er wordt dus begonnen bij de diepst liggende operator (deze heeft altijd een waarde uit de database of een door de gebruiker ingevulde waarde). Dit gebeurt voor elk “nest-niveau”. Wat hierbij erg belangrijk is, is dat een operator pas uitgevoerd mag worden op het moment dat al zijn kind-operators zijn uitgevoerd en dus niet eerder. Dit komt voor op het moment dat een gedeelte van het <BDB_CONDITION> element meer geneste operators heeft dan een ander gedeelte ervan. Hieronder volgt een voorbeeld waarbij met tussen <!-- -->-elementen wat commentaar is toegevoegd:

```
<BDB_CONDITION>
<BDB_EQUAL> <!-- Nest-niveau 1 -->
      <BDB_PART>
         <BDB_GREATER> <!-- Nest-niveau 2 -->
            <BDB_PART>24</BDB_PART>
            <BDB_PART> 20</BDB_PART>
          </BDB_GREATER>
      </BDB_PART>
```

```
        <BDB_PART>
          <BDB_AND> <!-- Nest-niveau 2 -->
              <BDB_PART>
                <BDB_GREATER_EQUAL> <!-- Nest-niveau 3 -->
                    <BDB_PART>50</BDB_PART>
                    <BDB_PART>50</BDB_PART>
                </BDB_GREATER_EQUAL>
              </BDB_PART>
              <BDB_PART>
                <BDB_SMALLER_EQUAL> <!-- Nest-niveau 3 -->
                    <BDB_PART>33</BDB_PART>
                    <BDB_PART>33</BDB_PART>
                </BDB_SMALLER_EQUAL>
              </BDB_PART>
           </BDB_AND>
        </BDB_PART>
</BDB_EQUAL>
</BDB_CONDITION>
```

Het <BDB_EQUAL> element bevindt zich hier op “nest-niveau 1” en moet de uitkomst van zijn twee <BDB_PARTS> elementen (true of false) met elkaar vergelijken. De operator van het eerste <BDB_PART> element bevindt zich op “nest-niveau 2”. De <BDB_AND> operator van het tweede <BDB_PART> element ook. Echter bevat dit <BDB_AND> element twee operators die zich op “nest-niveau 3” bevinden. Omdat er steeds wordt begonnen bij de operators die een waarde bevatten en van daaruit naar lagere niveaus wordt gewerkt, zal het eerste <BDB_PART> element dus eerder op true worden gezet dan het tweede <BDB_PART> element. Deze loopt immers op dat moment nog een “ronde” achter. Hierdoor kan het <BDB_EQUAL> element nog niet uitgevoerd worden. Deze moet daarom in de “wacht” worden gezet totdat ook het tweede <BDB_PART> element berekend is en op true is gezet.
In de source-code is dit op de volgende manier geregeld.

```
private void IfIterator(Node node)
{
     //Een operator-element komt binnen... (De eerste keer is dit het volledige BDB_CONDITION element.)
     Element element = (Element)node;
     NodeList partList = element.getElementsByTagName("BDB_PART");
     Node nodeParent = null;

      //Voor elk BDB_PART-element van het operator-element wordt het volgende uitgevoerd:
      for (int a=0; a<partList.getLength(); a++)
     {
        Element elem = (Element)partList.item(a);
        NodeList list = elem.getElementsByTagName("*");
        //Nu voor elk BDB_PART element die nog een operator heeft, nogmaals de IfIterator uitvoeren...
        if (list.getLength() > 0)
        {
          IfIterator(partList.item(a));
        }

        //Pas als alle kind-nodes berekend zijn, mag het operator-element op true/false worden gezet...
        if (a == partList.getLength() - 1)
        {
          nodeParent = partList.item(a).getParentNode();
          if (processOperator(nodeParent))
          {
            Element newElem = nodeParent.getOwnerDocument().createElement("TRUE");
            nodeParent.getParentNode().replaceChild(newElem, nodeParent);
          }
          else
          {
            Element newElem = nodeParent.getOwnerDocument().createElement("FALSE");
            nodeParent.getParentNode().replaceChild(newElem, nodeParent);
          }
        }
     }
}
```

8.8.3 Testen

Om te testen of de functionaliteit ook naar behoren werkte (zoals omschreven in het systeemconcept en ontwerp), heb ik voornamelijk tijdens het programmeren testen uitgevoerd. Hierdoor kon ik steeds controleren of de code die ik tot dan toe had geschreven ook daadwerkelijk correct werkte. Hierbij maakte ik gebruik van een zeer uitgebreide templates waarin alle operators voorkwamen in verschillende combinaties, ook in geneste <BDB_IF> elementen. Ook kwamen hier andere elementen zoals SQLLoops en FORLoops voor om te testen of het IF/THEN/ELSE statement ook goed zou werken met overige functionaliteiten. Toen er geen fouten meer optraden bij het uitvoeren van deze templates, was de pilotontwikkeling voltooid.

De pilotontwikkeling van deze pilot vond ik de lastigste tijdens mijn afstudeerperiode; toch verliep de pilotontwikkeling vrij soepel en snel.

Tussenconclusie
Omdat de gebruiker nu de mogelijkheid heeft om keuzes te maken in de template aan de hand van waarden uit de database, is voldaan aan de gestelde systeemeisen en het systeemconcept.

8.9 Pilot VIII

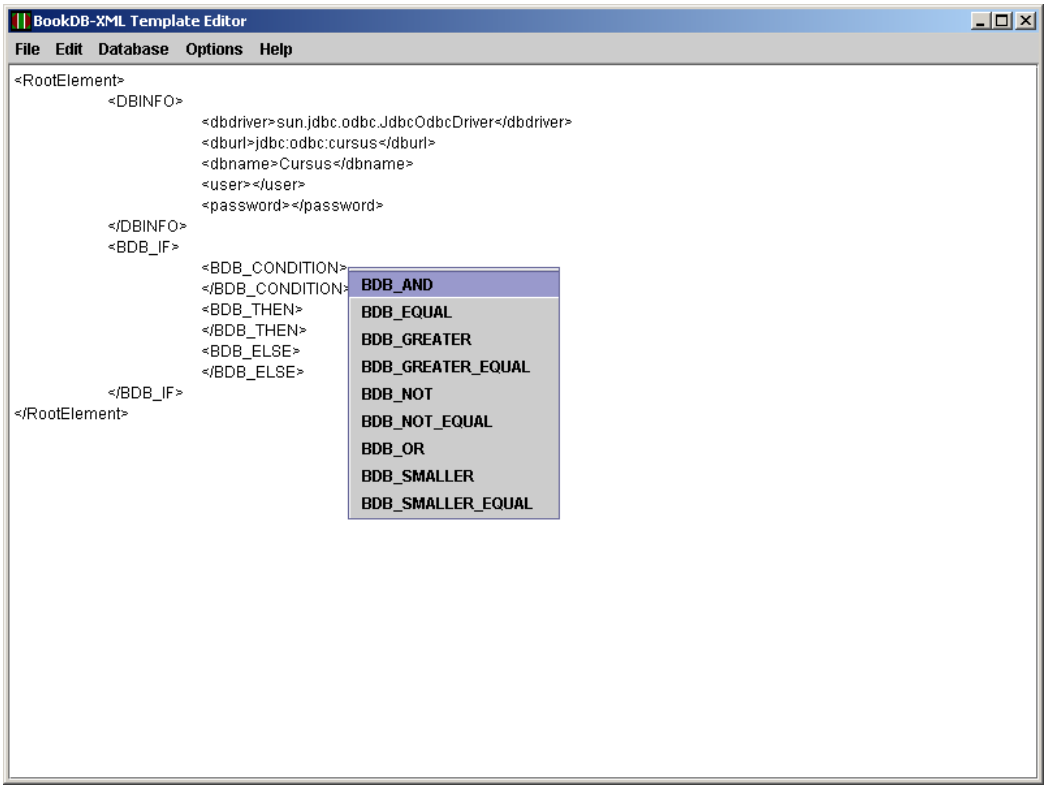
Deze pilot betreft het ontwerpen van een Grafische User Interface (GUI) voor het samenstellen van templates. Omdat ik na het ontwerpen van de GUI nog genoeg tijd over had, besloot ik om de GUI ook te implementeren.

8.9.1 Ontwerp

Het maken van een template is niet makkelijk voor de gebruiker, er is immers kennis van XML en SQL voor vereist om een correcte template samen te stellen die precies de juiste uitvoer produceert die de gebruiker wil. De gebruiker moet immers zelf bepalen welke data er precies nodig is en hoe deze moet worden gestructureerd. Het is daarom gewoon niet mogelijk om dat te automatiseren. Het kan voor de gebruiker alleen maar gemakkelijker worden gemaakt door de gebruiker bepaalde handelingen uit handen te nemen en te automatiseren. Dit betreft het automatisch aanmaken van elementen op plaatsen waar dat is toegestaan. Hierbij wordt ook de kans op het maken van fouten sterk verminderd. Tevens kan de gebruiker geholpen worden bij het schrijven van de SQL-queries door de beschikbare tabellen en kolommen weer te geven. Met deze techniek is het mogelijk om het grootste gedeelte van de template te genereren met behulp van alleen de muis.

Ik had ook een GUI kunnen ontwerpen waarin alle elementen grafisch worden weergegeven en dat deze in- en uitklapbaar zijn. Dit is echter zeer lastig te realiseren en bovendien geeft het vrijwel geen meerwaarde aan de gebruiker.

Ik heb ervoor gekozen om de GUI de naam “BookDB-XML Template Editor” te geven. Deze kan worden opgestart vanuit het hoofdmenu van BookDB-XML. Ik wilde de editor een zo kaal mogelijk uiterlijk geven, zodat er zo veel mogelijk ruimte over zou blijven voor het tekst-veld waarin de template zou komen te staan. Op de volgende pagina staat een screenshot van de BookDB-XML Template Editor.



In deze screenshot is al een gedeelte van een template gemaakt. Het root-element, DBINFO en BDB_IF zijn reeds aangemaakt. De gebruiker klikte op dit moment op de rechter muisknop vlak achter het BDB_CONDITION element. De gebruiker krijgt nu de keuze tussen een aantal elementen die is toegestaan binnen dit BDB_CONDITION element. Indien er op één van deze elementen wordt geklikt, zal deze in de template worden gezet.

De template wordt in een tekst-veld getoond aan de gebruiker. Het is mogelijk voor de gebruiker om rechtstreeks de template te wijzigen met zowel de opties die de template editor biedt als handmatig. De wijzigingen aan de template die de gebruiker doorvoert met behulp van de opties die de template editor biedt, kon op twee manieren plaatsvinden:

- Operaties op de template in de vorm van een XML-bestand uitvoeren.
  Hiermee moest er gebruik gemaakt worden van de methoden die de Java API biedt met betrekking tot XML-documenten.

  Voordelen:
  - Het zoeken naar elementen in een XML-document is makkelijk.
  - Het openen, opslaan en valideren van XML-documenten gaat makkelijk omdat deze reeds in XML-formaat zijn.
  - Het toevoegen van tabs (inspringen) gaat automatisch.

  Nadelen:
  - Het is niet mogelijk een door de gebruiker geselecteerde positie te “markeren” in het document.
  - Bij elke wijziging in de template moet deze weer geconverteerd worden naar String om hem vervolgens weer te tonen in het tekst-veld, omdat het tekst-veld alleen Strings kan tonen, geen XML-bestanden.

- Operaties op de template in de vorm van een String uitvoeren.
  Hiermee moest er gebruik gemaakt worden van de methoden die de Java API biedt aan het object String.

Voordelen:
- Door de gebruiker geselecteerde positie is makkelijk te “markeren” in een String.
- Wijzigingen in de template kunnen gelijk worden doorgevoerd, omdat het tekst-veld de template als String rechtstreeks toont. De gebruiker kan tevens rechtstreeks de template manipuleren door erin te typen.

Nadelen:
- Het gebruik van tabs (inspringen) in de template is lastig.
- Voor het opslaan, openen en valideren van een template moet de String geconverteerd worden in een XML-bestand en andersom.
- Voor het zoeken naar elementen in een String gaat lastig. De API biedt hier geen bestaande methoden voor.

De keuze tussen deze twee opties was vrij lastig. Beide opties zijn mogelijk en kunnen gebruikt worden. Uiteindelijk heb ik toch gekozen voor het gebruik van een String, omdat het bij een String mogelijk is een positie te “markeren” en de template rechtstreeks kon worden aangepast in het tekst-veld. Deze voordelen vond ik belangrijker dan de voordelen van de eerste optie.

8.9.2 Implementatie

De volledige template editor is in één klasse gemaakt, genaamd “TemplateEditor”. Hieronder de belangrijkste methoden van deze klasse met uitleg om inzicht te gegeven in de bouw van deze klasse.

- public static void initMainScreen()
  Deze methode genereerd de GUI van de BookDB-XML Template Editor.

- public static void addRootElement()
  Deze methode zorgt volledig voor het toevoegen van het root-element.

- public static void saveAs()
  Deze methode zorgt voor het opslaan van de template.

- public static void createDBINFO(String driver, String url, String dbname, String user, String password)
  Deze methode genereerd het DBINFO-element en voegt deze toe aan de template.

- public static void insert(String text, int position)
  Deze methode wordt aangeroepen om een bepaald stuk tekst op een bepaalde positie in het tekst-veld te zetten. Dit betreft in de meeste gevallen een element.

- public static int position()
  Dit is de huidige positie van de muis als op de rechter muisknop wordt geklikt.

- public static void showDatabaseContents(Vector toShow)
  Deze methode zorgt voor een overzichtelijk menu waarin alle tabellen en kolomnamen van alle bestaande DBINFO-elementen worden getoond. Hieraan wordt een Vector meegegeven die de tabel- en kolomnamen bevat van de DBINFO-elementen.

- public static void loadElements()
  Deze methode leest de bestaande BookDB-XML-elementen in uit het bestand bookdbelements.ini. Dit bestand kan naar wens worden aangepast in verband met gewenste attributen, tabs en spaties.

- public static JPopupMenu createElementList()
  Deze methode genereerd een menu waarin elementen staan die op de geselecteerde positie voor mogen komen..

- public static void checkAllowedElements()
  De methode bepaalt welke elementen er voor mogen komen op de huidige positie.

- public static int checkTabs()
  Deze methode controleert hoeveel tabs er op de huidige positie moet worden ingesprongen voor het toe te voegen element.

- public static String addTabs(String text, int level)
  Deze methode voegt daadwerkelijk de tabs toe aan het toe te voegen element.

- public static int countStringInString(String toSearchIn, String toBeCounted)
  Deze methode wordt gebruikt om te kijken hoe vaak een bepaalde String in een andere String voorkomt.

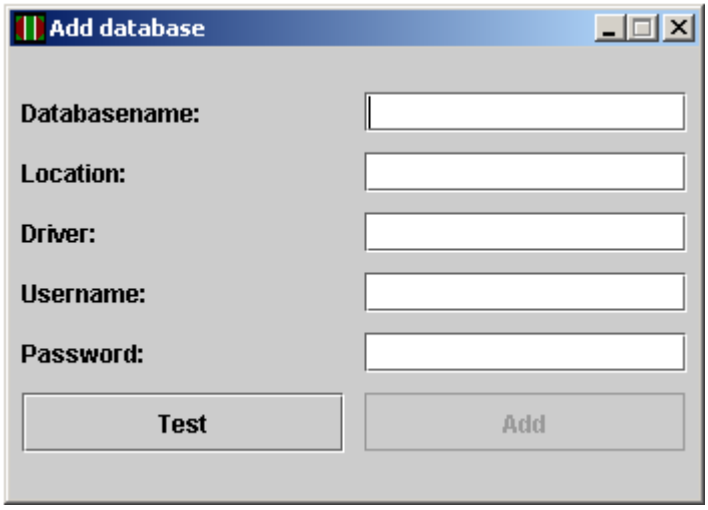
- public static void findAndReplace()
  Deze methode zorgt volledig voor de Find & Replace functie.

- public static boolean validateTemplate(boolean validateClicked)
  Deze methode controleert of de huidige template wel of niet valide is.

- public static void testDatabases(int way)
  Deze methode wordt gebruikt om te testen of de databaseconnecties wel of niet werken.

8.9.3 De werking

Omdat een template twee verplichte elementen moet hebben (een root-element en één of meerdere databaseconnecties in de vorm van een <DBINFO>-element), heb ik ervoor gekozen om het genereren van deze verplichte elementen volledig te automatiseren en de gebruiker te “dwingen” deze elementen toe te voegen aan de template. Hierbij hoeft de gebruiker alleen de naam voor het root-element in te voeren. Voor het genereren van het <DBINFO> element is meer informatie nodig. Zodra de gebruiker een database wil toevoegen, krijgt hij of zij het volgende scherm in beeld:

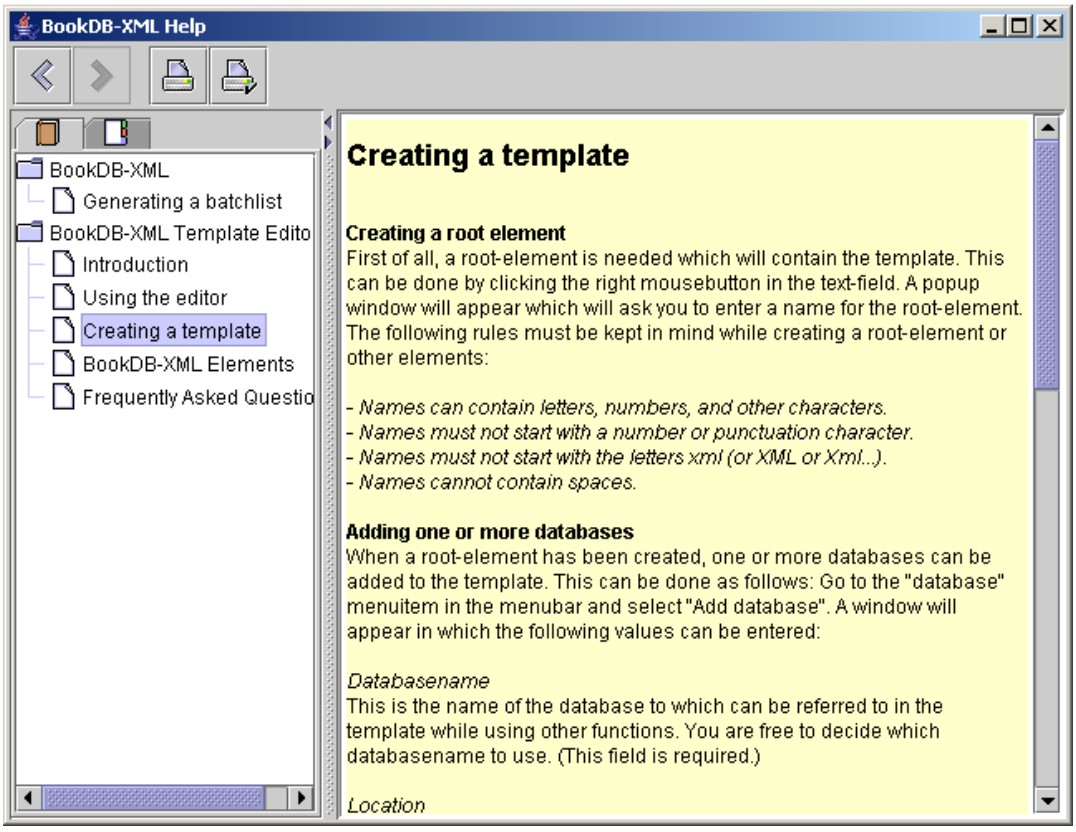


Hier kan de gebruiker de benodigde veldjes invullen en vervolgens testen of de database-connectie ook daadwerkelijk werkt. Pas als dat het geval is, kan met de “Add” button een kant en klaar <DBINFO> element aan de template worden toegevoegd. Indien de gebruiker de waarden veranderd, moet er opnieuw getest worden.

Zodra er een reeds bestaande template wordt geopend, is het uiteraard erg handig om te controleren of de hierin gedefinieerde database-connecties ook daadwerkelijk werken. Ook hiervoor is een functie gemaakt die dat kan testen. De gebruiker krijgt dan een popup-venstertje met daarin van elke database-connectie de informatie of de connectie werkt of niet.
De gebruiker heeft ook de optie om te testen of de template valide is of niet. Een template mag niet worden opgeslagen als deze niet valide is.

Op het moment dat de gebruiker op de rechter muisknop drukt op een bepaalde positie in het veld, wordt de template in twee Strings opgedeeld op het punt van de geselecteerde positie. Door in deze Strings te zoeken naar (delen van) bepaalde elementen, kan worden beslist of bepaalde elementen wel of niet voor mogen komen op deze positie. Aan de hand van deze informatie wordt de lijst met elementen bepaald waar de gebruiker dan uit kan kiezen. Een <FORLoopVar/>-element mag bijvoorbeeld alleen voorkomen als de cursor zich binnen een FORLoop begeeft en een <SQLField>-element mag alleen voorkomen indien de cursor zich binnen een SQLLoop begeeft. Bij het plaatsen van een <SQLField>-element kan met behulp van een popup-menu met dropdown-list worden gekozen op welke SQLLoop het <SQLField>-element betrekking moet hebben. Hierbij worden de attributen van het SQLField-element automatisch ingevuld. Hierbij worden uiteraard de SQLLoops die al afgesloten zijn voor het punt waar de cursor staat, niet als keuze aangeboden aan de gebruiker.

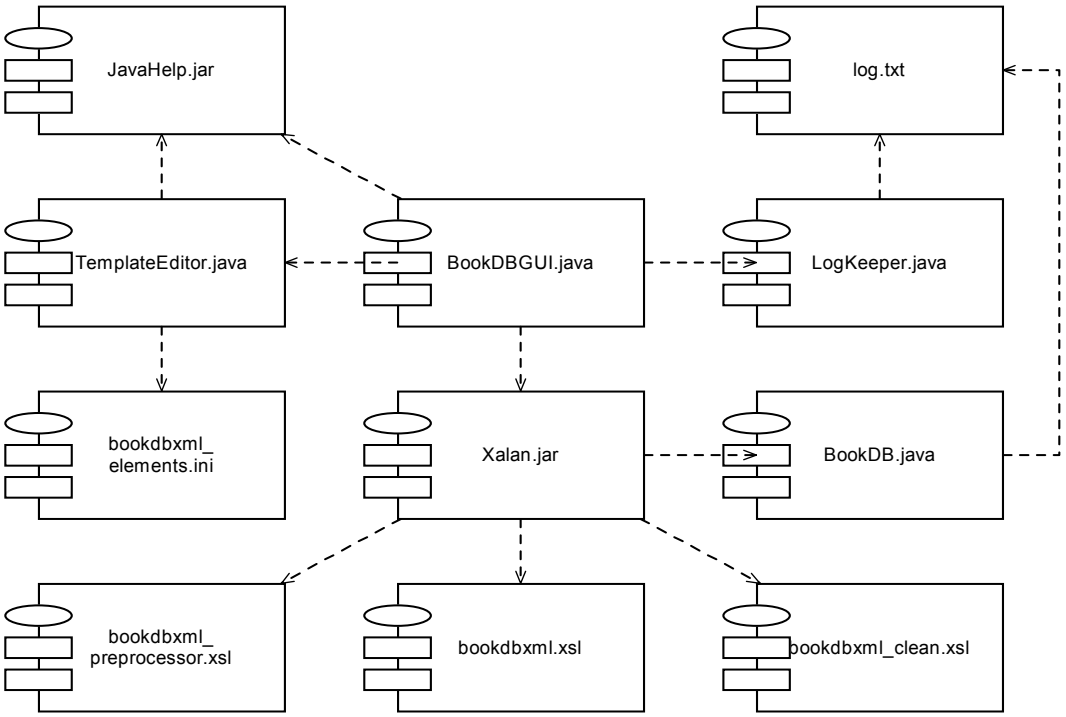
Als “extraatje” heb ik een help-functie gemaakt (op basis van het bestaande JavaHelp). Deze functie toont een uitgebreid help-menu. Hieronder is een screenshot van deze help-functie te zien.



Met behulp van deze help-functie kan de gebruiker op elk moment snel hulp oproepen. Met behulp van XML en HTML zijn de pagina’s opgemaakt. De help-functie kan makkelijk worden uitgebreid met nieuwe pagina’s.

8.9.4 Uiteindelijke architectuur

Na de realisatie van deze pilot was mijn afstudeeropdracht ten einde. De architectuur van BookDB-XML is daardoor veranderd. Het componentdiagram, al eerder genoemd in Hoofdstuk 5, ziet er nu als volgt uit:



De nieuwe klassen TemplateEditor en LogKeeper worden beide vanuit BookDBGUI geïnitieerd. LogKeeper en BookDB gebruiken het bestand log.txt voor het respectievelijk ophalen en wegschrijven van (fout)meldingen. TemplateEditor gebruikt bookdbxml_elements.ini voor het ophalen van bestaande element-definities. Het bestand JavaHelp.jar wordt gebruikt door zowel TemplateEditor als BookDBGUI. JavaHelp.jar gebruikt op zijn beurt de XML-bestanden en HTML-pagina's voor het weergeven van de help-functie. Deze bestanden zijn niet weergegeven in het figuur. Verder is er één XSL-bestand bijgekomen: bookdbxml_preprocessor.xsl.

Tussenconclusie

Omdat het nu voor de gebruiker mogelijk is om op een gebruiksvriendelijke manier een template samen te stellen, voldoet deze pilot aan de systeemeisen en het systeemconcept.

9. Conclusie

In dit hoofdstuk staat beschreven of de doelstelling van mijn afstudeeropdracht is behaald.

De probleembeschrijving en doelstelling van mijn afstudeeropdracht luiden als volgt:

Probleembeschrijving
Sabern Computer Software wenst dat de functionaliteit van BookDB-XML uitgebreid wordt. Er ontbreekt nog een aantal functionaliteiten. Deze ontbrekende functionaliteiten dienen te worden geanalyseerd, ontworpen en geïmplementeerd. Dit betreft de volgende punten:

- Finetunen van bestaande functionaliteiten.
- Ontwerpen en implementeren van algemene nieuwe functionaliteiten.
  Denk hierbij aan seriële executie van taken en aan de mogelijkheid om in een template iteratie en selectie op te nemen.
- Het ontwerpen en eventueel implementeren van een grafische user-interface voor het samenstellen van templates.

Doelstelling
De doelstelling van de feitelijke opdracht is de hierboven genoemde functionaliteiten te realiseren, zodat de functionaliteit van BookDB-XML uitgebreid wordt.

De tussenconclusies, genoemd aan het eind van de beschrijving van elke pilot of pilotdeel, tonen aan dat aan alle systeemeisen is voldaan. De gerealiseerde functionaliteiten voldoen aan het systeemconcept en zijn precies gebouwd volgens de ontwerpen. De bovenstaande functionaliteiten zijn geanalyseerd, ontworpen en geimplementeerd. Hiermee is de functionaliteit van BookDB-XML uitgebreid.

Er kan worden geconcludeerd dat de doelstelling met succes is behaald.

10. Evaluatie

In dit hoofdstuk is mijn persoonlijke visie gegeven over het afstudeerproces en de daarbij opgeleverde produkten. Ook de problemen die ik ben tegengekomen en verbeterpunten staan in dit hoofdstuk beschreven.

10.1 Het proces

Ik ben zelf tevreden met het gehele proces, omdat ik geprobeerd heb om alle activiteiten zo goed mogelijk via de regels van de systeemontwikkelingsmethode uit te voeren. Hierdoor had ik het project goed in de hand en was ik goed voorbereid op eventuele onzekere situaties. Alles verliep sneller dan verwacht.

Achteraf denk ik dat IAD de juiste systeemontwikkelingsmethode was om te gebruiken voor dit proces, omdat ik de doelstelling van mijn afstudeeropdracht met behulp van IAD met succes en zonder problemen heb kunnen behalen.

Zelfstandig kunnen werken vind ik belangrijk, daarom heb ik ook geprobeerd alles zoveel mogelijk zelf te doen. Alleen als ik echt ergens niet uit kwam, riep ik de hulp in van collega's. Dit heb ik twee keer gedaan bij programmeerproblemen bij Pilot IV en VII.

Tijdens mijn stage bij Dutch Space ben ik erachter gekomen dat de implementatie makkelijker en beter verloopt als je de analyse en het ontwerp goed hebt uitgedacht en je je goed hebt verdiept in het systeem. Door dit in het achterhoofd te houden en veel aandacht te besteden aan de analyses en de ontwerpen, verliep de realisatie van de pilots naar mijn mening ook vlot en goed. Hierdoor had ik ook het gevoel dat ik de opgeleverde produkten op de juiste manier had gerealiseerd op de manier waarop de opdrachtgever dat ook had gewild.

10.2 Het produkt

Zelf ben ik tevreden over de produkten die ik heb opgeleverd, omdat de produkten voldoen aan de systeemeisen en het systeemconcept. Tevens zijn de produkten precies gemaakt volgens de ontwerpen. De opgeleverde produkten dekken de probleemstelling volledig. Bovendien zijn er tot nu toe weinig bugs naar boven gekomen in de door mij opgeleverde produkten. De bugs die tot nu toe zijn ontdekt in de door mij geproduceerde produkten zijn tevens verwijderd.

Ik heb met veel plezier aan de produkten gewerkt en voelde me verantwoordelijk en betrokken bij het project. Tijdens mijn afstudeerperiode heb ik ideeën opgedaan met betrekking tot BookDB-XML. Ik heb deze ideeën uitgewerkt en vermeld in een visiedocument dat te vinden is in Bijlage F.

10.3 Problemen

Ik heb twee keer de hulp ingeroepen van een projectlid. Bij Pilot IV en VII had ik moeite met het schrijven van een tweetal methoden. Dit betrof recursieve methoden. Ik vond het erg lastig om een goed overzicht te krijgen voor mezelf van deze methoden, omdat ze zichzelf steeds aanroepen en het daardoor lastig is om te begrijpen wat er precies met de data gebeurd die door de methode wordt bewerkt. Doordat een projectlid mij hielp bij het opstellen van deze methoden, heb ik een goed inzicht gekregen in deze methoden en hun werking. Dit is op zich geen probleem, maar ik vond het wel moeilijk.

Voor de rest hebben er zich tijdens mijn afstudeerperiode geen problemen voorgedaan.

10.4 Verbeterpunten

Omdat ik goed heb nagedacht over alle beslissingen die ik heb genomen, ben ik van mening dat dit de juiste beslissingen zijn geweest. Ik sta er dan ook volledig achter. Uiteraard kunnen bepaalde punten altijd beter. Zo had ik bijvoorbeeld uitgebreidere tests uit kunnen voeren om meer bugs boven water te krijgen.

Bijlagen

Bijlage A - Definitieve opdrachtsomschrijving

"Uitbreiding van functionaliteiten van BookDB-XML bij Sabern Computer Software"

Sabern Computer Software B.V., opgericht in 1990, assisteert organisaties bij het opzetten van databasepublishing, content management, XML-applicaties, dynamische websites en documentenbeheer. Sabern ontwikkelt en levert standaard software en maatwerk toepassingen. Verder geeft Sabern cursussen en verleent consultancy.

BookDB-XML is database publishing software voor het opmaken van teksten, tabellen, plaatjes en grafieken. Het systeem maakt gebruik van XML voor de document opmaak. Met BookDB-XML kan een XML document worden gemaakt voor papieren of digitale uitvoer. Met BookDB-XML kunnen de gegevens afkomstig van database-systemen worden gebruikt als basis voor opgemaakte documenten in XML. In het template kunnen de vaste elementen en de opmaak-informatie worden gedefinieerd. In XSLT's worden de selecties op de database gedefinieerd en wordt de uitvoer opgebouwd.

Sabern wenst dat de functionaliteit van BookDB-XML uitgebreid wordt. Er ontbreekt nog een aantal functionaliteiten. Deze ontbrekende functionaliteiten dienen te worden geanalyseerd, ontworpen en geïmplementeerd. Dit betreft de volgende punten:

1. Finetunen van bestaande functionaliteiten.
2. Ontwerpen en implementeren van algemene nieuwe functionaliteiten. Denk hierbij aan seriële executie van taken en aan de mogelijkheid om in een template iteratie en selectie op te nemen.
3. Ontwerpen en eventueel implementeren van een grafische user-interface voor het samenstellen van templates.

Het doel van de afstudeeropdracht is de hierboven genoemde functionaliteiten te realiseren, zodat de functionaliteit van BookDB-XML uitgebreid wordt.

De volgende software zal gebruikt worden:

- Adobe FrameMaker voor het maken van documenten.
- Software voor het maken van ontwerpen (Paraben's FlowCharter).
- Een texteditor voor het schrijven en bekijken van de source-code (EditPlus 2).
- Een Java-compiler en runtime-environment.
- BookDB-XML.

Er is documentatie over BookDB-XML, bestaande uit gebruikershandleidingen en systeemdocumentatie in de source-code beschikbaar.

In het kader van de afstudeeropdracht zullen de volgende activiteiten verricht worden:

1. Verplichte functionaliteiten:
   - Definitiestudie.
   - Pilotontwikkeling en invoering pilot I, II, III, IV en V.
2. Overige functionaliteiten:
   - Definitiestudie.
   - Pilotontwikkeling en invoering pilot VI, VII en VIII.

Bij de uitvoering van de opdracht zal IAD gehanteerd worden. De iteratiestrategie die gebruikt zal worden is een combinatie van incrementeel ontwikkelen en incrementeel opleveren. Voor de door Sabern verplicht gestelde functionaliteiten wordt eenmalig de definitiestudie uitgevoerd, gevolgd door een pilotontwikkeling voor elke pilot. Hierna wordt eventueel voor elke pilot of meerdere pilots tegelijk de invoer ing uitgevoerd. Voor de overige functionaliteiten wordt nogmaals een definitiestudie uitgevoerd, gevolgd door een pilotontwikkeling voor elke pilot met daarna de invoering voor elke pilot of voor meerdere pilots tegelijk.

De volgende technieken zullen gebruikt worden:

- Communicatie met collega’s door middel van kleine workshops.
- Timeboxing.
- Juicy Bits First.
- Java.
- Extended Markup Language (XML).
- Extensible Stylesheet Language Transformations (XSLT).
- Unified Modelling Language (UML).

De volgende producten zullen opgeleverd worden:

- Plan van aanpak.
- Ontwikkelscenario.
- Systeemeisen.
- Systeemconcept.
- Pilotplan.
- Pilotontwikkelplannen.
- Ontwerpen van de software-bouweenheden.
- Pilots.
- Geactualiseerde handleidingen.

Bijlage B - Literatuurlijst / Bronvermelding

- “Adobe FrameMaker 7.0 - Classroom In A Book”
  The official training workbook developed by the staff of Adobe
  ISBN: 0-321-12168-1

- “The Complete Reference FrameMaker 7”
  Sarah S. O’Keefe & Sheila A. Loring
  ISBN: 0-07-222361-8

- “IAD - Het evolutionair ontwikkelen van informatiesystemen (Pilot 2)”
  R.J.H. Tolido
  ISBN: 90-395-0401-6

- “De UML Toolkit”
  Hans-Erik Eriksson & Magnus Penker
  ISBN: 90-395-1015-6

- http://www.w3schools.com
  Informatie over XML, XSL(T)

- http://java.sun.com/j2se/1.4.2/docs/api & http://java.sun.com/j2se/1.5.0/docs/api
  De Java API

Bijlage C - Contactinformatie

Gegevens afstudeerder:

Naam: Christian Jansen
Studentnummer: 20013115
Adres: Phia Berghoutlaan 1 2343 PM Oegstgeest
Tel: 071-5172794 / 0618477590
Mail: cwjansen@wanadoo.nl

Gegevens gastbedrijf:

Naam: Sabern Computer Software BV
Adres: Zijlbaan 28 2353 BN Leiderdorp
Postbus 28 2350 AA Leiderdorp
Tel: 071-5415841
Fax: 071-5421706

Gegevens Haagse Hogeschool:

Naam: Haagse Hogeschool sector Informatica
Adres: Johanna Westerdijkplein 75 2521 EN Den Haag
Post 13336 2501 EH Den Haag
Tel: 070-4458400

Bijlage D - Opgeleverde produkten

- Plan van aanpak + Ontwikkelscenario
- Systeemeisen
- Systeemconcept
- Pilotplan
- Pilotontwikkelplannen
- Pilotontwerpen

Plan van aanpak + Ontwikkelscenario

1. Inleiding

Dit plan van aanpak is bedoeld voor de opdrachtnemer en opdrachtgever. In dit plan van aanpak staat beschreven hoe het project over de uitbereiding van de functionaliteiten van BookDB-XML tot stand zal komen. De opdrachtsomschrijving, afbakening, randvoorwaarden, risicofactoren, wijze van rapporteren, benodigde mensen en middelen, globale planning, detail planning, deadlines en mijlpaalprodukten en kosten/baten analyse komen hier ter sprake. Het ontwikkelscenario is in dit plan van aanpak verwerkt, dat wil zeggen dat de punten die in een ontwikkelscenario horen te staan, ook in dit plan van aanpak naar voren komen.

Dit plan van aanpak is dynamisch, dat wil zeggen dat het steeds (indien nodig) na elke IAD-fase wordt bijgesteld, zodat dit plan van aanpak altijd up-to-date blijft en een steeds stabieler karakter krijgt. Indien er veranderingen in dit plan van aanpak zijn gemaakt is dit duidelijk weergegeven in het laatste hoofdstuk “Aanpassingen”.

2. Opdrachtsomschrijving

In dit hoofdstuk is eerst een korte omschrijving van Sabern BV weergegeven, gevolgd door een korte beschrijving van BookDB-XML en de bijbehorende probleembeschrijving. Hierna volgen de doelstelling, nadrukken en uitgangssituatie. Bij concrete werkzaamheden zijn de activiteiten, methodieken, technieken, talen en de op te leveren produkten uitvoerig weergegeven.

2.1. Omgevingsschets

Sabern Computer Software B.V., opgericht in 1990, assisteert organisaties bij het opzetten van database publishing, content management, XML-applicaties, dynamische Websites en documentenbeheer. Sabern ontwikkelt en levert standaard software en maatwerk toepassingen. Verder geeft Sabern cursussen en verleent consultancy.

2.2. Omschrijving BookDB-XML

BookDB-XML is database publishing software voor het maken van teksten, tabellen, plaatjes en grafieken. Het systeem maakt gebruik van XML voor de documentopmaak. Met BookDB-XML kan een XML document worden gemaakt voor papieren of digitale uitvoer. Met BookDB-XML kunnen de gegevens afkomstig van database management systemen worden gebruikt als basis voor opgemaakte documenten in XML. In het template kunnen de vaste elementen en de opmaakinformatie worden gedefinieerd. In XSLT’s worden de selecties op de database gedefinieerd en wordt de uitvoer opgebouwd.

2.3. Probleembeschrijving

Sabern Computer Software wenst dat de functionaliteit van BookDB-XML uitgebreid wordt. Er ontbreekt nog een aantal functionaliteiten. Deze ontbrekende functionaliteiten dienen te worden geanalyseerd, ontworpen en geïmplementeerd. Dit betreft de volgende punten:

- Finetunen van bestaande functionaliteiten.
- Ontwerpen en implementeren van algemene nieuwe functionaliteiten.
  Denk hierbij aan seriële executie van taken en aan de mogelijkheid om in een template iteratie en selectie op te nemen.
- Het ontwerpen en eventueel implementeren van een grafische user interface voor het samenstellen van templates.

2.4. Doelstelling

De doelstelling van de feitelijke opdracht is de hierboven genoemde functionaliteiten te realiseren, zodat de functionaliteit van BookDB-XML uitgebreid wordt.

2.5. Nadrukken

Het finetunen van de bestaande functionaliteiten en het toevoegen van enkele nieuwe functionaliteiten hebben de hoogste prioriteit. Pas als deze gerealiseerd zijn kan er eventueel verder worden gegaan met het realiseren van een of meer van de overige functionaliteiten.

2.6. Uitgangssituatie

De volgende software, hardware, aanwezige rapporten en ideeën zijn nodig om het project met succes af te ronden.

2.6.1. Software

- Adobe FrameMaker voor het maken van documenten
- Software voor het maken van ontwerpen (Paraben's FlowCharter)
- Een texteditor voor het schrijven en bekijken van de source-code (EditPlus 2)
- Een Java-compiler en runtime-environment
- De meest recente versie van BookDB-XML

2.6.2. Hardware

- Een computer waarop de bovenstaande software draait

2.6.3. Aanwezige rapporten

- Er is documentatie over BookDB-XML, bestaande uit gebruikershandleidingen en systeemdocumentatie in de source-code, beschikbaar.

2.6.4. Ideeën

- Deze komen tot stand naar aanleiding van de definitiestudie en/of pilotontwikkeling

2.7. Concrete werkzaamheden

Hier zijn de activiteiten, methodieken, technieken en talen en de op te leveren produkten weergegeven die voor dit project van toepassing zijn.

2.7.1. Activiteiten

Definitiestudie:

- Plan van aanpak opstellen met een concept-ontwikkelingsscenario erin verwerkt
- Bekend raken met BookDB-XML (eenmalige activiteit)
- Ontwikkelscenario-workshop houden (ontwikkelscenario finetunen met opdrachtgever)
- Pilotplan-workshop houden (systeemeisen, -concept en pilotplan opstellen)

Pilotontwikkeling:

- Pilotontwikkelplan opstellen
- Ontwerpen software-bouweenheden
- Bouwen software-bouweenheden
- Actualiseren gebruikershandleidingen

Invoering:

- Acceptatieworkshop houden
- Pilot(s) overdragen aan opdrachtgever

2.7.2. Methodieken

Bij de uitvoering van de opdracht zal IAD gehanteerd worden. De iteratiestrategie die gebruikt zal worden is een combinatie van incrementeel ontwikkelen en incrementeel opleveren. Voor de door Sabern verplicht gestelde functionaliteiten wordt eenmalig de definitiestudie uitgevoerd, gevolgd door een pilotontwikkeling voor elke pilot. Hierna wordt eventueel voor elke pilot of meerdere pilots tegelijk de invoering uitgevoerd. Voor de overige functionaliteiten wordt nogmaals een definitiestudie uitgevoerd, gevolgd door een pilot-

ontwikkeling voor elke pilot met daarna de invoering voor elke pilot of voor meerdere pilots tegelijk.

2.7.3. Technieken en talen

- Communicatie met de opdrachtgever/projectleider door middel van kleine workshops
- Time-boxing
- Juicy Bits First
- Java
- Extensible Markup Language (XML)
- Extensible Stylesheet Language Transformations (XSLT)
- Unified Modelling Language (UML)

2.7.4. Op te leveren produkten

- Plan van aanpak
- Ontwikkelscenario
- Systeemeisen
- Systeemconcept
- Pilotplan
- Pilotontwikkelplannen
- Ontwerpen van de software-bouweenheden
- Pilots
- Geactualiseerde handleidingen

3. Afbakening

Tijdens dit project zal de opdrachtnemer zich uitsluitend bezig houden met het realiseren van de functionaliteiten zoals genoemd in de probleembeschrijving. Functionaliteiten met de hoogste prioriteit zullen als eerst worden afgehandeld. Aan de hand hiervan kan er worden bepaald óf en in hoeverre er zal worden gewerkt aan de overige functionaliteiten.

4. Randvoorwaarden

Het finetunen van bestaande functionaliteiten en het toevoegen van nieuwe functionaliteiten, dienen tijdens dit project in ieder geval te worden gerealiseerd.

5. Risicofactoren

Risicofactoren kunnen uitmonden in een incident dat de voortgang van het project kan stagneren. Om dit te voorkomen is hieronder weergegeven hoe de gevaren hiervoor geminimaliseerd kunnen worden.

5.1. Gegevensverlies

Omdat er gevaar is op gegevensverlies door het ontstaan van een storing, is het noodzakelijk dat alle belangrijke documenten met betrekking tot dit project dienen te worden opgeslagen op zowel de locale workstation als de backup-server. Van de belangrijkste documenten, zoals het procesverslag, kan ook nog een kopie op de server van de Haagse Hogeschool worden gezet. Hierdoor kan het gevaar op gegevensverlies worden geminimaliseerd.

5.2. Gebrek aan kennis

Indien de voortgang van het project dreigt te stagneren omdat de benodigde kennis niet aanwezig is dient de opdrachtnemer, indien de projecttijd dat toelaat, zichzelf van de benodigde kennis te voorzien door middel van het lezen van documentatie, boeken of het raadplegen van bronnen op het Internet of collega's. Hierdoor wordt het gevaar op stagnatie van het project geminimaliseerd.

5.3. Niet halen van de deadlines

Indien bepaalde onderdelen binnen een bepaalde timebox niet op tijd afkomen, dan mag in geen geval de timebox worden "gerekt", maar dienen deze punten door te schuiven naar de volgende pilot. Indien er indicaties zijn dat het gevaar dreigt dat bepaalde functionaliteiten niet gerealiseerd

kunnen worden binnen de projecttijd, dienen de betrokkenen bij het project hiervan zo spoedig mogelijk op de hoogte te worden gesteld. Er kan dan worden gezocht naar een alternatieve oplossing, zoals het bijspringen van een projectlid die bepaalde taken overneemt. Hierdoor wordt het gevaar op stagnatie van het project geminimaliseerd.

5.4. Te vroeg halen van de deadlines

Indien een deadline te vroeg is gehaald en er tijd overblijft, dan dienen de pilots naar voren opgeschoven te worden. Hierbij dienen de vastgestelde deadlines en timeboxes voor de overige pilots gewoon aangehouden te worden, tenzij anders afgesproken met de projectleider. Hierdoor wordt het gevaar op stagnatie van het project geminimaliseerd.

5.5. Onbereikbaarheid projectleider

Indien de projectleider gedurende een bepaalde tijd niet aanwezig is, dan dient men terug te vallen op de afstudeerbegeleider en/of projectleden voor eventuele vragen en beslissingen. Omdat de projectleider gaat over de beoordeling van de mijlpaalprodukten, dienen de beoordelingsworkshops van de desbetreffende pilots zo snel mogelijk gehouden te worden. Indien er uit de feedback blijkt dat er nog iets moet gebeuren aan een Pilot, dan zal dit worden ingepland in de timebox van de volgende pilot. De opdrachtnemer dient in dit geval door te gaan met de volgende pilot. Hierdoor wordt het gevaar op stagnatie van het project geminimaliseerd.

5.6. Ziekte of andere redenen van afwezigheid

Indien de opdrachtnemer ziek is en hierdoor niet kan functioneren, zullen de dagen dat de opdrachtnemer afwezig is worden ingehaald indien dat mogelijk is binnen de door school opgestelde projecttijd. Indien de opdrachtnemer om een andere reden niet in staat is om het kantoor van Sabern BV te bereiken maar toch in staat is te werken, dient er de mogelijkheid te zijn om thuis te werken. Hierdoor wordt het gevaar op stagnatie van het project geminimaliseerd.

6. Wijze van rapporteren

De opdrachtgever wordt met behulp van workshops en informele gesprekken op de hoogte gehouden van de vorderingen van het project. Dit dient op een dusdanige wijze te gebeuren dat alle betrokkenen optimaal op de hoogte zijn van de vorderingen van het project.

7. Benodigde mensen en middelen

7.1. Mensen

- De bedrijfsmentor: K. Lelieveld
- De projectleider: F. de Paus
- Projectlid: G. van der Laan
- De opdrachtnemer: Christian Jansen

7.2. Middelen

- Zie benodigde software/hardware in de uitgangssituatie van Opdrachtsomschrijving

8. Globale planning

Globaal gezien kan het gehele project worden ingedeeld in twee aparte delen:

- De verplichte functionaliteiten analyseren, ontwerpen en implementeren
- De optionele functionaliteiten analyseren, ontwerpen en implementeren

Omdat het eerste deel het belangrijkst is en binnen de vastgestelde projecttijd (15 weken) gerealiseerd dient te zijn, zal eerst alle aandacht op dat deel worden gericht. Indien er daarna nog tijd over is, kan er pas worden begonnen met het tweede deel. De details voor dit deel zullen daarom ook pas nadat het eerste deel is gerealiseerd, worden onderzocht.

8.1. Verplichte functionaliteiten

Om de verplichte functionaliteiten te realiseren zal de combinatie van de strategieën “incrementeel opleveren” en “incrementeel ontwikkelen” worden gebruikt. De fase definitiestudie zal éénmalig worden uitgevoerd, waarna er voor elke pilot de fase pilotontwikkeling en voor elke pilot of meerdere pilots tegelijk een invoering zal worden doorlopen.

8.1.1. Definitiestudie (éénmalig):

Als eerst wordt er een algemeen plan van aanpak opgesteld, waarin alle afspraken betreffende het project staan. Dit plan van aanpak zal indien dat nodig is steeds worden bijgesteld naarmate het project vordert. Hierdoor krijgt het plan van aanpak een steeds stabieler karakter. De opdrachtnemer zal zelf een ontwikkelscenario opstellen en verwerken in het plan van aanpak. In het ontwikkelscenario zal de aanpak, structuur, besturing, doelstellingen en richting van het project in beeld worden gebracht. Het ontwikkelscenario dient ervoor om alle betrokkenen inzicht te geven in hoe het project is gestructureerd. Het plan van aanpak en ontwikkelscenario zal daarna in een korte ontwikkelscenario-workshop worden geëvalueerd met de opdrachtgever/projectleider en zonodig worden bijgesteld. Het opstellen van het plan van aanpak en het ontwikkelscenario dient in de tweede week van de projecttijd klaar te zijn.

Vervolgens zal de opdrachtnemer ervaring opdoen met BookDB-XML. Dit is tevens een eis van de opdrachtgever. Dit zal bestaan uit het maken van een catalogus met behulp van BookDB-XML. Ook het bekijken van de source-code zal het inzicht van de opdrachtnemer vergroten betreffende de software. Het bekend raken met BookDB-XML dient in de derde week van de projecttijd voltooid te zijn.

Hierna zal een pilotplan-workshop worden gehouden, welke zal bestaan uit een uitgebreid gesprek met de opdrachtgever. Tijdens deze workshop zullen de systeemeisen volledig in kaart worden gebracht, gecategoriseerd en geprioriteerd. Hierna zal een gedetailleerde functionele oplossing worden opgesteld in de vorm van een systeemconcept. Vervolgens zal er een pilotplan worden opgesteld waarin in detail wordt beschreven volgens welke pilots het informatiesysteem tot stand zal komen. Het systeemconcept en pilotplan dienen aan het eind van de derde week van het project klaar te zijn.

8.1.2. Pilotontwikkeling (per pilot):

Vervolgens wordt voor elke pilot de fase pilotontwikkeling doorlopen. In deze fase wordt het plan van aanpak bijgewerkt en zal de pilot zal worden opgedeeld in pilotdelen welke achter elkaar zullen worden ontwikkeld. Hiervan zal een pilotontwikkelplan worden opgesteld. Er zal hierbij gebruik worden gemaakt van time-boxes waarbinnen de pilots en pilotdelen dienen te worden gerealiseerd. De indeling van de pilotdelen gebeurd met behulp van de Juicy Bits First strategie.

Hierna zal de pilot daadwerkelijk worden ontworpen en gebouwd, waarna de bouweenheden worden geïntegreerd. Ook het eventueel wijzigen van de handleidingen hoort hierbij.

8.1.3. Invoering (per pilot):

Tenslotte volgt de invoering. Deze fase zal in principe voor elke pilot worden uitgevoerd, tenzij de projectleider anders beslist. Tijdens deze fase zal een kleine acceptatieworkshop plaatsvinden waarna de pilot worden opgeleverd aan de projectleider.

8.2. Optionele functionaliteiten

De ontwikkelingsstrategie van de optionele functionaliteiten zal bestaan uit een combinatie van de strategieën “incrementeel opleveren” en “incrementeel ontwikkelen”. De fase definitiestudie zal éénmalig worden uitgevoerd, waarna er voor elke pilot de fase pilotontwikkeling en voor elke pilot of meerdere pilots tegelijk een invoering zal worden doorlopen. Hierbij zullen de delen uit die vorige definitiestudie die hergebruikt kunnen worden, ook daadwerkelijk worden hergebruikt. Een ontwikkelscenarioworkshop zal hier niet worden gehouden, omdat al vast staat hoe dit gedeelte van het project doorlopen zal worden.

9. Detail planning

Hier staat de voorlopige (gedetailleerde) planning zoals waaraan de opdrachtnemer zich tijdens dit project aan dient te houden. De activiteiten betreffende de pilots zijn weergegeven met het nummer van de desbetreffende pilot (1 t/m 8). Voor het gehele project is in totaal 15 weken beschikbaar gesteld. Deze planning zal, naarmate het project vordert, steeds worden bijgesteld mocht dat nodig zijn. Hier zijn de activiteiten tegen de tijd in weken uitgezet.

Detail planning

	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15
A	x														
B	x	x	x												
C		x													
D		x						x	x			x			
E			1	23			4	56	7			8			
F			1	23	3		4	56	7	7		8			
G			1	2	3	3	34	5	6		7		8	8	8
H				12			34	5			7				8
I							123	45			7				8
J							123	45			7				8
K	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
L					1		2		6	3		47		5	8

A. Opstellen plan van aanpak en ontwikkelscenario
B. Bekend raken met BookDB-XML
C. Ontwikkelscenario-workshop houden + ontwikkelscenario/plan van aanpak aanpassen
D. Pilotplan-workshop houden + systeemeisen, systeemconcept en pilotplan opstellen
E. Pilotontwikkelplan opstellen
F. Ontwerpen software-bouweenheden
G. Bouwen software-bouweenheden
H. Actualiseren handleidingen
I. Acceptatieworkshop houden
J. Pilot(s) opleveren aan projectleider
K. Schrijven procesverslag
L. Deadlines voor pilots

Totaal: 15 weken / 75 dagen / 600 uur

10. Deadlines en mijlpaalprodukten

10.1. Deadlines

- Pilot I uiterlijk vrijdag week 5 (11-3) opleveren aan projectleider
- Pilot II uiterlijk vrijdag week 7 (25-3) opleveren aan projectleider
- Pilot III uiterlijk vrijdag week 10 (15-4) opleveren aan projectleider
- Pilot IV uiterlijk vrijdag week 12 (29-4) opleveren aan projectleider
- Pilot V uiterlijk vrijdag week 14 (13-5) opleveren aan projectleider

- Pilot VI uiterlijk vrijdag week 9 (8-4) opleveren aan projectleider
- Pilot VII uiterlijk vrijdag week 12 (29-4) opleveren aan projectleider
- Pilot VIII uiterlijk vrijdag week 15 (20-5) opleveren aan projectleider

10.2. Mijlpaalprodukten

Het behalen van deze mijlpaalprodukten is een cruciale succesfactor voor het project.

- Pilot Ia - Variabelen bij naam benoemen
- Pilot Ib - Database-selectie SQLSingleResult en SQLAggregatedResult
- Pilot II - Alternatieve oplossing SQLLoops
- Pilot IIIa - (Fout)meldingen beter afvangen en tonen
- Pilot IIIb - Seriele executie van taken
- Pilot IV - Mechanisme voor algemene loop
- Pilot V - Mechanisme voor headers en footers voor gebalanceerde tabellen
- Pilot VI - Workaround/Oplossing voor Java-bug
- Pilot VII - IF/THEN/ELSE-statement voor in template
- Pilot VIII - Grafical User Interface voor genereren van templates

11. Kosten/baten analyse

11.1. Kosten opdrachtnemer

- De opdrachtnemer zal in totaal 15x40=600 uur besteden aan het project. Hierbij wordt van hem verwacht dat hij zich volledig zal inzetten voor het project.

11.2. Kosten opdrachtgever

- Een vergoeding van ongeveer 230 euro per maand aan de opdrachtnemer
- Voorzieningen voor de opdrachtnemer om het project succesvol af te ronden
- Beschikbaar zijn voor begeleiding van de opdrachtnemer en gesprekken (workshops)

12. Aanpassingen

Eventuele aanpassingen in het plan van aanpak worden hier vermeld.

- 07-02-2005 - Plan van aanpak gegenereerd en voorlopig ontwikkelscenario hierin verwerkt.
- 16-02-2005 - Plan van aanpak gewijzigd na ontwikkelscenario-workshop en pilotplan-workshop. Planning verder gedetailleerd.
- 28-02-2005 - Plan van aanpak gewijzigd. Risicofactoren gewijzigd.
- 30-03-2005 - Plan van aanpak gewijzigd. Planning gewijzigd na realisatie van Pilot V.
- 31-03-2005 - Plan van aanpak gewijzigd na pilotplanworkshop.
- 01-04-2005 - Plan van aanpak gewijzigd na definieren definitieve opdrachtsomschrijving.
- 20-05-2005 - Plan van aanpak gewijzigd na realisatie Pilot VIII. Planning ge-update met gerealiseerde tijdsindeling.

Systeemeisen

Dit document beschrijft de systeemeisen voor BookDB-XML die gerealiseerd dienen te worden. Deze systeemeisen zijn hier gecategoriseerd en in volgorde van prioriteit weergegeven.

1.1 Variabelen bij naam kunnen benoemen

Momenteel is het alleen mogelijk om variabelen met een nummer aan te roepen. Het moet nu ook mogelijk zijn om variabelen te kunnen benoemen met een naam, zodat de gebruiker deze variabelen een duidelijke naam kan geven en daarmee weet met welke variabelen hij op dat moment werkt als hij of zij de template samenstelt. Hierdoor wordt eventuele verwarring vermeden.

1.2 Database-selectie SQLSingleResult en SQLAggregatedResult

Het moet mogelijk zijn om bij de functies SQLSingleResult en SQLAggregatedResult de te gebruiken database mee te geven als parameter via een XML-attribuut, zodat deze functies kunnen worden gebruikt indien er meer dan één database wordt gebruikt voor invoer van BookDB-XML.

2. Alternatieve oplossing voor SQLLoops

Momenteel past BookDB-XML de XSL bookdbxml.xsl net zo lang toe tot er geen SQLLoop statements meer in het tussenresultaat voorkomen. Daarna wordt er een XSL bookdbxml_clean.xsl uitgevoerd om tijdelijke elementen te verwijderen. Er dient een aparte XSL te komen die alleen de SQLLoop statements verwerkt en deze net zo lang toepast tot er geen SQLLoop statements meer zijn, om vervolgens de echte XSL toe te passen. Hierdoor wordt vermeden dat BookDB-XML onnodig veel loops maakt en soms zaken onterecht dubbel toepast.

3.1 (Fout)meldingen afvangen en tonen

De gebruiker moet goed op de hoogte worden gesteld van eventuele fouten die optreden bij het genereren van een XML-bestand. Momenteel worden foutmeldingen in het DOS-venster of als dialog-box aan de gebruiker gepresenteerd. Deze foutmeldingen bevatten echter weinig relevante informatie. Er dient een mogelijkheid te komen waarmee de gebruiker goed op de hoogte wordt gesteld van fouten. Het moet de gebruiker dan duidelijk zijn wat er precies mis is gegaan en hoe het kan worden voorkomen.

3.2 Seriële executie van taken

Er moet een mogelijkheid komen om meerdere taken automatisch achter elkaar uit te laten voeren. De gebruiker moet de mogelijkheid krijgen om meerdere templates te selecteren (of een selectie hiervan) en in een door hem of haar bepaalde volgorde uit te laten voeren. Hierbij moeten ook de gewenste parameters kunnen worden meegegeven. Omdat de gebruiker anders alle templates een voor een zou moeten afhandelen, bespaart dit tijd.

4. Mechanisme voor algemene loop

Er dient een mechanisme te komen waarmee de gebruiker vanuit een template een loop uit kan laten voeren, waarbij de gebruiker zelf bepaald hoeveel keer de loop moet worden uitgevoerd. De waarde van deze loopvariabelen dienen dan vervolgens te kunnen worden opgevraagd.

5. Mechanisme voor headers en footers voor gebalanceerde tabellen

Er is een opzet gemaakt om met BookDB-XML gebalanceerde tabellen te kunnen genereren. Dit mechanisme moet verder worden uitgewerkt, zodat aan deze tabellen ook headers en footers kunnen worden toegekend.

6. Oplossing of workaround voor Java-bug

Zodra er een Scrollable Java ResultSet wordt gebruikt met een query waarin het keyword DISTINCT voorkomt, dan wordt het aantal results onjuist doorgegeven. Hierdoor maakt BookDB-XML onterecht teveel loops. Het is de bedoeling dat het aantal results juist moet worden doorgegeven, zodat BookDB-XML geen onnodige loops meer maakt. Indien er voor deze Java-bug nog geen

oplossing is, dient hier een workaround voor te worden ontworpen en geïmplementeerd, mits dit mogelijk is binnen vijf werkdagen.

7. IF/THEN/ELSE statements in template

Het moet mogelijk zijn voor de gebruiker om IF/THEN/ELSE statements te gebruiken tijdens het maken van een template om een selectie te maken, zodat er aan de hand van waarden uit de database beslissingen genomen kunnen worden die effect hebben op het eindresultaat.

8. GUI om een template mee te genereren

Momenteel is het alleen mogelijk voor de gebruiker om de template zelf te schrijven. Het moet mogelijk zijn voor de gebruiker om een template samen te stellen met behulp van een graphical user-interface. Het is de bedoeling hierbij dat het opstellen van een template voor de gebruiker makkelijker wordt gemaakt. Dit kan bereikt worden door de gebruiker te helpen met het opstellen van een template. Hierbij kan worden gedacht aan het gedeeltelijk automatisch genereren van XML-elementen en database-connecties. Er dient een ontwerp te komen van deze functionaliteit. Indien de tijd het toelaat, kan de ontworpen functionaliteit ook gedeeltelijk of volledig geïmplementeerd worden.

Systeemconcept

In dit systeemconcept worden de te realiseren functionaliteiten vanuit het oogpunt van de gebruiker weergegeven. Er is per functionaliteit weergegeven wat de huidige situatie is, de gebeurtenis die reden is voor het gebruiken van de nieuwe functionaliteit (trigger), de benodigde informatie die nodig is om de functionaliteit goed te laten werken en het beoogde resultaat van de functionaliteit. Elke in dit document genoemde functionaliteit is een mijlpaal.

Voor alle functionaliteiten die beschreven staan in dit Systeemconcept (behalve nummer 8), is er slechts één actor: de gebruiker van BookDB-XML, tenzij anders aangegeven.

1. Variabelen bij naam kunnen benoemen

1.1 Huidige situatie

De gebruiker van BookDB-XML kan nu variabelen aanmaken bij het samenstellen van zijn of haar template met de functie “initvar”. Dit doet de gebruiker door de waarde van de variabele op te geven en hier een nummer aan toe te kennen waarmee de variabele geïdentificeerd kan worden. Naar deze variabelen kan momenteel alleen worden verwezen met een nummer. Dit is kan voor de gebruiker erg onhandig zijn als er meerdere variabelen worden gebruikt. De gebruiker moet dus onthouden bij welke variabele welk nummer hoort.

1.2 Trigger

De gebruiker van BookDB-XML wil een variabele aanmaken en deze aanroepen met een voor hem of haar duidelijke naam, zodat de gebruiker weet naar welke variabele deze naam verwijst.

1.3 Benodigde informatie

Om de variabele een naam te kunnen geven is de volgende informatie nodig:

- De naam van de variabele óf het nummer van de variabele
- De waarde van de variabele

1.4 Beoogd resultaat

De gebruiker kan nu duidelijk naar de variabele verwijzen door er een naam aan toe te kennen. Doordat de gebruiker deze naam zelf kan specificeren zal hij of zij precies weten welke variabele aan welke naam verbonden is. Hierdoor wordt de kans op verwarring geminimaliseerd.

2. Database-selectie SQLSingleResult en SQLAggregatedResult

2.1 Huidige situatie

De functies SQLSingleResult en SQLAggregatedResult kunnen door de gebruiker van BookDB-XML gebruikt worden tijdens het opstellen van zijn of haar template om respectievelijk één resultaat en het aantal resultaten van een bepaalde query uit de database te halen. Momenteel gebruiken deze functies alleen de standaard database die is aangegeven met de DBINFO-functie aan het begin van de template. De gebruiker roept de functies aan met de query en naam van het resultaat als parameters.

2.2 Trigger

De gebruiker van BookDB-XML wil SQLSingleResult en SQLAggregatedResult gebruiken om resultaten te verkrijgen uit een andere database, dus niet de standaard database die gedefinieerd is met behulp van de DBINFO-functie aan het begin van de template.

2.3 Benodigde informatie

Om de functies SQLSingleResult en SQLAggregatedResult te kunnen gebruiken heeft de gebruiker de volgende informatie nodig:

- De query om de resultaten uit de database te halen
- De naam van de resultset
- De naam van de te gebruiken database (optioneel)

Deze gegevens worden meegegeven aan de functie als parameters. De eerste twee parameters zijn verplicht. De naam van de te gebruiken database is optioneel. Als deze niet is ingevuld zal de query worden uitgevoerd op de standaard database.

2.4 Beoogd resultaat

Zodra de gebruiker van BookDB-XML de SQLSingleResult en SQLAggregatedResult gebruikt met de optionele parameter voor de te gebruiken database, zal de door de gebruiker opgegeven query worden uitgevoerd op de door hem of haar gespecificeerde database. Het resultaat zal dan worden gebruikt door BookDB-XML om het uiteindelijke XML-bestand mee te genereren.

3. Alternatieve oplossing voor SQLLoops

3.1 Huidige situatie

Momenteel is het zo dat de XSL “bookdbxml.xsl” net zo lang op het template wordt toegepast totdat er geen SQLLoop-statements meer in het tussenresultaat voorkomen. De tussenresultaten worden opgeslagen in de bestanden StepN.xml, waarvan N het nummer van de diepte van de geneste SQLLoop is. Hierna wordt de XSL “bookdbxml_clean.xml” toegepast om alle tijdelijke elementen te verwijderen. In deze huidige situatie worden er onnodig veel loops gemaakt, waardoor zaken onterecht dubbel toegepast worden. Als de gebruiker van BookDB-XML bijvoorbeeld een euroteken voor elke artikelprijs wil hebben in de catalogus die hij of zij aan het genereren is, kan de gebruiker dit aangeven in de XSL “bookdbxml.xsl” of in een apart XSL-bestand. Beide worden echter even vaak uitgevoerd (afhankelijk van het aantal SQLLoops), waardoor er meerdere eurotekens voor de artikelprijs komen te staan. Dit is niet de bedoeling.

3.2 Trigger

De gebruiker van BookDB-XML wil in de catalogus die hij of zij aan het genereren is, voor elke artikelprijs één euro-teken hebben.

3.3 Benodigde informatie

- Een additionele XSL om de statements die éénmalig dienen te worden uitgevoerd te definiëren, in dit geval voor elke artikelprijs het euroteken.

3.4 Beoogd resultaat

De catalogus wordt gegenereerd waarin voor elke artikelprijs éénmaal het euroteken staat. Zaken die maar éénmalig uitgevoerd dienen te worden, zullen ook maar éénmalig uitgevoerd worden.

4. (Fout)meldingen afvangen en tonen

4.1 Huidige situatie

Momenteel is het zo dat als er iets fout gaat tijdens de executie van de door de gebruiker opgestelde template, de gebruiker nauwelijks op de hoogte wordt gesteld van wat er precies misgaat en hoe het eventueel voorkomen zou kunnen worden.

4.2 Trigger

Er gaat iets mis tijdens de executie van de template en de gebruiker wil weten wat er precies misgaat, zodat deze te weten komt waar de fout zit en hoe deze eventueel voorkomen zou kunnen worden.

4.3 Benodigde informatie
Het is voor de gebruiker niet nodig om hier informatie aan het systeem toe te voegen.

4.4 Beoogd resultaat
Zodra er een fout optreedt tijdens de executie van de template, zullen de foutmeldingen in een chronologische volgorde worden weergegeven in een apart log-tabblad in de grafische user-interface. Ook zal worden weergegeven waar de fout precies zit en hoe de fout eventueel voorkomen kan worden.

5. Seriele executie van taken

5.1 Huidige situatie
Indien de gebruiker van BookDB-XML meerdere templates wil executeren (voor bijvoorbeeld het in stappen opzetten van een boek), zal de gebruiker deze templates één voor één uit moeten voeren. Dit is een tijdrovende klus. Het is echter wel mogelijk een batchbestand aan te maken die achter elkaar de templates uitvoert door de inputfile, outputfile, parameters, XSL's en splitoption als parameters mee te geven aan het commandversie van BookDB-XML. Hierdoor wordt BookDB-XML steeds weer opnieuw opgestart. Zowel het aanmaken van het batch-bestand met de daarbij behorende parameters, als het steeds weer opnieuw opstarten van BookDB-XML is erg tijdrovend.

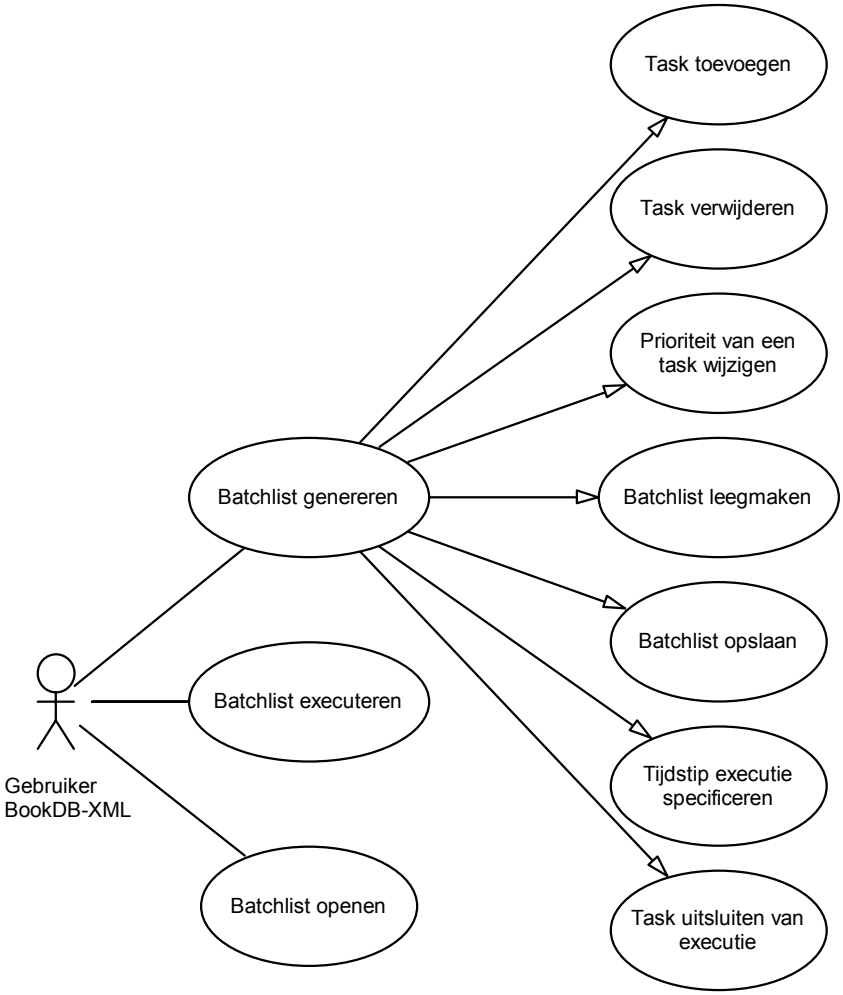
5.2 Trigger
De gebruiker wil op een makkelijke manier meerdere templates met de bijbehorende inputfile, outputfile, parameters, XSL's en splitoption van deze templates in een door hem of haar bepaalde volgorde laten executeren.

5.3 Benodigde informatie
Om meerdere templates te laten executeren, is per template de volgende informatie nodig. Deze informatie kan dan in de vorm van een "task" worden toegevoegd aan een lijst (batchlist) waarin alle tasks staan die uitgevoerd dienen te worden.

- De naam van de template (input-file)
- De naam van de output-file
- De eventueel mee te geven parameters
- De eventuele XSL's
- De eventuele splitoptie
- De naam waarmee de template in de batchlist kan worden geïdentificeerd
- De volgorde waarin de gebruiker de templates wil executeren
- Eventueel het tijdstip waarop de executie dient plaats te vinden *
- Eventueel een template uitsluiten van executie (uitvinken in de batchlist) *

De twee punten met een * hebben een lage prioriteit en zullen niet worden uitgewerkt.

Een task is één rij in de batchlist en bestaat uit de executie van één template met de daarbij behorende inputfile, outputfile, parameters, XSL's en splitoption. Zodra een task wordt toegevoegd aan de batchlist worden deze instellingen overgenomen.
In het Use-case diagram is te zien welke acties de gebruiker uit moet kunnen voeren. (Het genereren van de task zelf valt buiten het domein van deze pilot en is immers al mogelijk. Daarom is dit niet opgenomen in het Use-case diagram.)



De gebruiker van BookDB-XML moet de volgende acties uit kunnen voeren:

- Batchlist genereren:
    - Task toevoegen
      Zodra de gebruiker een task heeft gespecificeerd (dit valt buiten deze pilot en is al mogelijk met het huidige systeem), kan de gebruiker deze task een naam geven en toevoegen aan de batchlist.
    - Task verwijderen
      De gebruiker kan een task in de batchlist selecteren en met een druk op een knop verwijderen uit de batchlist.
    - Prioriteit van een task wijzigen
      De gebruiker moet de prioriteit van een task kunnen wijzigen, om hiermee de volgorde van executie te bepalen. Dit kan door de positie van de task in de batchlist te wijzigen.
    - Batchlist leegmaken
      De gebruiker moet de mogelijkheid hebben om met een druk op een knop de volledige batchlist leeg te maken, zodat hij of zij een nieuwe batchlist kan maken.
    - Batchlist opslaan
      De gebruiker moet de mogelijkheid hebben om een gemaakte batchlist op te

slaan, zodat deze op een later moment geexecuteerd of aangepast kan worden.

- Tijdstip executie specificeren
  Het moet mogelijk zijn voor de gebuiker om de batchlist op een door hem of haar te bepalen tijdstip te executeren. Dit kan door het specificeren van een tijdstip.
  (Deze functionaliteit heeft een lage prioriteit en zal daarom niet worden uitgewerkt.)

- Task uitsluiten van executie
  Het moet mogelijk zijn voor de gebruiker om een task in de batchlist uit te sluiten van executie, door deze bijvoorbeeld uit te vinken. De task blijft echter wel in de lijst staan, maar wordt niet geexecuteerd.
  (Deze functionaliteit heeft een lage prioriteit en zal daarom niet worden uitgewerkt.)

- Batchlist executeren:
  De gebruiker kan de door hem gegenereerde batchlist met een druk op een knop laten executeren.

- Batchlist openen:
  De gebruiker kan een eerder opgeslagen batchlist openen, om vervolgens aan te passen of te executeren.

5.4 Beoogd resultaat
De geselecteerde templates die in de batchlist zijn gezet (met bijbehorende outputfile, parameters, XSL's en splitoption) in de vorm van een "task", worden in een door hem of haar bepaalde volgorde serieel geëxecuteerd.

6. Mechanisme voor algemene loop

6.1 Huidige situatie
Momenteel bestaat er nog geen mogelijkheid voor de gebruiker om zelf te bepalen hoe vaak een loop moet worden uitgevoerd.

6.2 Trigger
De gebruiker wil in zijn of haar template een loop uitvoeren waarbij de gebruiker zelf bepaald hoe vaak deze loop moet worden uitgevoerd en voor welke variabelen deze loop moet gelden.

6.3 Benodigde informatie
Om deze loop uit te laten voeren is de volgende informatie nodig:

- Het aantal keer dat de loop uitgevoerd dient te worden
- De naam of het nummer van de variabelen waarin de uitkomst van de loop dient te worden gezet

Deze gegevens worden samen als één parameter meegegeven aan de loop-functie.

6.4 Beoogd resultaat
De loop wordt net zo vaak uitgevoerd als dat de gebruiker wil, waarbij de waarden van de uitkomst van deze loop in de door de gebruiker gespecificeerde variabelen worden gezet.

7. Mechanisme voor headers en footers voor gebalanceerde tabellen

7.1 Huidige situatie
Momenteel bestaat er de mogelijkheid voor de gebruiker om gebalanceerde tabellen te kunnen genereren vanuit de template. Echter bestaat de mogelijkheid niet om aan deze tabellen een header- en footerrij toe te voegen.

7.2 Trigger
De gebruiker wil een gebalanceerde tabel genereren en hierbij een header en footer tonen, zodat er additionele informatie aan de tabel kan worden toegevoegd.

7.3 Benodigde informatie
Om een gebalanceerde tabel met header en footer te genereren is de volgende informatie nodig:

- Inhoud van de tabel
- Inhoud van de header en of footer

7.4 Beoogd resultaat
Door het aanroepen van de functie BalancedTable in de template, kan de gebruiker deinhoud van de tabel als parameter meegeven, samen met de inhoud van de header en of footer. BookDB-XML zal de template vervolgens executeren en de tabel met header en of footer genereren.

8. Oplossing of workaround voor Java-bug

8.1 Huidige situatie
Momenteel is het zo dat als een SQL-statement met een DISTINCT keyword erin wordt uitgevoerd en in een scrollable ResultSet wordt gezet, het aantal results niet klopt. Een SELECT-statement levert bijvoorbeeld 20 resultaten op. Zodra er een DISTINCT-keyword voorkomt (waardoor er alleen unieke resultaten uitkomen), horen dit er bijvoorbeeld 14 te zijn. Echter komen er in de scrollable ResultSet toch 20 te staan, met in de eerste 14 velden de juiste waarden, de overige 6 waarden zijn dan NULL. Hier schuilt het probleem. Het is onduidelijk of deze NULL-waarden uit de database komen of dat dit gewoon lege waarden zijn in de scrollable ResultSet. Hierdoor maakt BookDB-XML onterecht teveel loops als de resultaten worden gebruikt binnen een SQLLoop of FORLoop.

8.2 Trigger
De gebruiker van BookDB-XML wil aan de hand van de uitkomsten van een query met een DISTINCT-keyword een loop een aantal keren herhalen.

8.3 Benodigde informatie
- De uit te voeren SQL-query met DISTINCT-keyword

8.4 Beoogd resultaat
Nadat de gebruiker de template met daarin de SQL-query met DISTINCT-keyword heeft uitgevoerd, zal de scrollable ResultSet het juiste aantal results bevatten waardoor de SQLLoop of FORLoop correct wordt uitgevoerd.

9. IF/THEN/ELSE statements in template

9.1 Huidige situatie
Het is momenteel voor de gebruiker niet mogelijk door aan de hand van waarden uit de database beslissingen te nemen die van invloed zijn de op de output.

9.2 Trigger
De gebruiker wil aan de hand van waarden uit de database een selectie uitvoeren. Van elke in de database voorkomende persoon wordt de volledige naam en leeftijd weerge-

geven. Alleen als een persoon de achternaam “Jansen” heeft, komt er een footnote bij te staan met daarin: “Jansen is een veel voorkomende achternaam in Nederland!”. Indien de achternaam niet Jansen is, komt er niets in de output te staan. Indien de persoon een leeftijd heeft van 18 jaar of hoger, dan komt er in de output: “Mag zijn of haar rijbewijs halen!”. Indien dit niet zo is, komt er in de output te staan: “Mag nog geen auto rijden!”.

9.3 Benodigde informatie
Het IF/THEN/ELSE statement kan worden aangeroepen met het <BDB_IF>-element. Dit <BDB_IF>-element kan de volgende drie sub-elementen bevatten:

<BDB_CONDITION>
In dit element wordt de conditie geplaatst waarop getoetst gaat worden. Dit element is verplicht. Binnen dit element kunnen één of meer van de volgende operators worden gebruikt:

Boolean-operators:

- <BDB_AND>
  Alle waarden moeten waar zijn

- <BDB_OR>
  Minstens één waarde moet waar zijn

- <BDB_NOT>
  Kan worden gebruikt om de tegenovergestelde uitkomst van een BDB_AND of BDB_OR operator te selecteren.

Value-operators:

- <BDB_EQUAL>
  De waarden moeten gelijk zijn

- <BDB_NOT_EQUAL>
  De waarden moeten niet gelijk zijn

- <BDB_GREATER>
  De eerst waarde moet groter zijn dan de tweede waarde

- <BDB_SMALLER>
  De eerste waarde moet kleiner zijn dan de tweede waarde

- <BDB_GREATER_EQUAL>
  De eerste waarde moet groter of gelijk zijn aan de tweede waarde

- <BDB_SMALLER_EQUAL>
  De eerste waarde moet kleiner of gelijk zijn aan de tweede waarde

  Binnen deze operators kunnen de variabelen waarop getoetst dient te worden, worden aangegeven binnen <BDB_PART>-elementen. Dit kunnen vaste waarden zijn die door de gebruiker zijn gespecificeerd, maar ook bijvoorbeeld een SQLField, SQLSingleResult of SQLAggregatedResult.

<THEN>
In dit element kan een gedeelte van de template worden geplaatst dat uitgevoerd dient te worden op het moment dat de uitkomst van het <BDB_CONDITION>-element waar (true) is. Dit element is verplicht en dient altijd voor te komen in een BDB_IF-statement.

<ELSE>
In dit element kan een gedeelte van de template worden geplaatst dat uitgevoerd dient te worden op het moment dat de uitkomst van het <BDB_CONDITION>-element niet waar (false) is. Dit element is optioneel.

9.4 Beoogd resultaat
Het resultaat van het voorbeeld zou in dit geval kunnen zijn:

<Name>Christian Jansen</Name>
<Age>23</Age>
<Footnote>Jansen is een veel voorkomende achternaam in Nederland!</Footnote>
<Footnote>Mag zijn of haar rijbewijs halen!</Footnote>

<Name>Pietje Puk</Name>
<Age>16</Age>
<Footnote>Mag nog geen auto rijden!</Footnote>

<Name>Popi de Clown</Name>
<Age>32</Age>
<Footnote>Mag zijn of haar rijbewijs halen!</Footnote>

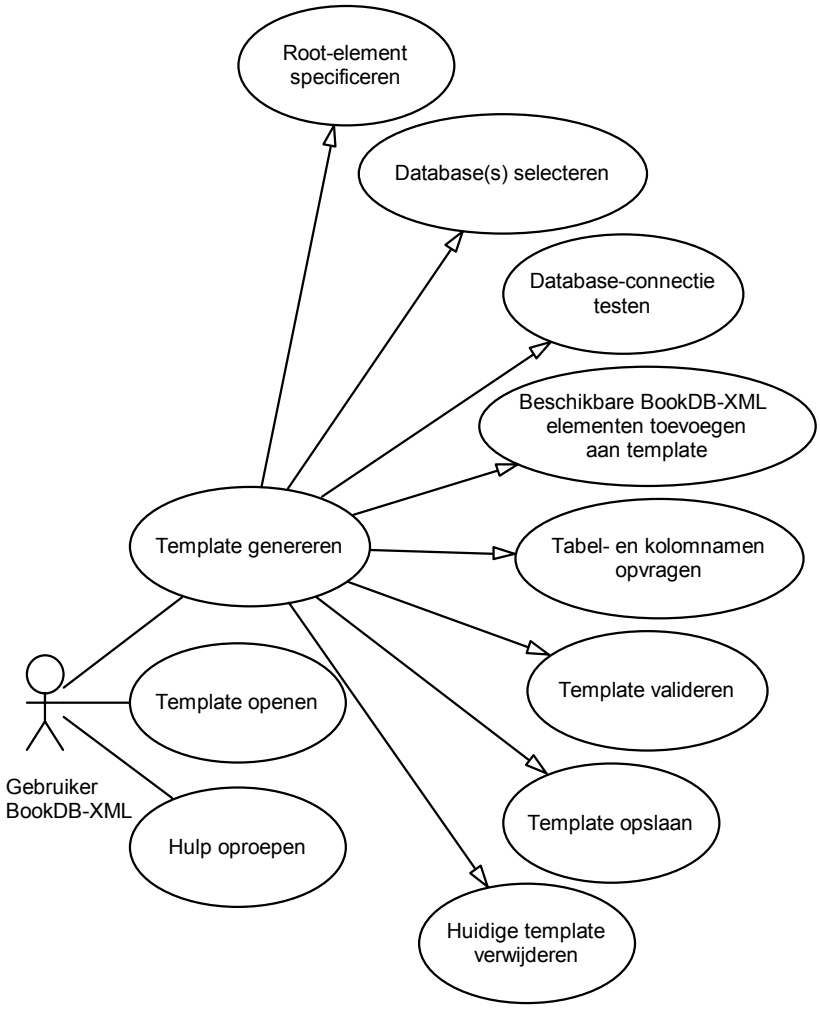
9.5 Aandachtspunten
- De IF/THEN/ELSE-statements moeten zowel numeriek als alfanumeriek werken
- Op een NULL-waarde uit de database zou ook getest moeten kunnen worden
- Voordat het statement wordt uitgevoerd, moet er een check worden gedaan op de syntax omdat essentiële elementen die nodig zijn voor het IF/THEN/ELSE-statement niet mogen missen.
- Het IF/THEN/ELSE-statement moet gebruikt kunnen worden binnen een SQLLoop.

10. GUI om een template mee te genereren

10.1 Huidige situatie
Momenteel heeft de gebruiker van BookDB-XML alleen de mogelijkheid om zelf een template te maken in een text-editor. Dit is voor veel gebruikers erg lastig, omdat zij zelf de template volledig moeten genereren. Hierbij komt het vaak voor dat de gebruiker een fout maakt, met als gevolg dat de template niet uit valide XML bestaat of dat de template niet juist is opgebouwd. Om deze redenen dient er een mogelijkheid te komen om het genereren van een template te automatiseren op plaatsen waar dat mogelijk is, zodat het opstellen van een template voor de gebruiker wordt vergemakkelijkt en er dus ook minder vaak fouten kunnen worden gemaakt.

10.2 Trigger
De gebruiker wil op een gebruiksvriendelijke manier een template samenstellen zonder dat hij of zij hierbij de template helemaal zelf hoeft te schrijven.

10.3 Benodigde informatie
Om een template te maken die als resultaat een XML-bestand wat voldoet aan de eisen van de gebruiker, is nodig om te weten hoe het eindresultaat er precies uit dient te gaan zien. Indien de gebruiker dat weet, kan hij of zij de volgende acties uitvoeren:



De gebruiker van BookDB-XML moet de volgende acties uit kunnen voeren:

- Template genereren:
    - Root-element specificeren
      Zodra de gebruiker begint aan het opstellen van een template, zal hij of zij het root-element moeten specificeren, dit is namelijk het verplichte element waarin de template zich zal bevinden.
    - Database-connectie testen
      Om er zeker van te zijn dat een database-connectie succesvol wordt opgezet, moet er de mogelijkheid zijn om deze database-connectie te testen.
    - Database(s) selecteren
      Om toegang te hebben tot de data die nodig is voor het uiteindelijke XML-bestand, zullen er één of meerdere databases worden gebruikt. Het moet voor de gebruiker mogelijk zijn om een connectie te maken met deze database(s).
    - Beschikbare BookDB-XML elementen toevoegen aan template
      Het moet mogelijk zijn voor de gebruiker om te kiezen uit beschikbare BookDB-XML elementen om deze toe te voegen aan de template.

    - Tabel- en kolomnamen opvragen
      Het moet voor de gebruiker mogelijk zijn om de tabel- en kolomnamen te weten die in de databases voorkomen, zodat de SQL-queries dan makkelijker en sneller kunnen worden opgesteld.

    - Template valideren
      Om er zeker van te zijn dat een template in valide XML is gegenereerd, moet de gebruiker de mogelijkheid hebben om een check uit te voeren op de template.

    - Template opslaan
      De gebruiker moet een template kunnen opslaan, voor eventuele latere executie of bewerking.

    - Huidige template verwijderen
      Indien de gebruiker een nieuwe template wil maken, moet de oude template verwijderd kunnen worden.

- Template openen
  Het moet mogelijk zijn een template te openen voor executie of bewerking.

- Hulp oproepen
  Indien de gebruiker in moeilijkheden komt tijdens het maken van een template of iets niet snapt, dan moet het mogelijk zijn om hulp in te roepen door middel van een help-systeem die de gebruiker voorziet van antwoorden op veel gestelde vragen en handleidingen.

10.4 Beoogd resultaat

Het beoogde resultaat is een template die, zodra hij is uitgevoerd, de correcte uitkomst geeft die de gebruiker verwacht zonder dat de gebruiker bij het maken van de template moeilijkheden heeft ondervonden.

Pilotplan

In dit Pilotplan staan de componenten uit het systeemconcept, ingedeeld in pilots en op volgorde van prioriteit. De pilots en hun componenten zullen serieel worden ontwikkeld. Dit pilotplan is opgesteld met de “Juicy Bits First”-strategie in gedachten. De pilots zijn geprioriteerd op de *frequentie van gebruik*, de pilots en/of pilotdelen met de hoogste prioriteit, zullen dus het meest gebruikt worden door de gebruiker van BookDB-XML. Dit pilotplan fungeert als een leidraad voor de ontwikkeling van BookDB-XML.

1. Pilot I

Mijlpaal 1: Variabelen bij naam kunnen benoemen

Verwachte inspanning:
- Ontwerp: 1 dag
- Implementatie 1 dag
- Documentatie wijzigen: 1 dag

Risico’s:
- Zie risico’s in plan van aanpak

Mijlpaal 2: Database-selectie SQLSingleResult en SQLAggregatedResult

Verwachte inspanning:
- Ontwerp: 1 dag
- Implementatie: 1 dag
- Documentatie wijzigen: 1 dag

Risico’s:
- Zie risico’s in plan van aanpak

Benodigde middelen en tijd:
- Meest recente versie van BookDB-XML
- Software en hardware genoemd in plan van aanpak
- Twee weken

Deadline: Vrijdag 11 maart

--- Deze pilot is gerealiseerd op maandag 28 februari. ---

2. Pilot II

Mijlpaal 3: Alternatieve oplossing voor SQLLoops

Verwachte inspanning:
- Ontwerp: 5 dagen
- Implementatie 9 dagen
- Documentatie wijzigen: 1 dag

Risico’s:
- Zie risico’s in plan van aanpak

Onderlinge afhankelijkheid:
- Het mechanisme dat in deze pilot wordt ontworpen, dient ook gebruikt te worden in Pilot IV.

Benodigde middelen en tijd:

- Pilot I
- Software en hardware genoemd in plan van aanpak
- Vier weken

Deadline: Vrijdag 25 maart

--- Deze pilot is gerealiseerd op woensdag 2 maart. ---

3. Pilot III

Mijlpaal 4: (Fout)meldingen afvangen en tonen

Verwachte inspanning:
- Ontwerp: 3 dagen
- Implementatie: 5 dagen
- Documentatie wijzigen: 1/2 dag

Risico's:
- Zie risico's genoemd in plan van aanpak

Mijlpaal 5: Seriële executie van taken

Verwachte inspanning:
- Ontwerp: 3 dagen
- Implementatie: 8 dagen
- Documentatie wijzigen: 1/2 dag

Risico's:
- Zie risico's genoemd in plan van aanpak
- Dit pilotdeel is groot. Eerst de meest essentiële delen implementeren.

Benodigde middelen en tijd:
- Pilot II
- Software en hardware genoemd in plan van aanpak
- Drie weken

Deadline: Vrijdag 15 april

--- Deze pilot is gerealiseerd op maandag 21 maart. ---

4. Pilot IV

Mijlpaal 6: Mechanisme voor algemene loop

Verwachte inspanning:
- Ontwerp: 4 dagen
- Implementatie: 5 dagen
- Documentatie wijzigen: 1 dag

Risico's:
- Zie risico's in plan van aanpak

Onderlinge afhankelijkheid:
- Het mechanisme voor deze algemene loop dient in overeenstemming te zijn met het mechanisme dat gerealiseerd is in Pilot II.

Benodigde middelen en tijd:
- Pilot III
- Software en hardware genoemd in plan van aanpak
- Twee weken

Deadline: Vrijdag 29 april

--- Deze pilot is gerealiseerd op vrijdag 25 maart. ---

5. Pilot V

Mijlpaal 7: Mechanisme voor headers en footers voor gebalanceerde tabellen

Verwachte inspanning:
- Ontwerp: 4 dagen
- Implementatie: 5 dagen
- Documentatie wijzigen: 1 dag

Risico's:
- Zie risico's in plan van aanpak

Benodigde middelen en tijd:
- Pilot IV
- Software en hardware genoemd in plan van aanpak
- Twee weken

Deadline: Vrijdag 13 mei

--- Deze pilot is gerealiseerd op woensdag 30 maart. ---

6. Pilot VI

Mijlpaal 8: Oplossing of workaround voor Java-bug

Verwachte inspanning:
- Het vinden van de Java-bug: 1/2 dag
- Onderzoek naar eventuele oplossingen: 2 dagen

  Afhankelijk van de uitkomst van dit onderzoek kan het volgende worden gedaan:

- Eventueel bestaande oplossing of workaround implementeren: 1 dag

  OF:

- Ontwerpen alternatieve workaround voor Java-bug: 1 1/2 dag
- Implementeren alternatieve workaround voor Java-bug: 1 dag (mits de tijd dit toelaat)

  Alle bevindingen die nuttig kunnen zijn voor de toekomst van het project worden gedocumenteerd.

Risico's:
- Zie risico's in plan van aanpak

Benodigde middelen en tijd:
- Pilot V

- Software en hardware genoemd in plan van aanpak
- Maximaal vijf werkdagen

Indien er uit het onderzoek blijkt dat er een oplossing (bug-fix of patch) voor de Java-bug is ontwikkeld, dient deze oplossing te worden gebruikt om de Java-bug op te lossen.
Indien dit niet het geval is, dient er (indien mogelijk) een workaround te worden ontwikkeld en geïmplementeerd.
Indien er meer dan vijf dagen nodig zijn om een eventuele workaround te realiseren, dan zal deze functionaliteit niet worden doorgeschoven naar een volgende pilot, maar worden afgekapt.

--- Deze pilot is gerealiseerd op woensdag 6 april ---

7. Pilot VII

Mijlpaal 9: IF/THEN/ELSE statement

Verwachte inspanning:
- Ontwerp: 5 dagen
- Implementatie: 9 dagen
- Documentatie wijzigen: 1 dag

Risico's:
- Zie risico's in plan van aanpak

Benodigde middelen en tijd:
- Pilot VI
- Software en hardware genoemd in plan van aanpak
- 15 werkdagen (3 weken)

Deadline: Vrijdag 29 april

--- Deze pilot is gerealiseerd op donderdag 21 april ---

8. Pilot VIII

Mijlpaal 10: Grafische user interface voor het samenstellen van templates

Verwachte inspanning:
- Ontwerp: ~~10 dagen~~ 3 1/2 dag (gerealiseerd)
- Gedeeltelijke implementatie: ~~6 dagen~~ 12 1/2 dag
- Documentatie wijzigen: 1 dag

Risico's:
- Zie risico's in plan van aanpak

Benodigde middelen en tijd:
- Pilot VII
- Software en hardware genoemd in plan van aanpak
- 17 werkdagen (4 weken) (3 vrije dagen)

Deadline: Vrijdag 20 mei

--- Deze pilot is gerealiseerd op vrijdag 20 mei ---

Tijdens het ontwerpen en bouwen van de pilot zal de opdrachtnemer zelf goed in de gaten

houden of de pilot aan de eisen voldoet. Tevens zal hij ook de projectleider op de hoogte houden van de vorderingen van de pilot, zodat eventuele dreigende gevaren vermeden kunnen worden.

De pilots dienen uiterlijk op de gestelde deadlines aan de projectleider opgeleverd te worden. Deze zal de pilots dan gelijk in gebruik nemen. Eventuele bugs kunnen hierbij aan het licht komen. In dat geval dienen deze bugs gelijk verholpen te worden.

Pilotontwikkelplan

In dit pilotontwikkelplan staat per pilot beschreven hoe deze tot stand zal komen binnen de begrenzing van de timebox die voor elke pilot is vastgesteld. Aan het begin van elke iteratie van de pilotontwikkelingsfase is die pilotontwikkelplan uitgebreid met de beschrijving van de desbetreffende pilot.

Pilot I

Dit pilotontwikkelplan beschrijft de wijze waarop Pilot I tot stand zal komen binnen de begrenzing van de timebox van twee weken die hiervoor is vastgesteld. Ook de bestede tijd is ingevuld in de tabel.

Table 1: Pilotontwikkelplan Pilot I

Nummer	Omschrijving	Benodigde tijd	Relevantie	*Bestede tijd*
1a	Variabelen bij naam benoemen (ontwerp)	1 dag	Basis	*1/2 uur*
1b	Variabelen bij naam benoemen (implementatie)	1 dag	Basis	*6 uur*
2a	Database selectie SQLAggregatedResult en SQLSingleResult (ontwerpen)	1 dag	Basis	*1/2 uur*
2b	Database selectie SQLAggregatedResult (implementatie)	1 dag	Basis	*1 1/2 uur*
2c	Database selectie SQLSingleResult (implementatie)	1 dag	Basis	*1 1/2 uur*
3	Handleiding wijzigen	1 dag	Basis	*1/2 uur*
	Totaal bestede tijd			*10 1/2 uur*

Pilot II

Dit pilotontwikkelplan beschrijft de wijze waarop Pilot II tot stand zal komen binnen de begrenzing van de timebox van twee weken die hiervoor is vastgesteld.

Table 2: Pilotontwikkelplan Pilot II

Nummer	Omschrijving	Benodigde tijd	Relevantie	*Bestede tijd*
4	Ontwerpen alternatieve oplossing voor SQLLoops	5 dagen	Basis	*8 uur*
5	Implementatie en testen alternatieve oplossing voor SQLLoops	9 dagen	Basis	*7 1/2 uur*
6	Handleiding wijzigen	1 dag	Basis	*0 uur*
	Totaal bestede tijd			*15 1/2 uur*

Pilot III

Dit pilotontwikkelplan beschrijft de wijze waarop Pilot III tot stand zal komen binnen de begrenzingen van de timebox van drie weken die hiervoor is vastgesteld. Punt 13 en 14 zijn optionele functionaliteiten die alleen zullen worden geimplementeerd indien hier tijd voor over is.

Table 3: Pilotontwikkelplan Pilot III

Nummer	Omschrijving	Benodigde tijd	Relevantie	*Bestede tijd*
7	Ontwerpen afvangen foutmeldingen	3 dagen	Basis	*18 uur*
8	Implementatie afvangen foutmeldingen	5 dagen	Basis	*6 1/2 uur*
9	Handleiding wijzigen	1/2 dag	Basis	*0 uur*
10	Ontwerpen seriële executie van taken	3 dagen	Basis	*11 uur*
11	Implementatie seriële executie van taken	8 dagen	Basis	*45 uur*
12	Handleiding wijzigen	1/2 dag	Basis	*1 uur*
13	Implementeren aanvinken of template wel of niet moet worden geexecuteerd	2 dagen	Luxe	*0 uur*
14	Implementeren tijdstip executie	2 dagen	Luxe	*0 uur*
	Totaal bestede tijd			*81.5 uur*

Pilot IV

Dit pilotontwikkelplan beschrijft de wijze waarop Pilot IV tot stand zal komen binnen de begrenzingen van de timebox van twee weken die hiervoor is vastgesteld.

Table 4: Pilotontwikkelplan Pilot IV

Nummer	Omschrijving	Benodigde tijd	Relevantie	*Bestede tijd*
15	Ontwerpen mechanisme voor algemene loop	4 dagen	Basis	*3 1/2 uur*
16	Implementeren mechanisme voor algemene loop	5 dagen	Basis	*25 uur*
17	Handleiding wijzigen	1 uur	Basis	*1 1/2 uur*
	Totaal bestede tijd			*30 uur*

Pilot V

Dit pilotontwikkelplan beschrijft de wijze waarop Pilot V tot stand zal komen binnen de begrenzingen van de timebox van twee weken die hiervoor is vastgesteld.

Table 5: Pilotontwikkelplan Pilot V

Nummer	Omschrijving	Benodigde tijd	Relevantie	*Bestede tijd*
18	Ontwerpen mechanisme voor headers en footers voor gebalanceerde tabellen	4 dagen	Basis	*5 1/2 uur*
19	Implementeren mechanisme voor headers en footers voor gebalanceerde tabellen	5 dagen	Basis	*1 1/2 uur*
20	Handleiding wijzigen	1 uur	Basis	*1/2 uur*
	Totaal bestede tijd			*7 1/2 uur*

Pilot VI

Dit pilotontwikkelplan beschrijft de wijze waarop Pilot VI tot stand zal komen binnen de begrenzingen van de timebox van één die hiervoor is vastgesteld. Hierbij moet worden vermeld dat er maximaal één week aan deze functionaliteit wordt gewerkt. Indien het gewenste resultaat niet binnen deze week wordt behaald, zullen de bevindingen worden gerapporteerd en zal er verder worden gegaan met de volgende pilot.

Table 6: Pilotontwikkelplan Pilot VI

Nummer	Omschrijving	Benodigde tijd	Relevantie	*Bestede tijd*
21	Het vinden van de Java-bug	1/2 dag	Basis	*2 uur*
22	Onderzoek naar eventuele oplossingen	2 dagen	Basis	*15 uur*
23a	Eventueel bestaande oplossing of workaround implementeren	1 dag	Comfort	*7 uur*
23b	Ontwerpen alternatieve workaround voor Java-bug	1 1/2 dag	Comfort	*0 uur*
24	Implementeren alternatieve workaround voor Java-bug	1 dag	Comfort	*0 uur*
	Totaal bestede tijd			*24 uur*

Als aan de hand van het onderzoek (punt 22) blijkt dat er een bestaande oplossing beschikbaar is voor de bug (een patch of een bugfix), dan zal deze oplossing worden geimplementeerd. De resultaten van dit onderzoek zullen worden gedocumenteerd. Als blijkt dat er geen oplossing is, zal er geprobeerd worden een workaround te ontwerpen voor deze bug. Mocht er nog tijd over zijn binnen de timebox, dan zal deze workaround ook worden geïmplementeerd. Indien er geen tijd meer is, dan zal deze pilot worden beeindigd.

Pilot VII

Dit pilotontwikkelplan beschrijft de wijze waarop Pilot VII tot stand zal komen binnen de begrenzingen van de timebox van drie weken die hiervoor is vastgesteld.

Table 7: Pilotontwikkelplan Pilot VII

Nummer	Omschrijving	Benodigde tijd	Relevantie	*Bestede tijd*
25	Ontwerpen IF/THEN/ELSE statement	5 dagen	Basis	*21,5*
26	Implementeren IF/THEN/ELSE statement	9 dagen	Basis	*34*
27	Handleiding wijzigen	1 dag	Basis	*3 uur*
	Totaal bestede tijd			*58 1/2 uur*

Pilot VIII

Dit pilotontwikkelplan beschrijft de wijze waarop Pilot VIII tot stand zal komen binnen de begrenzingen van de timebox van vier weken (18 werkdagen) die hiervoor is vastgesteld. (Hierbij zijn vrije dagen bij ingecalculeerd.) Na het ontwerp, is dit pilotontwikkelplan weer aangepast om de hieruit volgende softwarebouweenheden te specificeren en ordenen op prioriteit. Omdat deze pilot niet volledig geimplementeerd zal worden (dit hoeft ook niet), zullen alleen de belangrijkste onderdelen behandeld worden, mits zij op tijd af kunnen komen.

Table 8: Pilotontwikkelplan Pilot VIII

Nummer	Omschrijving	Benodigde tijd	Relevantie	*Bestede tijd*
28	Ontwerpen GUI voor genereren template	10 dagen	Basis	*28 uur*
29	*Implementeren GUI voor genereren template:*	~~8 dagen~~ 12,5 dagen totaal	Comfort	*74 (totaal)*
29a	Layout template-editor	1 dag	Comfort	*3 uur*
29b	Template openen/opslaan	1/2 dag	Comfort	*4 uur*
29c	Root-element specificeren	1/2 dag	Comfort	*1 uur*
29d	Database selecteren	1 1/2 dag	Comfort	*12 uur*
29e	Database-connectie testen	1 1/2 dag	Comfort	*10 uur*
29f	Beschikbare BookDB-XML elementen toevoegen aan template	4 dagen	Comfort	*30 uur*
29g	Tabel- en kolomnamen opvragen	1 dag	Comfort	*6 uur*
29h	Template valideren	1/2 dag	Comfort	*2 1/2 uur*
29i	Huidige template verwijderen	1/2 dag	Comfort	*1 1/2 uur*
29j	Hulp oproepen	1 1/2 dag	Luxe	*4 uur*
30	Handleiding wijzigen	1 dag	Basis	*6 uur*
	Totaal bestede tijd			*108 uur*

Ontwerp Pilot I

1. Variabelen bij naam kunnen benoemen

De gebruiker van BookDB-XML moet in staat zijn variabelen aan te kunnen maken bij het samenstellen van zijn of haar template en deze met een naam te kunnen benoemen. Hiervoor dienen er drie dingen te veranderen:

- De functie “initvar” dient dusdanig te worden aangepast zodat een string met de naam van de variable en een string met de waarde van de variabele kan worden meegegeven als parameter. Op deze manier blijft het ook mogelijk om de variabele met een nummer aan te roepen indien de gebruiker dat wenst.
- De waarden dienen op te worden geslagen in een Hashtable, hiermee is het mogelijk om objecten op te slaan en deze met een string te identificeren.
- De functie “readvar” dient dusdanig te worden aangepast zodat een string (met de naam van de variable) kan worden meegegeven als parameter waardoor deze functie de inhoud van die variabele uit de Hashtable kan halen.

De syntax van de functieaanroepen zal er als volgt uit kunnen zien:

public boolean initvar(String index, String value)
public String readvar(String index)

2. Database selectie SQLSingleResult en SQLAggregatedResult

De gebruiker van BookDB-XML moet in staat zijn deze functies vanuit zijn of haar template aan te kunnen roepen en daarbij eventueel ook een alternatieve database als parameter mee te geven. Hiervoor dienen er per functie twee dingen te veranderen:

- Er moet een nieuwe functie met dezelfde naam, maar met een extra parameter (usedb) worden toegevoegd aan BookDB.java. Aan deze parameter kan de naam van de te gebruiken database worden meegegeven. Zodra deze functie wordt aangeroepen, zal de variabele CurrentDatabase tijdelijk worden veranderd in de naam van de parameter. Zodra de functie is uitgevoerd zal de originele waarde van CurrentDatabase weer terug worden gezet.
- Bookdbxml.xsl dient aangepast te worden voor de functie, zodat deze ook daadwerkelijk herkent wordt als de gebruiker de functie aanroept. Apache XALAN zal de functie dan herkennen en hem uitvoeren.

De bestaande functies SQLSingleResult en SQLAggregatedResult in BookDB.java blijven gewoon bestaan, zodat deze gewoon uitgevoerd kunnen worden indien er een standaard database wordt gebruikt.

De syntax van de functieaanroepen zal er als volgt uit kunnen zien:

public String SQLSingleResult(String query, String field, String usedb)
public String SQLAggregatedResult(String query, String field, String usedb)

Ontwerp Pilot II

Alternatieve oplossing voor SQLLoops

Momenteel wordt “bookdbxml.xsl” net zo lang uitgevoerd totdat er geen SQLLoops meer voorkomen in het tussenresultaat. Omdat “bookdbxml.xsl” ook bepaalde elementen in het uiteindelijke XML-bestand zet die maar één keer voor dienen te komen, bijvoorbeeld een euro-teken voor de prijs van een artikel, worden deze ook meerdere keren uitgevoerd, waardoor er meerdere eurotekens voor de prijs komen te staan. Het idee is om zaken die éénmalig uitgevoerd dienen te worden, in een apart XSL-bestand te zetten. Dit bestand zal dan éénmalig uitgevoerd moeten worden.

1. Eerst dienen alle SQLLoops te worden verwerkt, zodat het tussenresultaat zo ver mogelijk uitgewerkt is. Dit zal worden gedaan door een nieuw XSL-bestand, genaamd “bookdbxml_preprocessor.xsl”. Dit bestand wordt net zovaak uitgevoerd totdat alle SQLLoops uit het tussenresultaat verdwenen zijn.
2. Vervolgens zal het bestand “bookdbxml.xsl” éénmalig worden uitgevoerd om alle overige statements uit te voeren en de resultaten daarvan in het tussenresultaat zetten. De additionele XSL-bestanden die door de gebruiker kunnen worden gespecificeerd, zullen tevens aan dit bestand worden gekoppeld en éénmalig worden uitgevoerd.
3. Tenslotte zal “bookdbxml_clean.xsl” éénmalig uitgevoerd worden om alle tijdelijke elementen te verwijderen.

In de “bookdbxml_preprocessor.xsl” zullen de statements worden ondergebracht die de volgende elementen zullen verwerken, mochten die voorkomen in de door de gebruiker gemaakte template:

- DBINFO
- SQLLoop
- SQLField
- SQLLoopQuery
- SQLLoopResult
- SQLSingleResult
- SQLAggregatedResult

In de “bookdbxml.xsl” zullen de statements worden ondergebracht die de overige elementen zullen verwerken, mochten die voorkomen in de door de gebruiker gemaakte template:

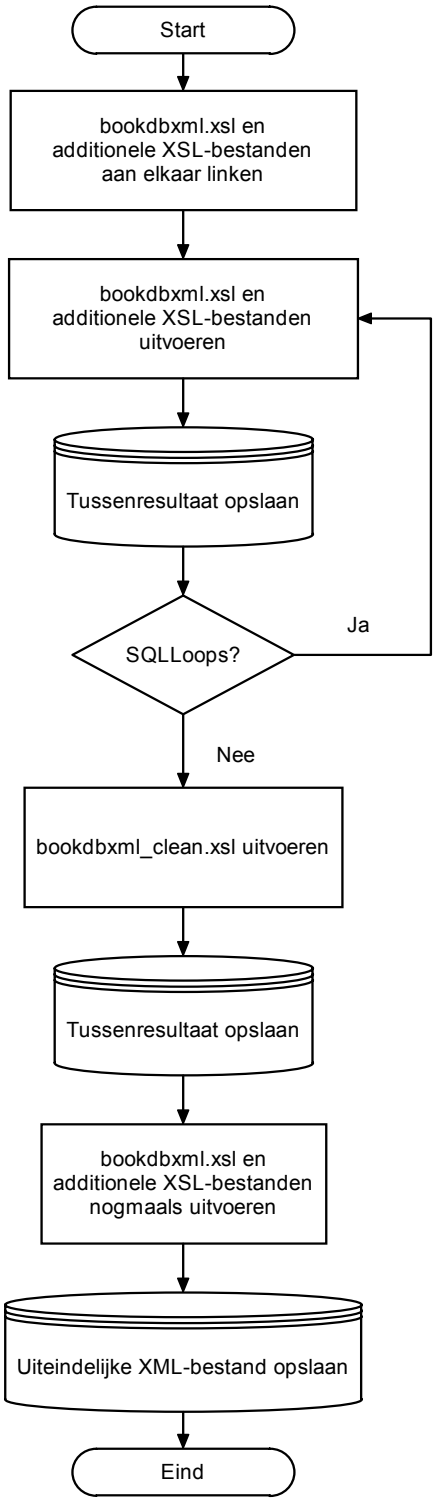
- SetCounter
- IncrCounter
- GetCounter
- SetVar
- GetVar

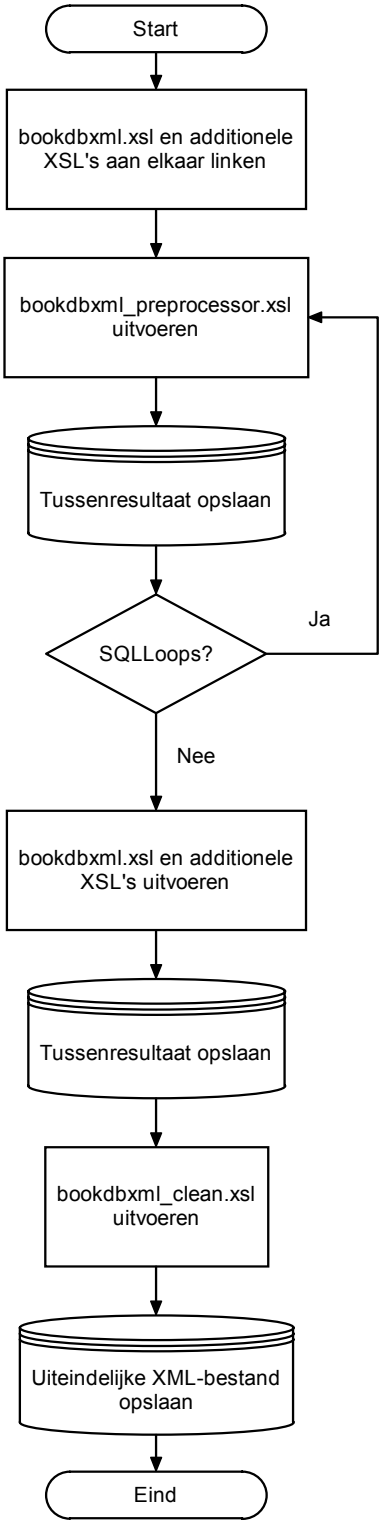
In de “bookdbxml_clean.xsl” zullen de statements worden ondergebracht die alle tijdelijke elementen zullen verwijderen. Dit bestand blijft ongewijzigd.

Met behulp van deze oplossing zullen de statements die meerdere keren uitgevoerd dienen te worden, gescheiden worden van de statements die éénmalig uitgevoerd dienen te worden. Door de statements die meerdere keren uitgevoerd dienen te worden eerst uit te voeren en daarna éénmalig de statements die éénmalig dienen te worden uitgevoerd uit te voeren, worden er geen statements dubbel uitgevoerd. Hierdoor wordt voorkomen dat zaken dubbel worden uitgevoerd. Bovendien werkt het systeem ook iets sneller dan voorheen.

In de volgende figuur zijn de As-Is en To-Be situaties weergegeven in de vorm van flowcharts.

As-Is / To-Be flowcharts





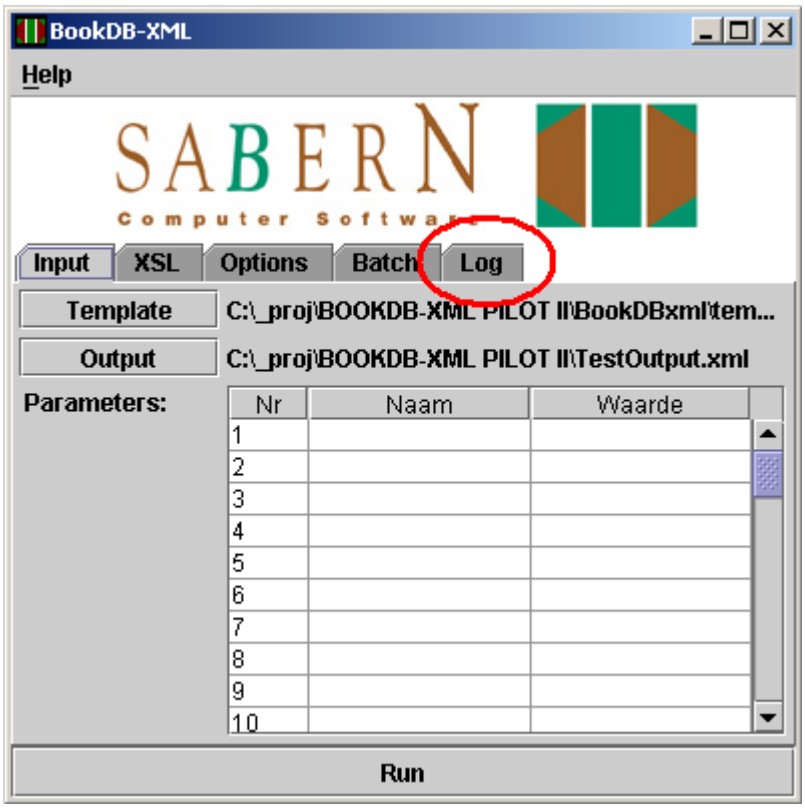
Ontwerp Pilot III

1. (Fout)meldingen beter afvangen en tonen

Momenteel worden de (fout)meldingen die voorkomen tijdens het gebruik van BookDB-XML, weergegeven in de DOS-box of met behulp van dialog-boxes. Deze (fout)meldingen bevatten echter te weinig informatie waar de gebruiker niets mee kan.

Indien er fouten of meldingen voorkomen tijdens het draaien van BookDB-XML, moeten deze aan de gebruiker van BookDB-XML getoond worden op een overzichtelijke manier en op één centrale plaats. Tevens moet de gebruiker op de hoogte gebracht worden van wat de oorzaak van de fout was en wat hij of zij kan doen om de fout te voorkomen. Het idee is om de (fout)meldingen in een apart logtabblad te weergeven.

Het logtabblad zou in BookDB-XML op de in de figuur weergegeven plaats kunnen staan. Zodra de gebruiker op “Run” klikt om het XML-bestand te genereren, zou het logtabblad automatisch open kunnen springen, zodat de gebruiker gelijk kan zien wat er gebeurd. Eventuele meldingen worden dan gelijk aan het logtabblad toegevoegd. Op het logtabblad zou een scrollable JTextField kunnen komen waaraan de nieuwste melding steeds onderaan wordt toegevoegd. Zodra de gebruiker op de “Run”-botton klikt, zal de thread LogKeeper worden gestart. Deze LogKeeper krijgt een methode LogMe waarmee meldingen kunnen worden toegevoegd aan de logtab. Dit kan alleen vanuit de BookDBGUI-klasse, vanuit de klasse BookDB.java is dit niet mogelijk. Indien in deze klasse toch een foutmelding optreedt, zal deze melding vanuit de plaats waar de fout optreedt worden toegevoegd aan een text-bestandje genaamd log.txt in de BookDB-XML-directory. De LogKeeper zal dit bestand lezen nadat de methode TransformerXSLXML is uitgevoerd en kijken of er meldingen in staan. Indien dat zo is, zullen deze meldingen in het logtab worden gezet. Op deze manier is het toch mogelijk om vanuit elke klasse in BookDB-XML een (fout)melding te kunnen weergeven in het logscherm. Hierna zal de LogKeeper worden gestopt en is het proces van het generen van het XML-bestand afgelopen.



2. Seriele executie van taken

Momenteel is het alleen mogelijk om één taak te gelijk uit te voeren. Het moet mogelijk zijn om een hele reeks taken serieel uit te laten voeren.

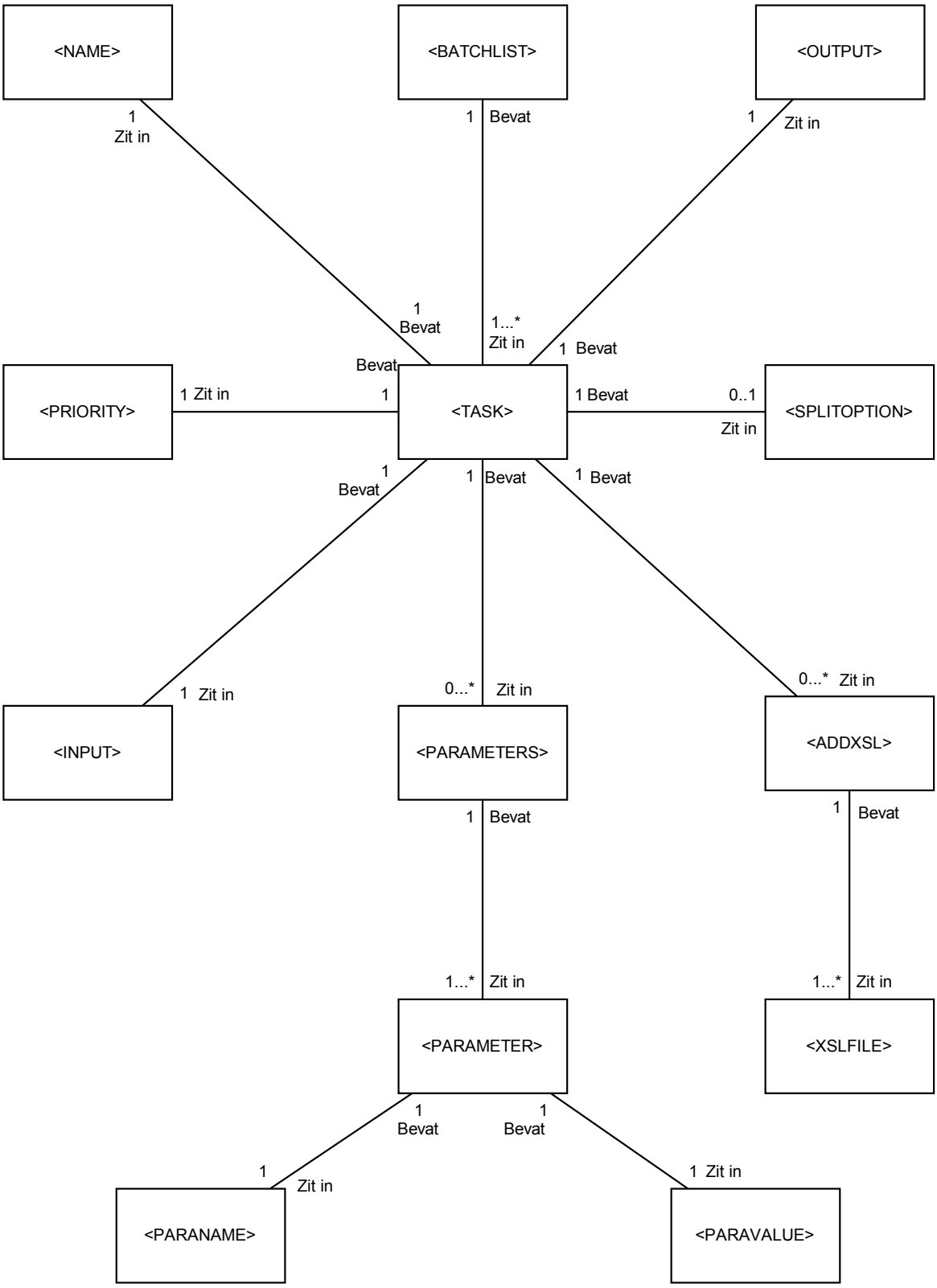
Op het Input-tabblad, zullen drie extra componenten komen:

- Een knop “Add to batchlist” waarmee alle op dat moment ingestelde waarden (template, output-file, parameters, XSL’s en options) toe kunnen worden gevoegd aan het batch-tabblad als een “task” onder een door de gebruiker te bepalen naam. Zodra er op deze knop wordt geklikt, zullen alle ingevulde waarden uit de GUI worden gehaald en worden weggeschreven in een XML-bestand in het geheugen. Op de volgende pagina staat een schematische weergave waarin staat afgebeeld uit welke elementen een batchlist mag bestaan.
- Een textfield om een naam mee te geven waarmee de ingestelde waarden voor de template kunnen worden geïdentificeerd in het batch-tabblad.
- Een knop “Modify” waarmee de gewijzigde waarden voor de geselecteerde template kunnen worden geupdate in de batchlist. Zodra er op deze knop zal worden geklikt, zullen de bestaande waarden in de XML-file overschreven worden door de nieuwe waarden uit de GUI.

Op het Batch-tabblad (zie figuur van Pilot IIIa voor de positie), zullen zeven componenten komen:

- Een lijst met de namen van de uit te voeren templates. Dit is de batchlist. Zodra een item in de lijst wordt geselecteerd, worden de bijbehorende waarden ingevuld in de componenten van de GUI. De gebruiker kan dan de bijbehorende waarden zien en eventueel wijzigen (met de knop “Modify”) mocht dat nodig zijn.
- Een “move up” en “move down” button waarmee de positie van het geselecteerde item in de batchlist kan worden veranderd. Hiermee veranderd ook automatisch de prioriteit. De prioriteit van een task (een integer) bepaald de volgorde van de uit te voeren tasks. (Nummer 1 gaat eerst, dan 2, dan 3 etc.)
- Een “delete” button waarmee het geselecteerde item uit de lijst (en dus ook uit de XML-file) kan worden verwijderd.
- Een “reset” button waarmee de batchlist leeg kan worden gemaakt.
- Een “save” en “open” button om de lijst (de XML-file) mee op te slaan en te openen.
- Een knop “Run batch” waarmee alle items in de lijst op volgorde worden geëxecuteerd.

Het onderstaande klassendiagram geeft de structuur van een (opgeslagen) batchlist in XML weer.


<NAME>
<BATCHLIST>
<OUTPUT>
1
Zit in
1
Bevat
1
Zit in
1
Bevat
1...*
Zit in
1 Bevat
Bevat
<PRIORITY>
1 Zit in
1
<TASK>
1 Bevat
0..1
Zit in
<SPLITOPTION>
1
Bevat
1
Bevat
1 Bevat
1 Zit in
<INPUT>
0...*
Zit in
<PARAMETERS>
0...*
Zit in
<ADDXSL>
1
Bevat
1
Bevat
1...*
Zit in
<PARAMETER>
1...*
Zit in
<XSLFILE>
1
Bevat
1
Bevat
1
Zit in
<PARANAME>
1 Zit in
<PARAVALUE>

Ontwerp Pilot IV

Mechanisme voor algemene loop

Het moet voor de gebruiker van BookDB-XML mogelijk zijn om zelf te bepalen hoe vaak een bepaalde loop moet worden uitgevoerd. Tevens moet het mogelijk zijn variabelen mee te geven welke gebruikt kunnen worden in de loop. Dit is momenteel nog niet mogelijk.

Om dit mogelijk te maken, kunnen er in de klasse BookDB.java twee extra methoden worden aangemaakt. Aan een van deze methoden kan een parameter worden meegegeven (een String), waarin comma-separated de namen en/of nummers van variabelen staan. Deze String zal dan worden opgesplitst en opgeslagen in een Vector, waarna de loop n keer doorlopen zal worden, waarin n hier het aantal namen en/of nummers is dat is meegegeven als parameter.

De gebruiker zal de functie in de template als volgt aan kunnen roepen:

<FORLoop ID=”1, 3, 7, 8”>

In dit geval zal de loop vier keer worden uitgevoerd, waarbij per iteratie de waarde van de variabele kan worden gebruikt, in dit geval dus 1, 3, 7 en 8. Zodra deze functie voorkomt in de template, zal Apache XALAN deze herkennen. Eerst wordt een methode aangeroepen in BookDB.java, waaraan de variabelen worden meegegeven in één String. Deze methode zal de String opsplitsen in losse variabelen, welke worden opgeslagen in een Vector. Vervolgens zal een andere methode worden aangeroepen die de daadwerkelijke FORLoop uitvoert. Indien er in deze FORLoop een element <FORLoopVar/> voorkomt, zal deze worden vervangen door de variabele. Dit zal geschieden in de volgorde waarop de variabelen zijn ingevuld in de FORLoop. Deze waarde kan dan worden gebruikt in een SQL-Statement van een andere functie. Indien er binnen deze FORLoop nog een andere FORLoop voorkomt, zal die pas worden verwerkt op het moment dat de bookdbxml_preprocessor.xsl weer wordt uitgevoerd.

De syntax van de functieaanroepen in BookDB.java zal er als volgt uit kunnen zien:

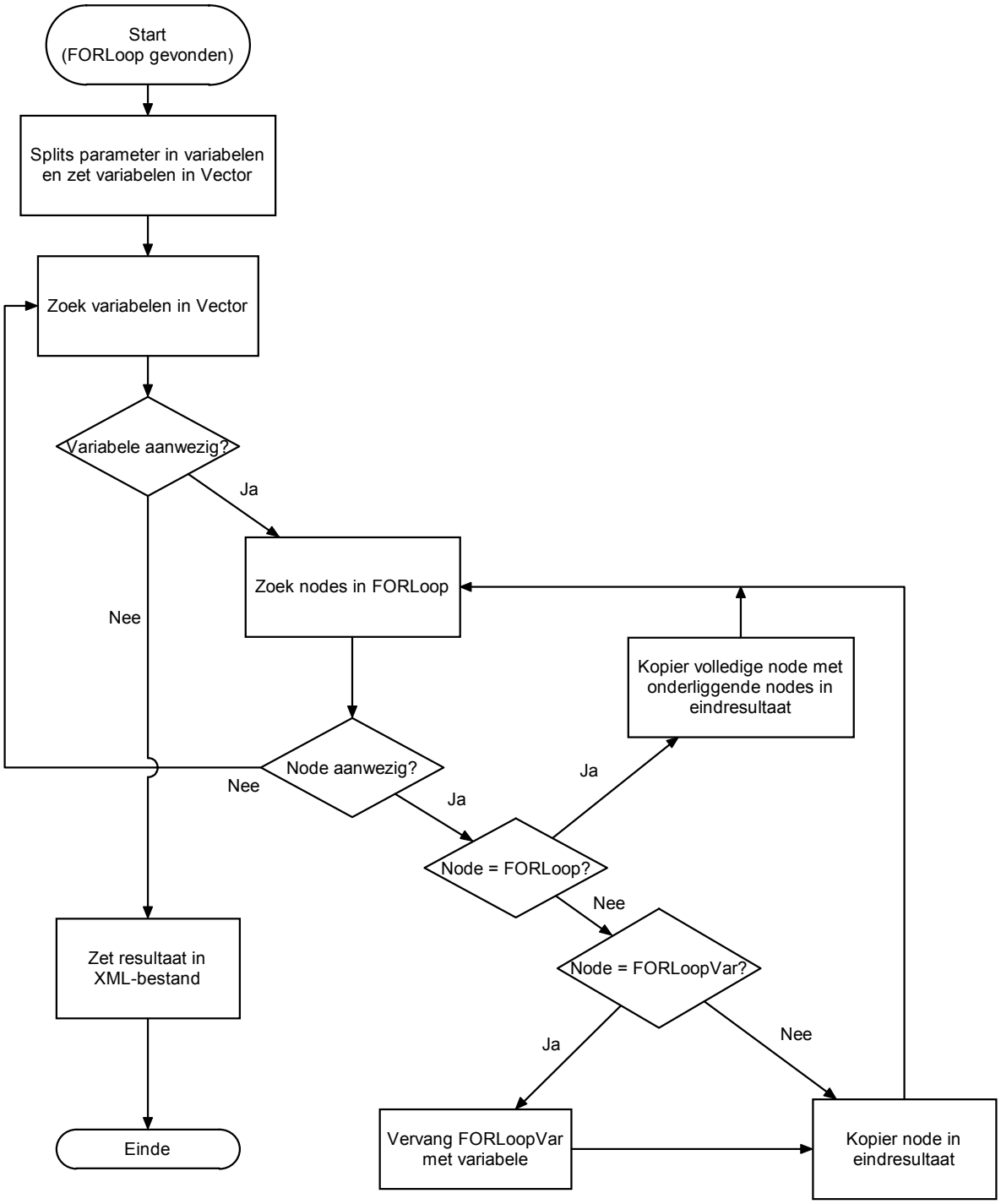
public boolean prepare_forloop(String parameters)
Deze methode zal de String parameters opsplitsen in de losse variabelen, zoals meegegeven in de FORLoop. Deze variabelen worden vervolgens opgeslagen in een Vector.

public DocumentFragment FORLoop(XSLProcessorContext context, org.w3c.dom.Element elem)
Deze methode zal de FORLoop daadwerkelijk n keer uitvoeren en de <FORLoopVAR/> en daarbij de volgende methode aanroepen:

private DocumentFragment FORVarLoop(Document doc, Node node, String forNode, int level)
Deze methode zal de elementen vervangen met de variabele die bij de desbetreffende iteratie hoort.

Om eventueel geneste FORLoops uit te voeren, zal aanroep van de FORLoop in de bookdbxml_preprocessor.xsl komen te staan. (Zie Pilot II.) Omdat deze net zo lang wordt uitgevoerd totdat alle SQLLoops en nu dus ook alle FORLoops verwerkt zijn, is het mogelijk om eventuele geneste FORLoops te verwerken.

In de onderstaande flowchart is het gehele proces overzichtelijk weergegeven.


Start
(FORLoop gevonden)
Splits parameter in variabelen en zet variabelen in Vector
Zoek variabelen in Vector
Variabele aanwezig?
Ja
Nee
Zoek nodes in FORLoop
Kopier volledige node met onderliggende nodes in eindresultaat
Node aanwezig?
Nee
Ja
Ja
Node = FORLoop?
Nee
Node = FORLoopVar?
Ja
Nee
Zet resultaat in XML-bestand
Einde
Vervang FORLoopVar met variabele
Kopier node in eindresultaat

Ontwerp Pilot V

Mechanisme voor headers en footers voor gebalanceerde tabellen

Momenteel bestaat er de mogelijkheid om gebalanceerde tabellen te genereren vanuit de template. De mogelijkheid om hier een header- en footer-rij aan toe te voegen, bestaat nog niet. Deze mogelijkheid moet er komen.

De methode BalancedTable kan via de template worden aangeroepen door de gebruiker. Apache XALAN zal deze functie herkennen en eerst de methode SetBalancedAttributes aanroepen om de variabelen in BookDB.java te vullen met de meegegeven parameters. De methode BalancedTable genereert dan aan de hand van deze gevulde variabelen een tabel. Het aantal kolommen, de naam van de tabel, de naam van de rijen en het element wat in de cellen moet komen te staan kunnen allen als parameter worden meegegeven in de template. Het idee is om hier ook twee optionele parameters aan toe te voegen om een header- en/of footer-rij mee te definiëren. Deze zullen dan als elementen worden toegevoegd aan het begin en/of aan het einde van de tabel. In FrameMaker kunnen deze dan later worden ingesteld als header en/of footer.

Om dit te realiseren zal de bestaande methode SetBalancedAttributes worden aangepast. Deze zal de volgende functieaanroep hebben:

public boolean SetBalancedAttributes(int _cols, String _tblname, String _rowname, String _cellname, String header, String footer)

Aan deze methoden kunnen dan naast de verplichte parameters (aantal kolommen, tabelnaam, rijnaam en het element wat in de cellen moet komen) ook de optionele parameters worden meegegeven om een header en/of footer te genereren. (Deze kunnen ook leeg worden gelaten door als parameter “” mee te geven.) Deze methode zal dan de hiervoor bedoelde variabelen vullen met de waarden van de parameters, waarna de methode BalancedTable zal worden aangeroepen om de tabel te genereren. De methode BalancedTable zal ook worden aangepast zodat deze ook eventueel een header en/of footer zal genereren.

Bevindingen Pilot VI

De bug

Zodra een SQL-query wordt uitgevoerd in BookDB-XML, worden de resultaten hiervan in een scrollable ResulSet geplaatst. Deze tabel is “scrollable”, wat wil zeggen dat er met behulp van een denkbeeldige cursor door de waarden in de tabel kan worden gebladerd. Zodra er een scrollable ResultSet wordt gebruikt om het resultaat van een SQL-query op te slaan waarin een DISTINCT-keyword voorkomt, gaat het mis: De ResultSet krijgt een grootte welke gelijkwaardig aan het aantal waarden wat uit een dezefde query komt, maar dan zonder het DISTINCT-keyword erin. Omdat uit queries zonder een DISTINCT-keyword in de meeste gevallen meer waarden komen dan uit een query met een DISTINCT-keyword, wordt de ResultSet dus te groot. De juiste waarden worden aan het begin van de ResultSet geplaatst, waarna de overige cellen met NULL-waarden worden gevuld. Hetzelfde is het geval met een query waarin een GROUP BY keyword voorkomt. Het weglaten van de onderste cellen met NULL-waarden is hiervoor geen oplossing. Dit werkt wel bij een query op een enkele tabel, maar niet bij een query die betrekking heeft op meerdere tabellen, omdat... Java maakt geen onderscheid tussen een NULL-waarde uit de database en een Java-null waarde.

Scrollable ResultSet

SELECT	SELECT DISTINCT
380	NULL
380	380
380	385
NULL	390
NULL	395
NULL	400
385	NULL
385	NULL
390	NULL
395	NULL
400	NULL
400	NULL

In de bovenstaande tabel is goed te zien wat er mis gaat. In de eerste kolom is de uitkomst van een gewone SELECT-query te zien. In de tweede kolom is de uitkomst van dezelfde SELECT-query te zien, maar dan met een DISTINCT-keyword erin. De eerste zes records kloppen, echter wordt de rest van de records (die er niet horen te zijn), gevuld met NULL-waarden. Dit hoort niet, hierdoor maakt BookDB-XML onterecht teveel loops.

Omdat het wel goed gaat als de query rechtstreeks op de database wordt uitgevoerd, kan worden geconcludeerd dat de bug ook daadwerkelijk in Java zelf zit en niet ergens anders. Dit kan zijn in de JDBC-driver of in de klasse ResultSet. Er is nog geen bestaande oplossing voor deze bug. In de meest recente Java-versie (1.5.0), komt deze bug ook nog steeds voor. Er is geen bestaande bug-fix of patch die het probleem ook daadwerkelijk op kan lossen.

Mogelijke workarounds:

Er kan alleen een workaround als (tijdelijke) oplossing worden gebruikt. Hierdoor blijft de bug echter wel bestaan, alleen zorgt de bug niet meer voor last.

- Het gebruik van een PreparedStatement in plaats van een Statement om de query uit te voeren
  Iemand op een Java-forum beweert dat bij het uitvoeren van een PreparedStatement wel de juiste ResultSet wordt teruggegeven. BookDB-XML werkt echter alleen met Statements en niet met PreparedStatements. Het is niet mogelijk om binnen een week alle Statements door PreparedStatements te vervangen, omdat beide een net iets andere werking hebben. Een PreparedStatement blijkt tevens sneller te werken dan een Statement, omdat de query al deels wordt gecompileerd voordat hij wordt uitgevoerd.

- Een alternatieve JDBC-driver gebruiken
  Omdat de bug misschien huist in de Sun JDBC-driver, zullen er twee alternatieve drivers worden uitgeprobeerd.

Deze workarounds zullen beide worden uitgeprobeerd. Als blijkt dat een van deze (of beide) workarounds werken, dan zullen deze worden toegepast om de bug te omzeilen. Indien dit niet lukt, zal er worden doorgegaan met de volgende pilot.

Voor de volgende workarounds (die eventueel wel kunnen werken) is niet gekozen:

- Een methode maken om alle redundante NULL-waarden weg te halen uit de ResultSet.
  Dit werkt wel bij een simpele query, maar niet bij een query waarin meerdere tabellen worden gebruikt.

- Een niet-scrollable ResultSet gebruiken
  Dit werkt in principe wel, maar daardoor zullen andere delen van BookDB-XML niet meer correct werken, omdat zij wel een scrollable ResultSet gebruiken. Bovendien is de scrollable ResultSet nodig voor toekomstige functionaliteiten.

- Een algoritme maken die zelf een DISTINCT of GROUP BY uitvoert.
  Er zou een algoritme kunnen komen die, zodra er een query met een DISTINCT- of GROUP BY-keyword uitgevoerd moet worden, de “DISTINCT” of “GROUP BY” er tussen uit haalt, de query zonder het keyword uitvoert en daarna zelf de waarden die meerdere keren voorkomen, elimineert in het eindresultaat en het resultaat terugzet in een scrollable ResultSet. Dit is echter vrij ingewikkeld en niet binnen een week te realiseren.

Conclusie Pilot VI

Workaround voor Java-bug

Er kan worden geconcludeerd dat de bug in de JDBC-driver van Sun zit. De bug komt in ieder geval voor bij het gebruik van een MS Access database. De bug doet zich ook voor bij het gebruik van andere databases, waaronder MySQL. Het gebruik van PreparedStatements blijkt niet te werken; ook hier wordt de verkeerde ResultSet doorgegeven.

Momenteel is de enige manier om de bug te omzeilen het gebruik van een alternatieve JDBC-driver. Er zijn veel alternatieve JDBC-drivers beschikbaar. Sommige van deze drivers zijn vrij te gebruiken, voor anderen moet worden betaald.

Een werkende JDBC-driver waarmee de JDBC-bug is te omzeilen is InfoZoom's JadoZoom. Het pakket kan gebruikt worden om met verschillende databases (welke precies wordt niet genoemd op de website) te communiceren vanuit Java. Het is met name gespecialiseerd in MS Access en MS SQL Server. De driver heeft ook meer mogelijkheden dan de JDBC-driver van Sun met betrekking tot MS Access databases. Een daarvan is het creeeren van een nieuwe MS Access database vanuit Java.

InfoZoom's JadoZoom kost 149 euro. Indien ook de source-code gewenst is, kost het pakket 1149 euro. Op de website van InfoZoom is een trial-versie van JadoZoom te downloaden, welke 30 dagen gebruikt kan worden voor test-doeleinden.

Tevens is de JDBC-bug toegevoegd aan de bug-database van Sun. De kans bestaat dus dat deze bug in de toekomst wordt opgelost. Tot dan kan JadoZoom gebruikt worden voor klanten waarbij het gebruik van queries met een DISTINCT- en/of GROUP BY-keyword vereist is.

Meer informatie over JadoZoom is te vinden op:
http://www.infozoom.de/en_jadoZoom.shtml

Ontwerp Pilot VII

IF/THEN/ELSE statement voor in template

1. De opbouw

Het IF/THEN/ELSE statement kan worden aangeroepen met het <BDB_IF>-element. Dit <BDB_IF>-element kan de volgende drie sub-elementen bevatten:

<BDB_CONDITION>
In dit element wordt de conditie geplaatst waarop getoetst gaat worden. Dit element is verplicht. De uiteindelijke uitkomst van de condition is altijd óf TRUE óf FALSE. Binnen dit element kunnen één of meer van de volgende operators worden gebruikt:

De volgende operators kunnen op waarden uit de database en op door de gebruiker ingevulde waarde worden toegepast (deze zullen TRUE zijn als aan de omschrijving wordt voldaan, anders zijn ze FALSE). Deze operators worden aangeduid als V-operators, operators die op Variabelen uitgevoerd kunnen worden:

- <BDB_EQUAL>
  De waarden moeten gelijk zijn

- <BDB_NOT_EQUAL>
  De waarden moeten niet gelijk zijn

- <BDB_GREATER>
  De eerste waarde moet groter zijn dan de tweede waarde

- <BDB_SMALLER>
  De eerste waarde moet kleiner zijn dan de tweede waarde

- <BDB_GREATER_EQUAL>
  De eerste waarde moet groter of gelijk zijn aan de tweede waarde

- <BDB_SMALLER_EQUAL>
  De eerste waarde moet kleiner of gelijk zijn aan de tweede waarde

Binnen deze subtags moeten de variabelen waarop getoetst dient te worden, worden aangegeven binnen <BDB_PART>-elementen. Dit kunnen vaste waarden zijn, maar ook bijvoorbeeld een SQLField, SQLSingleResult of SQLAggregatedResult. Ook binnen een <BDB_PART>-element kan een operator opgenomen worden.

Aan alle V-operators kan een attribuut “compare” worden meegegeven. Deze kan de volgende waarden hebben:

- compare=”case_insensitive”
  De operator zal geen onderscheid maken tussen hoofdletters en kleine letters. Als dit attribuut wordt weggelaten, zal de operator wél onderscheid maken tussen hoofdletters en kleine letters.

- compare=”focus_on_digits”
  De operator zal een waarde die begint met één of meerdere cijfers, gevolgd door letters, eerst vergelijken op het cijfer-gedeelte en dan pas op het letter-gedeelte. Deze functie is niet beschikbaar voor de operators BDB_EQUAL en BDB_NOT_EQUAL.

De volgende operators (boolean-operators) kunnen worden gebruikt om de resultaten van de bovenstaande operators te toetsen. Deze operators worden aangeduid als B-operators, operators die op Booleans uitgevoerd kunnen worden:

- <BDB_AND>
  Alle waarden moeten waar zijn

- <BDB_OR>
  Minstens één waarde moet waar zijn

- <BDB_NOT>
  Deze operator kan gebruikt worden om één BDB_AND of BDB_OR in te zetten. De tegenovergestelde waarde van de BDB_AND of BDB_OR wordt gegeven.

Ook hier dienen de waarden waarop getoetst dient te worden binnen <BDB_PART>-elementen te staan.

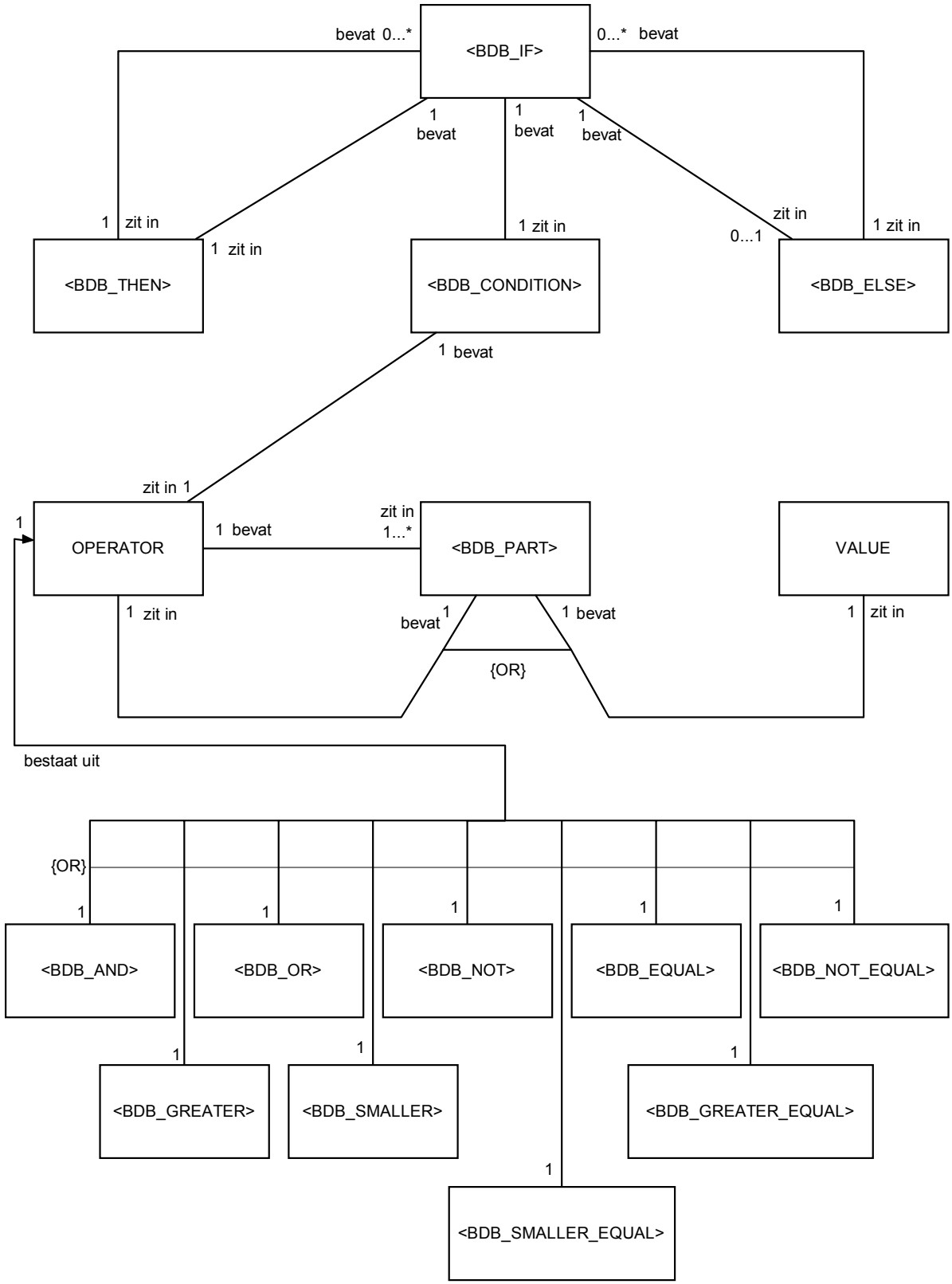
<BDB_THEN>
In dit element kan een gedeelte van de template worden geplaatst dat uitgevoerd dient te worden op het moment dat de uitkomst van het <BDB_CONDITION>-element TRUE is. Dit element is verplicht en dient altijd voor te komen in een BDB_IF-statement.

<BDB_ELSE>
In dit element kan een gedeelte van de template worden geplaatst dat uitgevoerd dient te worden op het moment dat de uitkomst van het <BDB_CONDITION>-element FALSE is. Dit element is optioneel.

Verder gelden de volgende regels:

- Een NULL-waarde uit de database is alleen gelijk met een door de gebruiker ingevulde waarde “NULL” of een leeg <BDB_PART>-element. In alle andere gevallen is hij kleiner.
- Indien men teksten wil vergelijken met de operators GREATER, SMALLER, GREATER_EQUAL en SMALLER_EQUAL, wordt dit op alfabetische volgorde afgehandeld. (Bijvoorbeeld: Amsterdam < Brussel = TRUE, omdat de A eerder in het alfabet staat dan de B.)
- Teksten die beginnen met een cijfer zijn altijd kleiner dan teksten die beginnen met een letter.
- Indien men teksten wil vergelijken die beginnen met cijfers, dan worden de cijfers eerst behandeld. (Bijvoorbeeld: 25cm < 50cm = TRUE.)

In de onderstaande figuur is een klassendiagram weergegeven waarin te zien is hoe een <BDB_IF>-element in de template opgebouwd dient te worden.


bevat 0...*
<BDB_IF>
0...* bevat
1 bevat
1 bevat
1 bevat
1 zit in
zit in 0...1
1 zit in
1 zit in
<BDB_THEN>
<BDB_CONDITION>
<BDB_ELSE>
1 bevat
zit in 1
1
OPERATOR
1 bevat
zit in 1...*
<BDB_PART>
VALUE
1 zit in
bevat 1
1 bevat
1 zit in
{OR}
bestaat uit
{OR}
1
<BDB_AND>
<BDB_OR>
<BDB_NOT>
<BDB_EQUAL>
<BDB_NOT_EQUAL>
<BDB_GREATER>
<BDB_SMALLER>
<BDB_GREATER_EQUAL>
<BDB_SMALLER_EQUAL>

2. De werking

De verwerking van een IF/THEN/ELSE statement zou in de volgende stappen plaats kunnen vinden:

1.
Als eerst worden alle SQL-statements en SQLLoops uitgevoerd, zodat alle benodigde waarden in de IF-boom staan op het moment dat deze uitgevoerd gaat worden. Dit is een voorwaarde voordat er met de verwerking van de IF-boom begonnen kan worden.

2.
Zodra er een <BDB_IF> element voorkomt in de template, zal deze worden herkend door Apache XALAN en zal in BookDB.java de volgende methode worden aangeroepen waaraan het complete <BDB_IF>-element zal worden meegegeven:

public DocumentFragment IfThenElse(XSLProcessorContext context, org.w3c.dom.Element elem)

3.
Deze methode zal als eerst de methode “private boolean prepareIfThenElse(Node node)“ aanroepen die controleert of het <BDB_IF>-element én alle eventueel geneste <BDB_IF>-elementen op de juiste manier zijn opgebouwd. Als dit niet het geval is, wordt een melding gegeven en de transformatie van het <BDB_IF>-element wordt beeindigd. Hiermee wordt voorkomen dat er tijdens de executie van het IF-statement een foutmelding op kan treden. Als het <BDB_IF> element correct is opgebouwd, zal de executie beginnen.

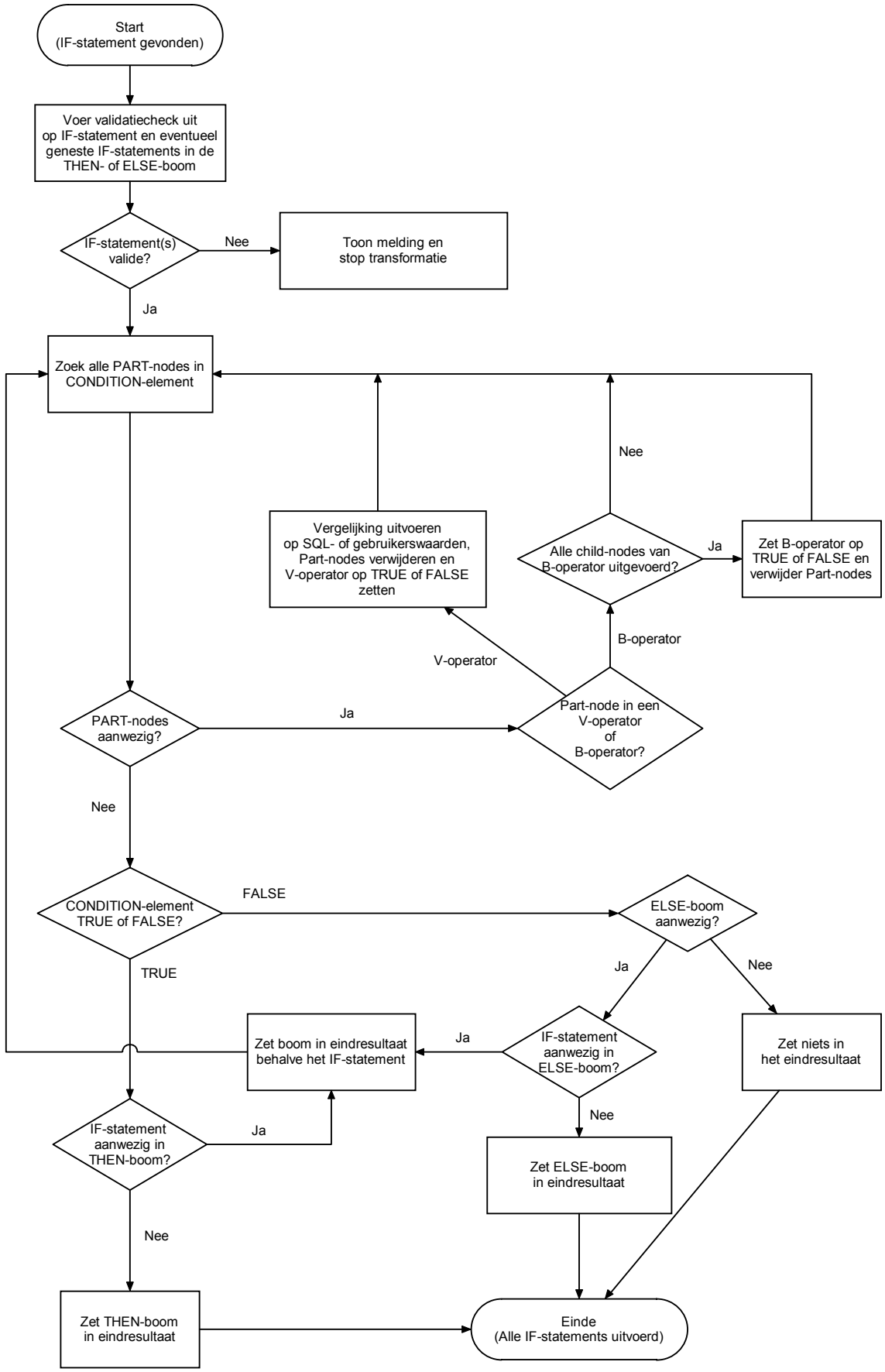
4.
Als eerst wordt het <BDB_IF>-element door de methode “private Node runIfThenElse(Node node)“ uitgevoerd. Deze methode geeft het <BDB_CONDITION>-element mee aan de methode “private void IfIterator(Node node)”. Deze methode zal alle operators zoeken en deze laten berekenen (mist alle dochter-parts ook al zijn uitgevoerd) met behulp van de methode “private boolean processOperator(Node node)”. De methode is recursief en zal zichzelf dus net zo lang aanroepen totdat alle operators zijn behandeld. Aan de hand van de uitkomst (true of false), zal IfIterator de operator vervangen voor een element met de naam TRUE of FALSE. Zodra er geen operators meer zijn, bevat het <BDB_CONDITION> element alleen nog maar één element met de naam TRUE of FALSE.

5.
Indien het TRUE is, zal het <BDB_THEN> element in de XML-boom worden geplaatst. Indien het FALSE is, zal het <BDB_ELSE> element (indien aanwezig) in de XML-boom worden gezet.

6.
Indien er in het resultaat nog meer IF-statements voorkomen, worden ook deze berekend op dezelfde manier zoals hierboven beschreven.

7.
Tenslotte worden de tags verwijderd tijdens de uitvoer van bookdbxml_clean.xsl, zodat alle overbodige tags niet in het eindresultaat komen te staan.

De onderstaande flowchart toont het volledige proces van de verwerking van het IF/THEN/ELSE-statement.


Start
(IF-statement gevonden)
Voer validatiecheck uit
op IF-statement en eventueel
geneste IF-statements in de
THEN- of ELSE-boom
IF-statement(s)
valide?
Nee
Toon melding en
stop transformatie
Ja
Zoek alle PART-nodes in
CONDITION-element
Nee
Vergelijking uitvoeren
op SQL- of gebruikerswaarden,
Part-nodes verwijderen en
V-operator op TRUE of FALSE
zetten
Alle child-nodes van
B-operator uitgevoerd?
Ja
Zet B-operator op
TRUE of FALSE en
verwijder Part-nodes
B-operator
V-operator
PART-nodes
aanwezig?
Ja
Part-node in een
V-operator
of
B-operator?
Nee
CONDITION-element
TRUE of FALSE?
FALSE
ELSE-boom
aanwezig?
TRUE
Ja
Nee
Zet boom in eindresultaat
behalve het IF-statement
Ja
IF-statement
aanwezig in
ELSE-boom?
Zet niets in
het eindresultaat
IF-statement
aanwezig in
THEN-boom?
Ja
Nee
Zet ELSE-boom
in eindresultaat
Nee
Zet THEN-boom
in eindresultaat
Einde
(Alle IF-statements uitvoerd)

Ontwerp Pilot VIII

Graphical User Interface voor het genereren van templates

Om het mogelijk te maken voor de gebruiker om op een gemakkelijke en snelle manier templates te maken, kan gebruik worden gemaakt van een template-editor die delen van de template automatisch genereert en de elementen op de juiste plek zet.

Dit ontwerp is opgebouwd volgens het MVC-concept. Volgens dit concept bestaat de Grafische User Interface uit drie aparte delen die samen één werkend geheel vormen. Dit zijn het Model, de View en de Controller. Deze drie punten worden hieronder apart behandeld.

1. Het Model

Het model is het applicatie-object dat vanuit en met behulp van de GUI wordt gemanipuleerd en gepresenteerd.

Hier is het model de template. De template zal bestaan uit een stuk tekst (String) welke is opgebouwd in valide XML.

2. De View

De view is het object dat bepaalt hoe het onderliggende model precies wordt weergegeven.

Op het hoofdvenster van BookDB-XML zal een knop komen die de view zal openen in een nieuw venster. De view bevat tevens de middelen te behoeve van de interactie van de gebruiker met de applicatie. De view zal bestaan uit de volgende componenten:

- Een groot tekstveld met scrollbars (indien nodig):

  Dit tekstveld zal de volledige template weergeven in gewone tekst.

- Een menu-balk met de volgende items met bijbehorende functies:

  - **File**:

    - “New”
      Er zal gevraagd worden of de huidige template bewaard moet blijven. Zo ja, dan zal een JFileChooser zich openen en vragen onder welke naam de template moet worden opgeslagen. Zo nee, dan zal het tekstveld leeg worden gemaakt. Indien er op cancel wordt geklikt, zal er niets gebeuren.

    - “Open”
      Er zal gevraagd worden of de huidige template bewaard moet blijven. Zo ja, dan zal een JFileChooser zich openen en vragen onder welke naam de template moet worden opgeslagen. Zo nee, dan zal er een JFileChooser zal verschijnen waarmee een bestaande template kan worden geselecteerd. Na het openen van deze template, zal de template in het tekst-veld verschijnen. Indien er op cancel wordt geklikt, zal er niets gebeuren.

    - “Close”
      Er zal worden gevraagd of de huidige template bewaard moet blijven. Zo nee, dan zal de BookDB-XML Template-editor zichzelf sluiten. Indien er op cancel wordt geklikt, zal er niets gebeuren.

    - “Save”
      Als eerst zal er worden gecontroleerd of de template in valide XML is geschreven.

Zo nee, dan wordt een melding gegeven en kan de template niet worden opgeslagen. Indien de template nog niet eerder is opgeslagen zal er een JFileChooser verschijnen waarin een naam voor de template kan worden opgegeven. Indien de template als eens eerder is opgeslagen, zal de oude template zonder melding worden overschreven door de huidige template.

- "Save as"
  Als eerst zal er worden gecontroleerd of de template in valide XML is geschreven. Zo nee, dan wordt een melding gegeven en kan de template niet worden opgeslagen. Een JFileChooser zal verschijnen. Er kan een naam voor de template worden opgegeven, waarna de template zal worden opgeslagen.

- **Edit**:

  - "Find/Replace"
    Een window zal zich openen waarin een stukje tekst kan worden ingevuld waarnaar kan worden gezocht in de template. Tevens kan er een ander stuk tekst worden meegegeven waarmee de eerst genoemde waarden kan worden overschreven.

  - "Validate"
    Er zal een check worden uitgevoerd die controleerd of de template in valide XML is opgebouwd. Een popup-window zal verschijnen met de melding of de template al dan niet valide is. Indien de template niet valide is, wordt er tevens gemeld waarom de template niet valide is.

- **Database**:

  - "Add database"
    Er zal een window openen met een vijftal tekstveldjes, waarin de naam van de database, de driver, de locatie, de gebruikersnaam en wachtwoord kan worden ingevuld. Zodra er op "Test" wordt geklikt, zal er worden gecontroleerd of er een connectie met de database kan worden gemaakt. Door hierna op de knop "Add" te klikken (dit kan alleen als de connectie werkt) zal er een compleet <DBINFO>-element worden toegevoegd aan de template.

  - "Test database"
    Er zal een popup-window verschijnen waarin voor elk <DBINFO>-element wordt aangegeven of de connectie werkt of niet.

  - "Show tables and columns"
    Er zal zich een nieuw venstertje openen waarin van alle gebruikte databases de tabel- en kolomnamen staan. Dit venster kan men open laten staan terwijl men bezig is een template te genereren. Dit is handig voor het opstellen van SQL-queries.

- **Options:**

  - "Editable"
    Door deze optie aan te zetten, is het mogelijk voor de gebruiker handmatig elementen en stukken tekst toe te voegen aan de template. Deze optie staat standaard uit, om te voorkomen dat de gebruiker bepaalde elementen of teksten op de verkeerde plaats zet.

- **Help**:

  - "Help topics"
    Er zal een uitgebreid Help-venster (gebaseerd op JavaHelp) verschijnen waarin onder andere een handleiding van de BookDB-XML Template Editor, definities van

BookDB-XML Elementen en een lijst met veel gestelde vragen (FAQ's) staan.

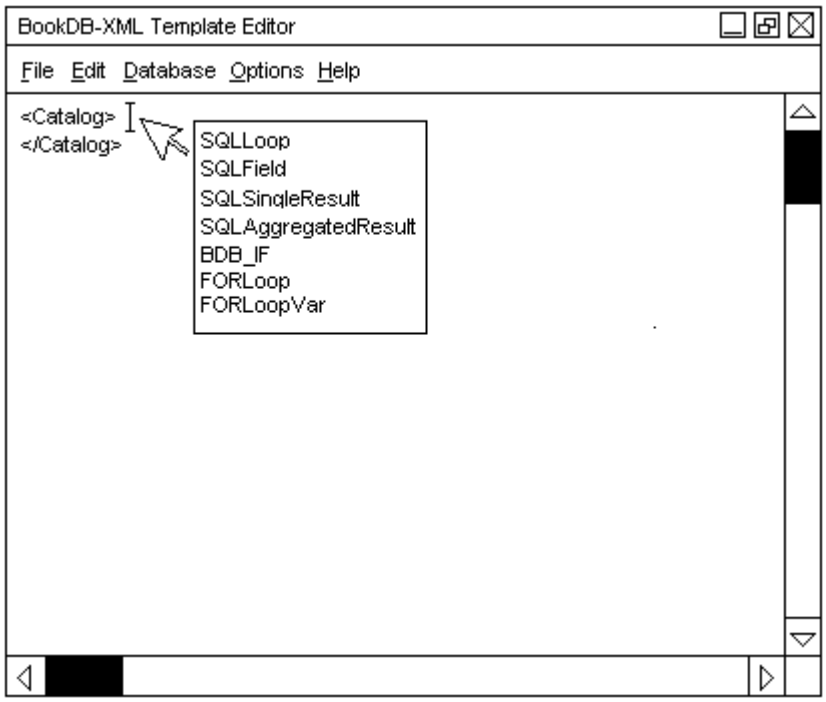
- "About"
  Een klein popup-venstertje met informatie over de BookDB-XML Template-editor zal verschijnen.

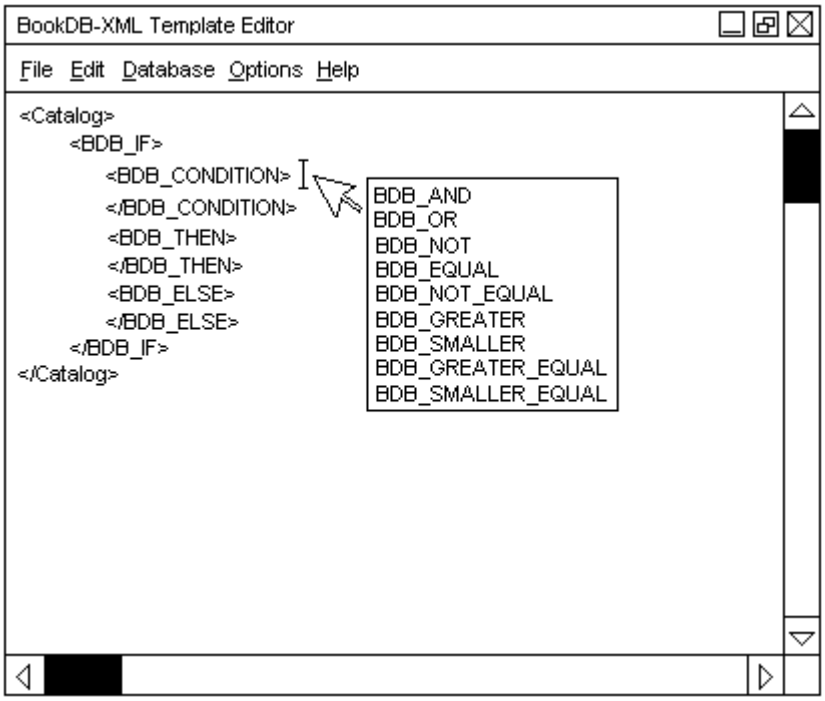
Verder bevat de view de volgende mogelijkheden:

- In het tekstveld is het mogelijk voor de gebruiker om met behulp van de muis een positie in de template te selecteren. Zodra de gebruiker op de rechter muisknop klikt, zal zich een menu openen waarin een element kan worden geselecteerd wat op op de positie van de cursor kan worden ingevuld.

- Indien de gebruiker begint met een lege template, zal hij of zij alleen kunnen kiezen voor een root-element. Dit is een verplicht element waarin de complete template zal komen te staan.

  De onstaande figuren weergegeven een schets van het uiterlijk van de GUI. Hier is te zien dat het Root-element reeds is aangemaakt en de naam Catalog heeft gekregen. De gebruiker klikt hier op de rechter muisknop waarna er een menu opengaat waarin alle elementen staan die op de plaats van de cursor mogen worden gezet. Zodra de gebruiker bijvoorbeeld op BDB_IF klikt, zal er een compleet BDB_IF element worden geplaatst op de plaats van de cursor. Indien de gebruiker hierna het <BDB_CONDITION>-element verder uit wil bouwen, dan kan de gebruiker weer met de rechter muisknop in het element klikken, hierna worden de elementen getoond die binnen het <BDB_CONDITION>-element voor mogen komen. Dit is te zien in de tweede figuur. Op deze manier kan de gebruiker met behulp van de muis het grootste gedeelte van de template genereren.





3. De Controller

De controller is het object dat alle communicatie met de applicatie afhandelt, doorgaans via muis- en toetsenbordinteractie.

De controller zal bestaan uit een groep methoden die zullen zorgen voor de bewerking van de gegevens die de gebruiker invoert.

De werking

Zodra de gebruiker de BookDB-XML Template Editor start en vervolgens met de rechter muisknop in het tekstvenster klikt, krijgt de gebruiker alleen de optie om een root-element toe te voegen. Hiervoor dient hij of zij dan een naam op te geven, waarna het root-element automatisch in het template wordt gezet.

Vervolgens kan de gebruiker in het menu op “Database” klikken, waarna de gebruiker een database kan opgeven voor gebruik. Hiervoor dienen de naam en locatie van de database, driver en gebruikersnaam en wachtwoord te worden ingevuld, waarna de BookDB-XML Template Editor zal controleren of deze gegevens kloppen. Als dat het geval is, zal er een compleet <DBINFO>-element worden aangemaakt en worden toegevoegd aan de template.

Nu de verplichte elementen zijn toegevoegd aan de template, kan de gebruiker met de muis op een positie gaan staan in het tekstveld. Als er dan op de rechter muisknop wordt geklikt, zal er een lijst worden gegenereerd met alle mogelijke elementen die op die plaats voor mogen en kunnen komen. Alle beschikbare elementen staan gedefinieerd in een bestand genaamd “bookdbxml_elements.ini”. In dit bestand staat precies hoe de elementen zijn opgebouwd.

Zodra er op de rechter muisknop in het testveld wordt geklikt, gebeurd er het volgende:

1. Controleer positie in tekstveld
2. Controleer elementen rond positie

3. Bepaal welke elementen er voor mogen/kunnen komen aan de hand van deze elementen
4. Genereer lijst en toon deze aan de gebruiker

Zodra er op een element is geklikt in de hierboven gegenereerde lijst, gebeurd er het volgende:

1. Controleer positie in tekstveld
2. Controleer geselecteerde element
3. Haal het kant en klare element uit bookdbxml_elements.ini
4. Vraag gebruiker eventueel om additionele informatie voor de atttributen van het element.
5. Zet element in tekstveld op positie

Indien de gebruiker bezig is met het opstellen van een SQL-query, is het ook mogelijk om een venster te openen met daarin alle beschikbare tabellen en kolomnamen, welke kunnen worden geselecteerd en geplaatst in de SQL-query. Zodra de connectie met de database is gemaakt, worden deze in het geheugen gezet.

Bijlage E - Buglijst

Bug1

Omschrijving:
In de lijst met parameters in de GUI wordt er, als een bepaalde batch-taak is geselecteerd in de batchlist, de verkeerde parameterwaarde in de GUI gezet. De parameterwaarde is hier de naam van de parameter. Dit klopt niet.

Oorzaak:
De parameternaam wordt in de GUI gezet in plaat van parametervalue. Er wordt dus een verkeerde variabele doorgegeven.

Oplossing:
De variabele ParaValue in de 2e kolom van de parametertabel zetten in plaats van ParaName.

Status:
Opgelost

Bug2

Omschrijving:
De commandline versie van BookDB-XML wil niet opstarten.

Oorzaak:
De LogKeeper wordt aangeroepen, terwijl deze niet is geinstantieerd als de commandline-versie van BookDB-XML wordt opgestart.

Oplossing:
De LogKeeper wordt nu ook geinstantieerd als de commandline-versie van BookDB-XML wordt opgestart.

Status:
Opgelost

Bug3

Omschrijving:
De parameterwaarde wordt niet in de XSLT gezet, terwijl deze toch is meegegeven in de parametertabel in de GUI.

Oorzaak:
De Transformer die zorgt voor de transformatie van bookdbxml.xsl en de additionele xsl's, krijgt geen parameters meegegeven.

Oplossing:
De parameters worden meegegeven aan de Transformer die zorgt voor de transformatie van bookdbxml.xsl en de additionele xsl's.

Status:
Opgelost

Bug4

Omschrijving:
Zodra een batchlist wordt geopend en een taak wordt uit de batchlist verwijderd, waarna er weer een taak wordt toegevoegd, wordt deze toegevoegde taak niet uitgevoerd. Ook als de batchlist wordt geopend, staat de toegevoegde taak niet op de batchlist. Hij staat echter wel in de batchlist zelf als deze wordt opgeslagen.

Oorzaak:
Zodra een taak uit de lijst wordt verwijderd, blijft de teller die bijhoudt welke prioriteit een nieuwe toegevoegde taak moet krijgen, gewoon doortellen. Daarom sluiten de prioriteiten niet meer op elkaar aan en krijg je een “gat” in de prioriteiten. (1, 2, 3, 5, 6) Hierdoor worden de taken achter het “gat” niet in de GUI geplaatst en niet uitgevoerd. Ze staan echter wel in het batchlist-bestand zelf.

Oplossing:
Zodra een taak uit de lijst wordt verwijdert, wordt de prioriteiten-teller met waarde 1 verlaagd:

priorityCounter--;

Status:
Opgelost

Bug5

Omschrijving:
Het opslaan van een batchlist geeft een FileNotFoundException zodra de Transformer het bestand daadwerkelijk naar de harddisk wil schrijven. Deze foutmelding geeft weer dat de bestandsnaam, directorynaam of volumenaam niet klopt, terwijl dit wel klopt.
Dit komt alleen voor in de Java Runtime Environment versie 1.5.0_02 en niet in 1.4.2_07.

Oorzaak:
De bug zit in de JFileChooser, welke zorgt voor een menu waarmee een bestand kan worden geopend of opgeslagen. De Transformer zelf werkt wel.

Oplossing:
-

Status:
Nog niets aan gedaan.

Bijlage F - Visiedocument

Dit document geeft mijn persoonlijke visie weer over de toekomst van BookDB-XML en is gebaseerd op ideeën die ik heb gekregen tijdens mijn afstudeerproject. Dit document is bedoeld voor Sabern BV.

BookDB-XML webserver

Wellicht is het mogelijk om bij Sabern BV een server in te richten waarop een web-based versie van BookDB-XML draait. De klanten kunnen dan met behulp van een website een template invoeren, executeren en vervolgens het resultaat downloaden. De te gebruiken databases kunnen worden geupload.

De voordelen hiervan zijn:

- Bugs en andere problemen in BookDB-XML kunnen door Sabern BV gelijk verholpen worden, zonder dat daarvoor een programmeur hoeft langs te komen bij het bedrijf.
- De klant heeft altijd de meest recente versie van BookDB-XML tot zijn of haar beschikking.
- Sabern BV kan BookDB-XML “verhuren” voor een door de klant bepaalde tijd. Hierdoor kunnen meer potentiële klanten worden geworven.
- De template kan sneller worden verwerkt vanwege de hoge rekencapaciteit van de server.
- Er is geen zware computer nodig voor het verwerken van de templates.
- Misbruik van BookDB-XML (illegale copieen maken) kan makkelijker worden voorkomen.
- Een gratis (gelimiteerde) demoversie kan beschikbaar worden gesteld voor alle geinteresseerden.
- Medewerkers van Sabern BV kunnen makkelijker een eventueel probleem (bijvoorbeeld een template die niet werkt) van een klant achterhalen.

De nadelen hiervan zijn:

- De klant is altijd afhankelijk van de BookDB-XML webserver.
- Het uploaden van een grote database kan lang duren indien met gebruik maakt van een trage internetverbinding.