Afstudeerverslag

Data flow visualisatie van metadata uit een Business Intelligence systeem

Titel	Afstudeerverslag
Project/Onderwerp	Data flow visualisatie van metadata uit een Business Intelligence systeem
Studentnummer	11112131
School en opleiding	De Haagse Hogeschool - Informatica
Door:	Maarten Koene
Datum	05-jun-2015
Periode	09-feb-2015 tot 05-jun-2015
Bedrijf	Info Support B.V.

Referaat

Afstudeerverslag, Maarten Koene, Data flow visualisatie van metadata uit een business intelligence systeem, Info Support te Veenendaal, Haagse Hogeschool, 5 juni 2015.

Dit afstudeerverslag beschrijft het proces rondom de afstudeeropdracht, die gedurende de periode van februari 2015 tot en met juni 2015. De opdracht is uitgevoerd voor Info Support. Deze opdracht is gerelateerd aan de opleiding Informatica aan de Haagse Hogeschool.

De afstudeeropdracht betreft het verrichten van een onderzoek naar de mogelijke visualisatie vormen voor een data flow. Verder wordt er onderzocht of er bestaande software kan worden gebruikt, zo niet dat worden er code library's onderzocht die een data flow visualisatie kunnen ondersteunen. Hierna wordt de data flow visualisatie module ontworpen of gevonden requirements. De module wordt daarna gebouwd en getest. Dit zodat Info Support een grafische inzage heeft in de metadata van een BI systeem.

Descriptoren:

- Data flow
- Datawarehouse
- Datavault
- ETL
- Java
- Mappings
- Metadata
- MoSCoW
- Netbeans Visual

Voorwoord

Het afstudeerverslag is geschreven in het kader van mijn studie Informatica aan de Haagse Hogeschool. De opdracht is uitgevoerd voor Info Support. Tijdens de opdracht wordt er een data flow visualisatie module ontwikkeld. Deze module moet de data flow uit het bestaande metadata platform van een Business Intelligence systeem tonen.

Tijdens mijn afstuderen heb ik kennis opgedaan. Deze kennis wil ik door middel van dit verslag met u delen. Verder wil ik in dit voorwoord graag een aantal mensen bedanken. Allereerst bedank ik Thomas van Bilsen, Orhan Uyan, Pascalle Hijl voor het intern begeleiden van de opdracht. Daarnaast wil ik Henk Brands bedanken voor het beschikbaar stellen van de werknemers van Info Support voor deze opdracht. Verder wil ik graag Arno Nederend en Tim Cocx bedanken voor het begeleiden van de opdracht en het leveren van bruikbare kritiek gedurende de opdracht.

Tot slot wil ik graag Daan van Berkel bedanken voor het reviewen van de code. Ook wil ik Ruben Zorgman, Bart Bijl, Hanjo de Wit, Allard Soeters en Tom Keim bedanken voor de prettige werksfeer binnen en buiten het kantoor.

Maarten Koene
Zoetermeer, 5-6-2015

Inhoudsopgave

Referaat 2

Voorwoord 3

1. Inleiding 6

2. De organisatie 7

2.1 Info Support 7

2.1.1 Missie en kernwaarden 7

2.2 Organisatiestructuur 8

3. Opdrachtomschrijving 9

3.1 Initiële opdracht 9

3.2 Aanleiding 9

3.3 Probleemstelling 10

3.4 Doelstelling 10

3.5 Resultaat 10

3.6 Belangrijkste globale werkzaamheden 11

4. Aanpak en methode 12

4.1 Huidige situatie 12

4.2 Methoden 14

4.2.1 Waarom RUP? 14

5. Inception 16

5.1 Fasering 16

5.2 Project planning 18

6. Onderzoeksfase 19

6.1 Wat is ISMetadata? 19

6.2 Wat is een data flow? 20

6.3 Welke vorm krijgt de data flow visualisatie? 21

6.4 Code library's 22

6.5 Conclusie 26

7. Elaboration 1 27

7.1 Eisen en wensen 27

7.1.1 Brainstorm 27

7.1.2 Prioritering 30

7.2 Ontwerpen 32
7.2.1 Packages 33
7.2.2 Schermontwerp 35
7.2.3 Design 36
7.2.4 Data flow algoritme 41
8. Construction & testing 1 43
8.1 Bouwen 43
8.2 Testen 47
9. Elaboration 2 51
9.1 Ontwerpen 51
10. Construction & testing 2 55
10.1 Bouwen 55
10.2 Testen 56
10.3 Transition 59
11. Evaluatie 60
11.1 Proces evaluatie 60
11.2 Product evaluatie 61
12. Beroepstaken 63
12.1 Beroepstaak: 1.3 Selecteren van standaardsoftware 63
12.2 Beroepstaak: 1.4 Uitvoeren analyse door definitie van requirement 63
12.3 Beroepstaak: 2.3 Uitvoeren gegevensconversie 64
12.4 Beroepstaak: 3.2 Ontwerpen systeemdeel 64
12.5 Beroepstaak: 3.3 Bouwen applicatie 65
12.6 Beroepstaak: 3.5 Uitvoeren van en rapporteren over het testproces 65
13. Bijlagen 66
13.1 Verklarende woordenlijst 66
13.2 Bronnenlijst 66
13.3 Bijlagenlijst 67

1. Inleiding

Voor u ligt het afstudeerverslag van de 17 weken durende afstudeeropdracht "*Data flow visualisatie van metadata uit een Business Intelligence systeem*". Deze opdracht, in het kader van het afstuderen bij aan de opleiding Informatica aan De Haagse Hogeschool te Zoetermeer, is uitgevoerd bij het bedrijf Info Support.

Het doel van dit afstudeerverslag is om inzicht te geven op het afgelegde afstudeerproject. Dit om de examinatoren en gecommitteerden in staat te stellen een positief oordeel over de afstudeeropdracht te kunnen geven over de opdracht.

Om dit te doen begint dit document met het beschrijven van de betrokken organisatie, Info Support. In hoofdstuk 2 wordt onder andere informatie over de achtergrond en het bedrijfsprofiel gegeven. Verder wordt de project organisatie structuur getoond van dit project. In hoofdstuk 3 wordt de uitgevoerde opdracht naar voren gebracht. Hierbij wordt de aanleiding, doelstelling en het resultaat van de opdracht beschreven. In hoofdstuk 4 wordt de gekozen aanpak naar voren gebracht. Dit wordt gedaan door eerst de huidige situatie te schetsen. Daarna wordt er een software ontwikkelmethode gekozen. Hierbij worden de waterval methode, Scrum en RUP tegen elkaar afgewogen. In hoofdstuk 5 wordt de gekozen fasering en planning naar voren gebracht. Daarna wordt het proces van het afstudeerproject beschreven. Hierbij wordt eerst het onderzoek beschreven gevolgd door de requirements analyse. Hierna worden de eerste ontwerpen getoond gevolgd door het bouwproces en testen. Hierna wordt de tweede iteratie besproken op dezelfde wijze als daarvoor, door eerst de ontwerpen te tonen gevolgd door het bouwen en testen. Het proces sluit af door de overdracht van het project te beschrijven. In hoofdstuk 11 wordt het proces en de opgeleverde producten geëvalueerd. In hoofdstuk 12 wordt het behalen van de beroepstaken besproken. Het document eindigt met een bronnenlijst en de bijlagen.

Als er in de tekst de volgende leestekens [..] wordt gevonden dan wordt er naar een bron in de bronnenlijst verwezen in hoofdstuk 13.2. Verder wordt in de tekst verwezen naar afbeeldingen door middel van een nummer. Deze nummers corresponderen met de nummering onder de afbeeldingen. Verder is er in de bijlagen een verklarende woordenlijst te vinden.

2. De organisatie

In dit hoofdstuk wordt het bedrijf Info Support beschreven. Hieruit zal duidelijk worden wat voor bedrijf Info Support is. Verder wordt in dit hoofdstuk de organisatiestructuur naar voren gebracht. Hierbij wordt duidelijk welke personen betrokken zijn tijdens het project en wat hun rol is.

2.1 Info Support

Info Support is een bedrijf dat zich specialiseert in ontwikkelen, beheren en hosten van software op maat. De producten, die worden gemaakt voor klanten, is software. Deze worden speciaal ontwikkeld voor en met de klant Hierbij wordt gebruik gemaakt van standaarden.

Info Support heeft momenteel meer dan 400 medewerkers. Deze medewerkers zijn verspreid over de vestigingen in Nederland en België. Bij Info Support worden de ervaringen en leringen van de medewerkers in het aanbod van trainingen. Er worden op dit moment meer dan 300 verschillende trainingen gegeven. Deze trainingen zijn gericht op ontwikkelaars binnen en buiten het bedrijf. Hierbij is een diversiteit aan aanbod van trainingen. Er is een ruim aanbod in Microsoft (.NET, C#, SQL Server), Oracle (Fusion Middleware, Weblogic Suite) en Java. [1]

Verder wordt er binnen het bedrijf een ruime kennisbank bijgehouden. Deze wordt KnowNow genoemd. Hierin kunnen medewerkers hun ervaringen, onderzoeken, inzichten en oplossingen kwijt. Dit gaat aan de hand van artikelen. Ook wordt er wekelijks een bijeenkomst gehouden. Hierbij zijn alle medewerkers uitgenodigd. Er kan dan vrijwillig worden geluisterd naar presentaties van collega's. Deze presentaties zijn vaak gericht op trends en ontwikkelingen in de ICT. Echter worden ook de jaarcijfers gepresenteerd in dit soort bijeenkomsten.

2.1.1 Missie en kernwaarden

Voor Info Support staat er één ding centraal en dat is: *'Het leveren van toegevoegde waarde voor onze opdrachtgevers'.* Hiervoor biedt Info Support een solide en innovatieve software oplossing die organisaties en bedrijven ondersteunen bij het behalen van hun doelen. Om dit te kunnen realiseren maakt Info Support gebruik van een aantal kernwaarden. Deze kernwaarden worden door iedere medewerker uitgestraald. De kernwaarden zijn afgebeeld in de hal van het hoofdkantoor. Deze afbeeldingen zijn gemaakt door de medewerkers van Info Support tijdens de Nieuwjaars borrel van 2013. Om de kernwaarden naar voren te brengen zullen de afbeeldingen worden getoond en de kernwaarden worden beschreven. [2]

Afbeelding 1, soliditeit

Soliditeit
Solide staat voor duurzaam, stevig en betrouwbaar. Er wordt dus gekozen voor de lange termijn en niet de snelle weg.


Afbeelding 2, integriteit

Integriteit
Aan afspraken houden en open, eerlijke omgang levert het beste resultaat.

Afbeelding 3, vakmanschap

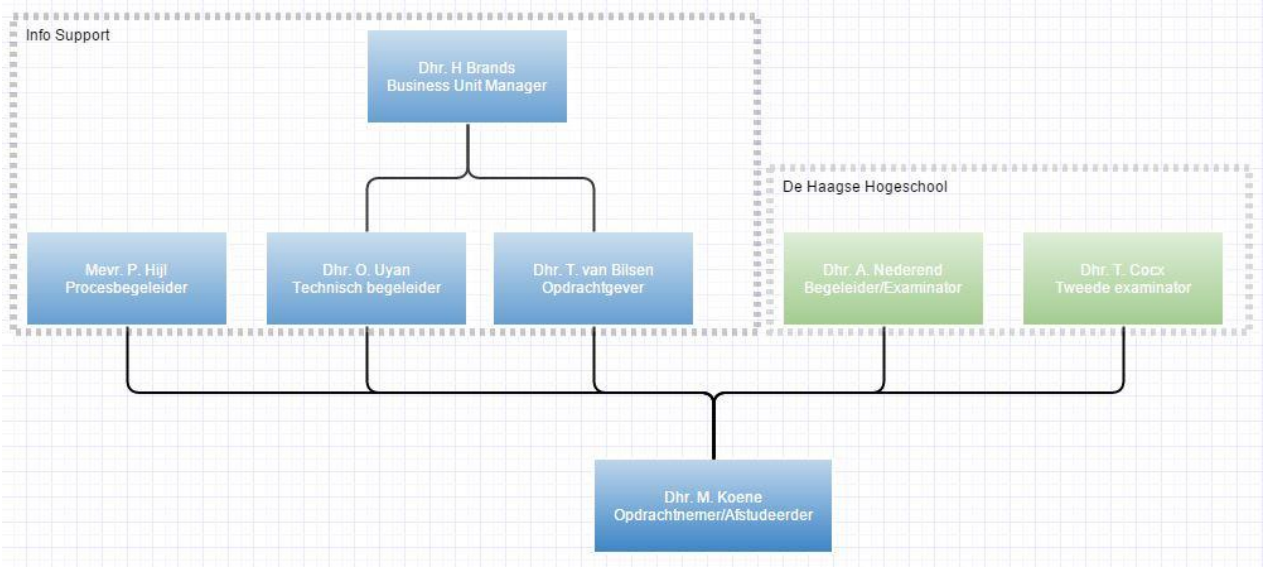
Vakmanschap
De techische kennis van de medewerkers wordt op peil gehouden met trainingen. Verder zijn oplossingen marktgericht en innovatief.

Afbeelding 4, passie

Passie
De medewerkers van Info Support werken met passie aan hun vak om de beste oplossing te realiseren.

2.2 Organisatiestructuur

Tijdens de duur van het project is er een tijdelijke organisatie opgezet. Om inzicht te geven in de structuur van deze organisatie is onderstaand een organigram weergegeven. Per betrokken persoon staat een functie aangegeven. De rolbeschrijving en verantwoordelijkheden staan beschreven in het plan van aanpak (hoofdstuk 4). Voor verdere informatie zie het plan van aanpak in bijlage B.



Afbeelding 5, Organigram project

3. Opdrachtomschrijving

In dit hoofdstuk wordt de uitgevoerde opdracht besproken. Hierbij wordt de aanleiding van de opdracht naar voren gebracht. Ook wordt het huidige situatie met de ontstane probleem geschetst. Verder wordt de doelstelling van de opdracht naar voren gebracht. Het uiteindelijke resultaat, de gewenste situatie, wordt ook naar voren gebracht. Verder zal het hoofdstuk afsluiten met een opsomming van de belangrijkste werkzaamheden

3.1 Initiële opdracht

Voor de start van het afstudeertraject in februari 2015 is de volgende opdrachtomschrijving voorgelegd [3]:

Ontwerp en implementeer een module voor het metadata platform van Info Support voor het weergeven en bewerken van dataflows. Momenteel worden dataflows in het metadata platform ingevoerd en bewerkt via een MS-Access en T-SQL query's.

Via de module die jij gaat ontwikkelen moet het in eerste instantie mogelijk zijn reeds gedefinieerde dataflows op een gebruiksvriendelijke manier te visualiseren. Hiernaast is het de bedoeling dat via de module ook nieuwe dataflows aangemaakt kunnen worden en reeds ingevoerde dataflows aangepast kunnen worden. De wijzigingen die via de module gedaan worden dienen meteen te worden doorgevoerd in de metadata database. Er is reeds een volwassen datastructuur beschikbaar waarop de module geïmplementeerd kan worden.

Voordat je begint met het implementeren van de module is het de bedoeling dat je een onderzoek doet. In dit onderzoek dien je uit te zoeken welke verschillende manieren er zijn voor het visualiseren van dataflows en welke voor ons het meest geschikt is. Hiernaast dien je te onderzoeken of er bestaande software en/of code library's beschikbaar zijn die je eventueel zou kunnen gebruiken voor de door jou te implementeren module.

3.2 Aanleiding

Een aantal klanten van Info Support maakt gebruik van Business Intelligence(BI) systemen. Deze systemen zijn op maat gemaakt voor deze klanten. Bij BI systemen wordt de data uit meerdere databronnen verzamelt. Deze gegevens worden dan opgeslagen in een andere database, deze wordt ook wel datawarehouse genoemd. Deze datawarehouses worden ingezet om analyse uit te voeren op de verzamelde data. Door deze analyse kunnen er nieuwe inzichten worden gecreëerd. Zo kan er bijvoorbeeld inzicht worden gegeven in de kosten van een zieke werknemer over een langere periode.

Om de data uit de verschillende bronnen te gebruiken moet deze worden geïmporteerd. Hiervoor moet de data vaak worden aangepast. Soms is het nodig om de data te aggregeren. Dit wordt gedaan met het oog op performance van het datawarehouse.

Door middel van ETL schema's kan de data uit de bronnen worden gehaald, aangepast en ingeladen in het datawarehouse. Het opstellen van de ETL schema's is repetitief werk. Hiervoor is binnen Info Support (een deel van) het proces geautomatiseerd. Bij deze geautomatiseerde werkwijze kan de structuur van databronnen worden

ingeladen in een metadata database. Op basis van de bronstructuren kan, door een tool, en datavault structuur worden gegenereerd. Hierna wordt met de hand een datawarehouse gemaakt. De structuur kan wederom worden opgeslagen in de metadata database.

Ook kan in de metadata tussen bron en bestemming worden opgeslagen, dit worden mappings genoemd. Deze mappings geven de data flow van een attribuut uit een bronsysteem aan.

De (deels) geautomatiseerde werkwijze van Info Support wordt beschreven in het hoofdstuk onderzoek.

3.3 Probleemstelling

In de metadata wordt de data flow van het attribuut uit de brondatabase naar het andere attribuut van een bestemmingsdatabase beschreven. Het probleem van het opslaan van de data flows in een metadata database is dat deze op beperkte wijze kan worden benaderd. Er is op dit moment geen mogelijkheid om de data flows in de metadata visueel te maken. In de huidige situatie wordt er door ontwerpers en ontwikkelaars door middel van query's en een excel sheet data flows aangemaakt en aangepast.

Het bijkomende probleem van deze werkwijze is dat het aanpassen van de data flows in de metadata een tijdrovende en ingewikkelde taak wordt. Het aanmaken of aanpassen van een datawarehouse in ontwikkeling kan hierdoor vertraging opleveren. Door de beperkte visuele mogelijkheden wordt de kans dat er fouten in de data flows worden gemaakt vergroot. De ontwikkelaars en ontwerpers kunnen het overzicht van de verschillende systemen kwijt raken. Door dit probleem kunnen er onvoorziene fouten ontstaan bij het aanpassen van een bestaand datawarehouse.

3.4 Doelstelling

De afstudeeropdracht heeft twee doelstellingen. De eerste doelstelling is onderzoeken, ontwerpen en implementeren van een data flow visualisatie module. Deze module moet de data flows in de metadata, zoals hierboven beschreven, visueel in beeld brengen. De tweede doelstelling is met de data flow visualisatiemodule nieuwe data flows aan te maken, bestaande te bewerken of verwijderen. Echter heeft de opdrachtgever aangegeven, dat het in beeld brengen van de data flows de belangrijkste functionaliteit is.

3.5 Resultaat

Het resultaat is een data flow visualisatie tool. Deze tool is ontworpen en geïmplementeerd naar de eisen en wensen van de opdrachtgever. Er kan met deze tool een data flow worden gevisualiseerd op basis van opgehaalde metadata uit de metadata database. Deze visualisatie vorm is ook naar wens van de opdrachtgever. Om een voorstel van visualisatie vorm te kunnen doen wordt er onderzoek gedaan naar verschillende visualisatie vormen. Ook wordt er onderzoek gedaan naar code library's die visualisatie van de data flow kunnen ondersteunen. De requirements en functionaliteiten zullen worden getest aan de hand van verschillende testvormen. Verder wordt er voor de Haagse Hogeschool een afstudeerverslag opgeleverd. Dit levert de volgende op te leveren producten op:

- Plan van aanpak;
- Onderzoeksrapport;
- Requirements rapport;
- Functioneel ontwerp;
- Technisch ontwerp;
- Dataflow module;
- Testrapportage;

3.6 Belangrijkste globale werkzaamheden

De volgende belangrijkste globale activiteiten zijn tijdens de afstudeerstage ondernomen:

- Een plan van aanpak schrijven;
- Onderzoek naar verschillende mogelijkheden van dataflow visualisatie;
- Onderzoek naar beschikbare software en code library's voor het visualiseren van dataflows;
- Vooronderzoek naar de huidige werkwijze, waarbij de totstandkoming van de metadata duidelijk wordt;
- Requirements opstellen aan de hand van eisen en wensen van de opdrachtgever;
- Ontwerpen van de dataflow visualisatie module, beschreven in een functioneel en technisch ontwerp;
- Ontwikkelen van de dataflow visualisatie module;
- Testen;
- Overdracht van het project en de data flow visualisatie software;

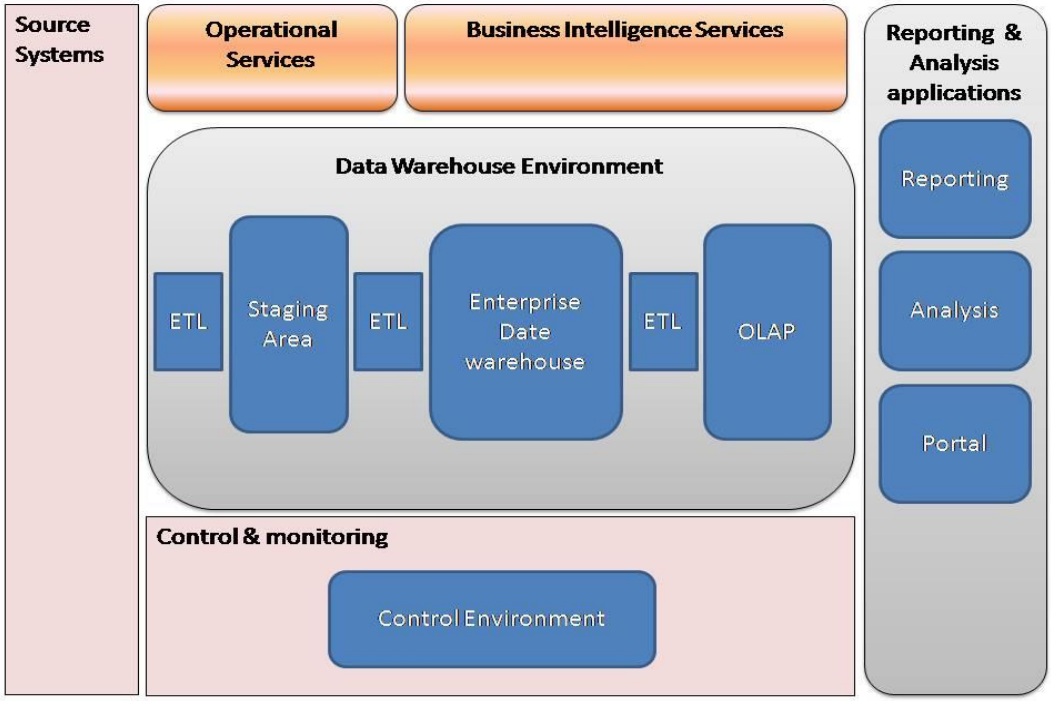
4. Aanpak en methode

In dit hoofdstuk wordt een schets gegeven van de huidige situatie. Daarnaast wordt het voorbereidende werk en opzet dat is gevolgd om tot een oplossing te komen voor de huidige situatie in beeld gebracht. Hierbij wordt in de tweede paragraaf de gekozen software ontwikkelmethode naar voren gebracht. Ook wordt de argumentatie gegeven over de gekozen methode.

4.1 Huidige situatie

Info Support heeft de afdeling Data Solutions, voorheen bekend als de unit Business Intelligence. Info Support maakt voor haar klanten Business Intelligence systemen. Dit wordt gedaan aan de hand van een vast traject met een vaste architectuur. Er is door Info Support hiervoor gekozen omdat de onderhoudbaarheid van het systeem kan worden gewaarborgd. In dit hoofdstuk wordt de huidige gebruikte architectuur voor het opzetten van een Business Intelligence systemen toegelicht. Hieruit zal duidelijk worden welke knelpunten worden ondervonden tijdens het proces.

Voor de klanten van Info Support worden Business Intelligence systemen op maat gemaakt. Om dit proces inzichtelijk en onderhoudbaar te maken is er een standaard architectuur opgesteld. Onderstaande afbeelding toont de huidige architectuurvorm die wordt gebruikt bij klanten van Info Support.



Afbeelding 6, Business Intelligence architectuur van Info Support

Deze afbeelding is afkomstig uit de documentatie van Info Support. Het TRA BI reference document geeft de bovenstaande afbeelding als standaard architectuur.

Om de architectuur toe te lichten wordt het proces van het opzetten van een Business Intelligence systeem weergegeven. De klanten van Info Support willen een Business Intelligence systeem. Dit zorgt voor een systeem waarbij het op de juiste manier combineren van data tot nieuwe inzichten kan leiden. Hierdoor kunnen bedrijfsprocessen efficiënter worden ingericht.

Om de data uit de databronnen, in de afbeelding weergegeven als Source Systems, te combineren worden er een aantal stappen ondernomen. De eerste stap is het extraheren van de data uit de databronnen naar een tijdelijke opslagruimte. Door middel van een ETL schema worden de gegevens gekopieerd naar een Staging area. Een staging area is dus een database kopie van een bron. Deze stap wordt ondernomen zodat de data hierna kan worden bewerkt voor de volgende stap.

Door gebruik van een staging area wordt verzekerd dat de gewenste bewerkingen mogelijk zijn op de data en wordt het bronsysteem zo min mogelijk belast. Dit omdat bronsystemen van allerlei verschillende soorten en maten kunnen zijn. Zo kan een bronsysteem een excel sheet zijn, maar ook een volledige relationele database. Beide bronsystemen ondersteunen niet dezelfde bewerkingsacties.

Nadat de data in de staging area is gebracht wordt deze via een daarop volgend ETL schema aangepast en ingeladen in het op maat gemaakte Enterprise Date warehouse [4]. Dit datawarehouse is ontwikkeld door een medewerker van Info Support in samenwerking met de klant. De klant kan aangeven welke inzichten er in de data mogelijk moeten worden. Op basis van deze wensen en eisen wordt een datawarehouse ontworpen en gemaakt. Om de data te laten passen in het datawarehouse moet deze worden aangepast door middel van met de handgemaakte ETL schema's.

Nadat de gegevens zijn aangepast voor het datawarehouse worden deze opgeslagen in een OLAP omgeving. Een OLAP omgeving is een data opslag met meerdere dimensies. Het voordeel van het opslaan van data in een OLAP omgeving is dat er via verschillende invalshoeken naar data kan worden gekeken. Zo kan er bijvoorbeeld via productprijzen of verkooppunten naar de omzet van een bedrijf worden gekeken.

De OLAP omgeving [5] biedt een mogelijkheid tot analyse van data door de software in de rapportage en analyse omgeving van de architectuur. Deze software is erop gericht om de data grafisch te presenteren aan de eindgebruiker. Denk hierbij aan programma's als Qlik en power BI voor Excel.

Het proces van het opzetten wordt ondersteund door de controle omgeving. In deze omgeving worden tools gebruikt om de voortgang van het proces te ondersteunen en waarborgen. In deze omgeving bestaat de ISMetadata tool. Deze tool kan de metadata van verschillende systemen opslaan. Verder kunnen er mappings worden opgeslagen in de ISMetadata tool. Deze mappings beschrijven dan op welke wijze de attributen uit het bronsysteem naar het bestemmingssysteem worden verplaatst. Deze mappings kunnen met de hand worden toegevoegd.

Echter zoals eerder beschreven is het aanmaken en aanpassen van mappings tijdrovend werk. De ontwikkelaars en ontwerpers van BI systemen kunnen het overzicht niet bewaren. Hierdoor wordt de ontwikkeltijd aanzienlijk verlengt. Ook is de kans op fouten verhoogd.

Om de problemen met het aanmaken en aanpassen van mappings beter in kaart te brengen wordt in het project een onderzoek gedaan naar de exacte werking en de bijbehorende metadata van ISMetadata. Dit onderzoek zal worden beschreven in het hoofdstuk onderzoek.

4.2 Methoden

Voor het realiseren van het project wordt voor het ontwerpen ontwikkelen en testen van de data flow visualisatie tool een software ontwikkelmethode gehanteerd. Dit is om georganiseerd en gestructureerd te werk te kunnen gaan. Ik heb uitiendelijk gekozen om Rational Unified Process te gebruiken. In de volgende paragrafen zal deze keuze nader worden toegelicht en beargumenteerd. Hierbij zullen een aantal kenmerken van het project als criteria worden gebruikt.

4.2.1 Waarom RUP?

Voor de aanpak van dit project heb ik drie software ontwikkelmethoden overwogen. In dit hoofdstuk beschrijft mijn gekozen methodiek. Hierbij overweeg ik de waterval methode, Scrum en RUP. Ik gebruik hierbij een aantal kenmerkende elementen van het project om een overwogen keuze te maken.

Uit de opdrachtomschrijving wordt duidelijk dat er een aantal vaste functionaliteiten zijn verbonden aan de data flow visualisatie module. Er wordt hier gesproken van een eerste versie waarbij een data flow kan worden gevisualiseerd. Daarna moeten pas de functionaliteiten om de data flow aan te passen en aan te maken worden bijgevoegd. Ook blijkt uit de opdracht omschrijving dat de visualisatie vorm onduidelijk is, omdat hier een onderzoek naar moet worden gedaan. Deze visualisatie vorm wordt vastgesteld in de onderzoeksfase en bij de requirements.

De waterval methode [6] is een software ontwikkelmethode die lineair uitgevoerd dient te worden. De opdrachtgever en de opdracht omschrijving laten een duidelijke herhaling zien van het bouwen van de software. In de waterval methode wordt iedere fase eenmalig uitgevoerd. Daardoor is het niet mogelijk om een eerste versie waarbij een data flow wordt gevisualiseerd en daarna functionaliteiten bij te bouwen, zoals het aanmaken of aanpassen van een data flow. Bovendien is waterval uitermate geschikt voor projecten met een groot kostenrisico, voor dit interne project van Info Support is dit niet het geval. Vanwege het verwachtingspatroon van de opdrachtgever om meerdere opleveringen te hebben en omdat het project een intern project is met een laag kostenrisico is het gebruik van de watervalmethode afgevallen. [7]

Een andere software ontwikkelmethode, die ik heb overwogen, is scrum. [8] Scrum geeft ruimte om het project aan te passen aan veranderende eisen. [9] Uit de opdracht omschrijving blijkt dat er een duidelijk beeld is van de applicatie bij de stakeholders. De vorm van de visualisatie lijkt een risico factor, deze wordt echter vroeg in het project afgedekt door het onderzoek en de requirements. Het is hierdoor niet nodig om paraat te staan voor veel verschuivende requirements. Ik heb daarom gekozen om scrum niet te gebruiken als software ontwikkelmethode.

RUP is een iteratieve software methode [10]. In deze methode wordt van te voren de iteraties gepland.[11] Uit de opdracht omschrijving zijn er minimaal twee iteraties naar voren komen. De opdrachtgever heeft aangegeven tijdens het project betrokken te willen zijn. Ook zei hij een eerste, waarin een data flow kan worden gevisualiseerd, en een tweede versie, met aanpasbaarheid van de data flow, te verwachten. De mogelijkheid tot meerdere iteraties, waarbij er steeds een stuk software wordt gemaakt lijkt hier goed op aan te sluiten. Verder zullen de requirements van het project niet (veel) verschuiven. De visualisatie vorm wordt in een vroeg stadium van het project bepaalt, namelijk het onderzoek. Hierdoor kan ik van te voren het aantal iteraties plannen.

De keuze voor dit project is dus RUP[12]. Scrum geeft te veel ruimte voor veranderende functionaliteiten en eisen. Echter zijn veel eisen al bekend, door de opdracht omschrijving. De vorm van de visualisatie is een mogelijk veranderende factor. Dit risico wordt gedekt door het onderzoek naar de visualisatie vorm. Verder geven de

artefacten van RUP de mogelijkheid om een duidelijk communicatie middel te genereren naar de opdrachtgever. Ook kan ik door recent gebruik van RUP efficiënt de benodigde artefacten kiezen. Verder kan ik van te voren een duidelijke inrichting worden gemaakt van de iteraties. Immers verwacht de opdrachtgever eerst een data flow te kunnen bekijken alvorens deze kan worden aangepast of aangemaakt. Ook voorziet RUP de mogelijkheid om de betrokkenheid van de opdrachtgever ten diensten te doen.

5. Inception

Dit hoofdstuk laat de gekozen fasering naar voren komen. Hierbij laat ik zien op welke wijze het project vorm heeft gekregen. Bovendien worden de verwachte artefacten van RUP per fase naar voren gebracht. Het hoofdstuk sluit af met de projectplanning. Deze planning laat zien op welke momenten de verschillende producten uit de fasen worden verwacht.

5.1 Fasering

De gekozen software ontwikkelmethode bestaat uit een aantal fasen. In dit hoofdstuk worden de verschillende fasen van het project naar voren gebracht. Van elke fase wordt beschreven welke globale werkzaamheden er worden gedaan. Verder wordt er per fase de artefacten[13] van RUP naar voren gebracht.

Inception fase

In de eerste fase van het project zal er een plan van aanpak worden geschreven. Hiermee wordt de structuur van het project vastgesteld. Verder zal hiermee een planning worden geschreven en de op te leveren producten te worden naar voren gebracht. Zodra het plan van aanpak goed gekeurd is door de opdrachtgever, kan het project worden gestart door de opdrachtnemer.

Op te leveren producten aan het einde van Inception fase:

Product naam	Artefact RUP
Plan van aanpak	Iteratie plan (planning)
	Risico lijst
	Business model (huidige situatie)
	Project scope

Onderzoeksfase (Onderdeel van inception fase)

Tijdens dit project zal er onderzoek plaatsvinden. Dit onderzoek heeft betrekking op de bestaande vormen van dataflowvisualisatie. Ook moeten hierbij de bestaande software en code library worden meegenomen. Door dit onderzoek te doen wordt achterhaalt of er bestaande oplossingen zijn voor dataflowvisualisatie en of eventueel gebruik gemaakt kan worden van reeds bestaande oplossingen. Om de richting van het onderzoek te bepalen wordt de opdrachtgever benaderd. Hieruit is het mogelijk om requirements op te stellen. Verder wordt in deze fase de te gebruiken databronnen en datawarehouse vastgesteld. De kwaliteit en toepasbaarheid van de bestaande metadata beoordeelt. Op deze manier kan de metadata eventueel, bij ontbrekende of missende data, worden vervangen voor testdata.

Op te leveren producten aan het einde van Onderzoeksfase:

Product naam	Artefact RUP
Onderzoeksrapport	Requirements
Requirements rapport v1.0	Initiële use case diagram
	Requirements

Elaboration fase 1

Om vast te stellen welke functionaliteiten en eisen het eindproduct zal moeten krijgen wordt er gebruik gemaakt van requirements. Deze eisen worden aan de hand van interviews en gesprekken met de opdrachtgever opgesteld. Ook worden de gevonden requirements uit het onderzoek besproken met de opdrachtgever. De resultaten uit het onderzoek worden verder geïmplementeerd in het ontwerp van de data flow visualisatie module. Denk hierbij vooral aan de te gebruiken code library's.

Op te leveren producten aan het einde van Elaboration fase 1:

Product naam	Artefact RUP
Requirements rapport v1.5	Use case diagram
	Requirements (functioneel en niet functioneel)
	Use case beschrijvingen
	Technische beperkingen
Functioneel ontwerp	Analyse klassendiagram
	Schermontwerpen
Technisch ontwerp	Tools
	Design klassendiagrammen
	Sequence diagrammen

Construction & testing fase

Na de ontwerp fasen wordt de bouwfase gestart. Hierbij wordt conform het functioneel- en technisch ontwerp de dataflowvisualisatie module gebouwd. Deze module zal na het bouwen worden getest aan de hand van de beschreven requirements. De testsoorten kunnen op een later stadium worden overeengekomen met de opdrachtgever.

Op te leveren producten aan het einde van fase:

Product naam	Artefact RUP
Data flow visualisatie module	Deployable versie
Testrapport	Fysiek testontwerp
	Testrapport

Elaboration fase 2

Nadat de opdrachtgever de eerste versie van de visualisatie module voor zich heeft gehad kunnen er nieuwe inzichten ontstaan. In deze fase worden de nieuwe inzichten en overgebleven requirements geëvalueerd. Hierbij wordt opnieuw besloten welke van de requirements moeten worden vervuld. Deze zullen daarna worden uitgewerkt in het ontwerp. Zodat deze in de hierna komende bouwfase kunnen worden gebouwd.

Op te leveren producten aan het einde van Elaboration fase 2:

Product naam	Artefact RUP
Requirements rapport v1.6	Use case beschrijvingen
Functioneel ontwerp	Analyse klassendiagram
Technisch ontwerp	Design klassendiagrammen
	Sequence diagrammen

Construction & testing fase

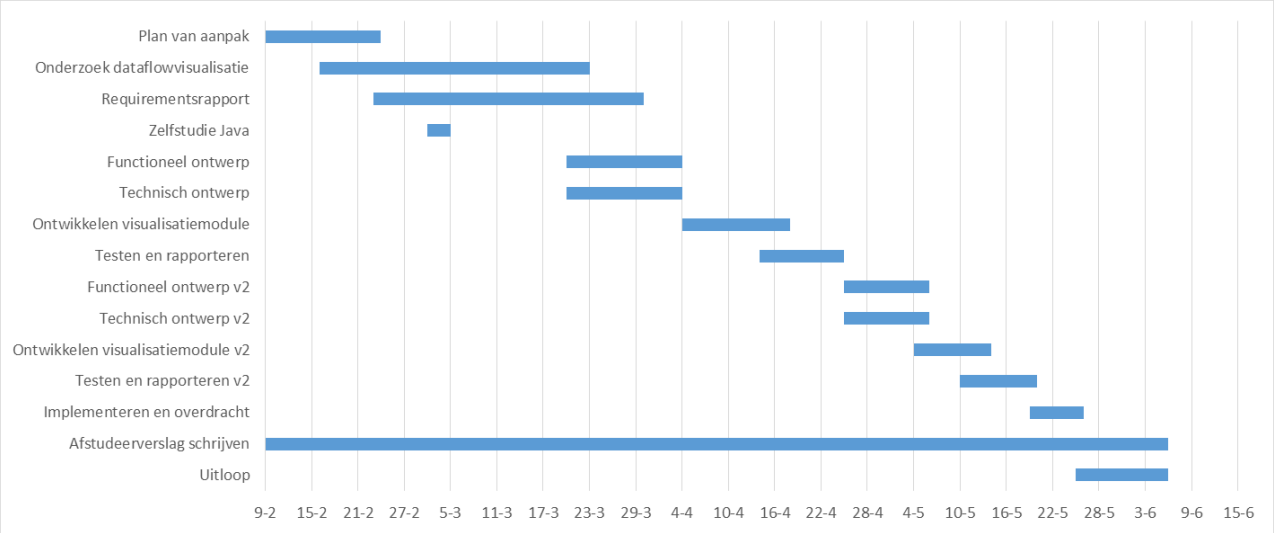
Na de ontwerp fasen wordt de bouwfase gestart. Hierbij wordt conform het functioneel- en technisch ontwerp de dataflowvisualisatie module gebouwd. Hierbij zullen de nieuwe requirements worden toegevoegd aan de bestaande visualisatie module. Deze module zal na het bouwen worden getest aan de hand van de beschreven requirements. Hierbij wordt gebruik gemaakt van de test set van de vorige construction fase.

Op te leveren producten aan het einde van fase:

Product naam	Artefact RUP
Data flow visualisatie module	Deployable versie
Testrapport	Fysiek testontwerp
	Testrapport

5.2 Project planning

Nadat het aantal iteraties is vastgesteld wordt in het plan van aanpak een project planning opgesteld. Deze planning heeft tot doel het overzichtelijk maken van het project. De planning geeft inzicht in de oplevermomenten van de overeengekomen producten. Dit zodat ik en de opdrachtgever een duidelijk beeld hebben over de voortgang van het project. Verder heb ik twee weken uitloop ingepland. Dit zodat er onvoorziene omstandigheden kunnen worden opgevangen.



Afbeelding 7, globale projectplanning

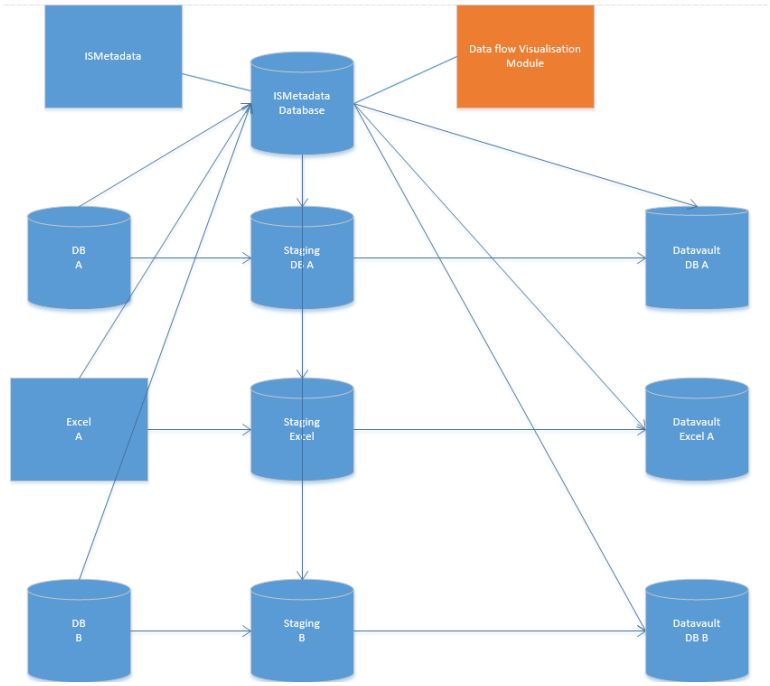
6. Onderzoeksfase

Uit de huidige situatie schets komt naar voren dat er een bestaande architectuur is voor het opzetten van een Business Intelligence systeem. Er wordt bij deze architectuur gebruikt gemaakt van een controle omgeving. In deze controle omgeving staat de ISMetadata tool. Deze tool is van belang voor dit project. Dit omdat de conceptuele mappingschema's worden opgeslagen in de database van de ISMetadata tool. Dit hoofdstuk zal inzicht geven in de werking van deze tool. Ook wordt er kort uitgelegd wat een data flow is. Door deze aspecten uit te zoeken kan aan de opdrachtgever een visualisatie vorm worden aangeraden. Verder wordt gekeken welke code library's deze visualisatie vorm kunnen ondersteunen.

6.1 Wat is ISMetadata?

Bij de huidige situatie wordt gebruik gemaakt van een controle omgeving. In deze controle omgeving staat de ISMetadata tool. Deze tool heeft een database met metadata. In deze metadata wordt de data flow beschreven. Om erachter te komen welke visualisatie vorm het beste past bij de metadata, is het van belang om de werking van de tool te begrijpen. Ook wordt op deze manier duidelijk op welke wijze de data flow visualisatie module onderdeel zal worden van de controle omgeving.

Om kennis op te doen over de werking van deze tool is de beschikbare documentatie van de ISMetadata tool doorgenomen. De werking is hieronder grafisch weergegeven. Hierbij is de data flow visualisatie module ook weergegeven om een beeld te krijgen van de wijze waarop de nieuwe tool in deze architectuur plaatsvindt.



Afbeelding 8, schets werking van ISMetadata

Hieronder wordt aan de hand van een Business Intelligence ontwerp proces de werking van de ISMetadata tool beschreven.

Het ontwerpen van een datavault wordt gedaan in een aantal stappen. De eerste stap is het identificeren van de bronsystemen. In bovenstaand figuur zijn dit “DB A”, “Excel A” en “DB B”. In werkelijkheid kunnen dit meer systemen zijn, ook kunnen er CSV files of andere vormen van dataopslag worden gebruikt als bronsysteem. Na de identificatie worden er staging area’s gemaakt voor de bronsystemen. Deze staging area’s worden gegenereerd door de ISMetadata tool en zijn qua structuur identiek aan de bronsystemen. Dit wordt gedaan door de structuur van de bronsystemen in te laden in de database van ISMetadata.

In de tweede stap wordt door ISMetadata, in het figuur weergegeven door “Metadata”, een analyse gedaan. Deze analyse wordt gedaan op het bronsysteem. Per bronsysteem ontstaat er een model voor de bijbehorende datavault. Deze modellen worden opgeslagen in de database van ISMetadata.

Hierna wordt het create script voor de datavault gegenereerd door ISMetadata. Dit wordt gedaan op basis van de modellen die zijn gegenereerd door ISMetadata. Ook worden de ETL schema’s gegenereerd voor de datavaults. Hierbij wordt de data gescheiden op statische informatie en informatie die mogelijk blijft veranderen. Dit zodat de data uit de staging area’s kunnen worden verplaatst naar de datavault.

Na deze stap wordt er door ontwerpers en ontwikkelaars een datawarehouse gemodelleerd. Dit gebeurd dus niet door het gebruik van ISMetadata tool. Nadat de datawarehouse gemodelleerd is kan de structuur worden ingeladen in ISMetadata. Hierbij wordt de structuur van de datawarehouse in de ISMetadata database opgeslagen. Door middel van een module binnen ISMetadata kan er met de hand de mapping worden aangemaakt. Dit zodat de attributen uit de datavault kunnen worden gekoppeld aan de bijbehorende attributen in het datawarehouse. Deze mappings worden hierna opgeslagen in de ISMetadata database.

De ISMetadata tool ondersteund het proces van ontwerpen van een Business Intelligence systeem. Er worden een aantal onderdelen daarvan opgeslagen in de database. De volgende onderdelen worden in de metadata database opgeslagen:

- Beschrijving van de structuur van bronsystemen, staging area’s en datawarehouse(tabellen en kolommen)
- Model beschrijvingen van de datavaults (tabellen, kolommen en relaties)
- Bewerkingen (inclusief bron- en bestemming attribuut) tussen de bronsystemen, staging area’s en de datavault
- Bewerkingen (inclusief bron- en bestemming attribuut) tussen de datavaults en het datawarehouse

De te ontwikkelen data flow visualisatie tool moet de metadata uit de ISMetadata database visualiseren. De data flow visualisatie module wordt dus gekoppeld aan de ISMetadata database.

6.2 Wat is een data flow?

In de ISMetadata database staan data flows beschreven. Om een duidelijk beeld te krijgen van een data flow visualisatie moet ik eerst weten wat een data flow is. Om overeenstemming hierin te vinden is dit aan de opdrachtgever gevraagd wat hij verstaat onder de term data flow. Hij beschrijft een data flow dan ook als volgt:

“Een data flow is een gegevens stroom. Hierbij gaat er data vanuit een bronsysteem naar een bestemmingssyteem. Hiervoor dient de data soms te worden bewerkt.”

Dit stemt overeen met de gegevens die ik heb gezien in de ISMetadata database. De conceptuele data flows in deze database worden beschreven op attribuut niveau. Er is daarom een aanname gedaan op basis van deze gegevens. Namelijk dat: een data flow moet op attribuutniveau van de databases worden gevisualiseerd.

Deze aanname is door de opdrachtgever bevestigd en gaf verder aan dat de volgende onderdelen in ieder geval in de data flow visualisatie naar voren dienen te komen:

1. bronattributen met naam, tabelnaam en databasenaam
2. bestemmingsattributen met naam, tabelnaam en databasenaam
3. bewerking die plaatsvindt op het bronattribuut alvorens deze het bestemmingsattribuut wordt

Om een data flow visualisatie tool te maken is het belangrijk om te weten welke visualisatie vorm er gewenst wordt. De opdrachtgever heeft bij het opstellen van het plan van aanpak gezegd dat hij geadviseerd wil worden over de vorm van de visualisatie. Hij gaf aan geen duidelijke vorm voor ogen te hebben. Er is daarom gekozen om een onderzoek te doen naar verschillende vormen van visualisatie voor een data flow. Verder is het nuttig om te onderzoeken welke code library's een conceptuele data flow visualisatie kunnen helpen realiseren.

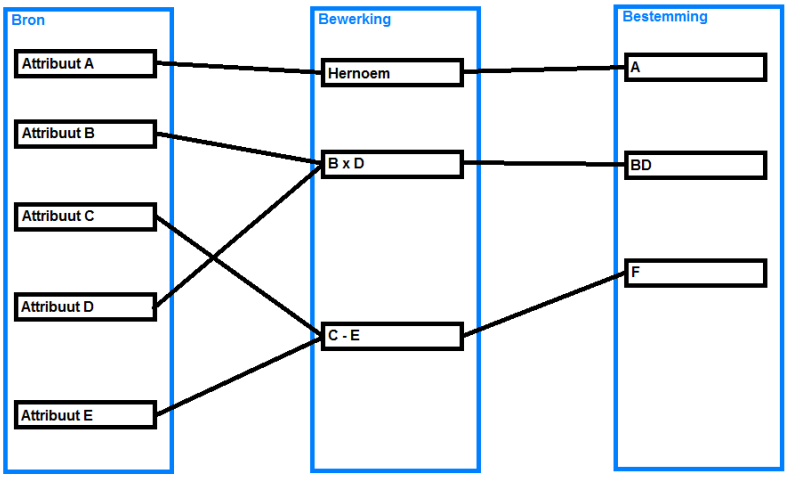
In de onderstaande hoofdstukken worden de belangrijkste resultaten getoond van dit onderzoek. Hierbij wordt de gekozen visualisatie vorm, met de argumentatie getoond. Verder worden de code library's, die naar voren zijn gekomen in het onderzoek besproken. Ook worden de criteria, waaraan de library's aan moeten voldoen, naar voren gebracht.

6.3 Welke vorm krijgt de data flow visualisatie?

In het onderzoek is onderzocht welke vormen van data flow visualisatie het beste passen bij de metadata in de ISMetadata database. De visualisatie moet een weergave geven van een conceptuele data flow. Dit betekent dat er conceptueel de flow van een database attribuut wordt getoond.

De vorm, die hierbij het beste past, is het *bron-bewerking-bestemming* model. Hierbij worden de drie onderdelen van een data flow naar voren gebracht. Dit model is als de meest bruikbare vorm naar voren gekomen. Ik heb hierbij de opdrachtgever overtuigd van het gebruiken van deze vorm. Dit is gedaan door middel van een toegepast voorbeeld.

Dit voorbeeld is het onderstaande figuur, met de volgende uitleg: de data wordt vanuit de bron gehaald en naar de bewerking gebracht. In de bewerkingsfase worden er een of meerdere aanpassingen gedaan (zoals omzetten in een andere eenheid, optellen van meerdere dataregels). Nadat de bewerking voltooid is wordt de data naar de bestemmingsfase gebracht. De bestemmingsfase is de eindfase waarin de data verkeerd. Hier is de data naar wens aangepast en kan deze worden gebruikt voor het doeleinde van deze data.



Afbeelding 9, bron, bewerking, bestemming model

Om de opdrachtgever verder te kunnen overtuigen zijn de voordelen van het bron, bewerking, bestemming model naar voren gebracht. Hierbij zijn de volgende voordelen aangedragen:

- Overzichtelijk
- Duidelijke scheiding tussen de verschillende fasen
- Er is een begin en een eind aan de data flow
- Alle onderdelen van de metadata komen voor in dit model (bronattribuut, bewerking, bestemmingsattribuut en een richting in de data flow)

Echter is er ook een nadeel aan dit model:

- Het gebruik van meerdere bronnen, bewerkingen en bestemmingen kan voor een omvangrijke grafische weergave zorgen

Voor de andere data flow visualisatie vormen kan het onderzoeksrapport worden nageslagen.

6.4 Code library's

Om een data flow te visualiseren is er een visualisatie vorm aangeraden. De data flow, die gevisualiseerd dient te worden, is op conceptueel niveau. In de ISMetadata database worden de data flows opgeslagen. Met deze opgeslagen data flow is de wens om in de toekomst een module te maken, die van de conceptuele data flow omzet naar data migratie packages. Om platform onafhankelijke data migratie packages te kunnen generen wordt er gewerkt met de ISMetadata database. De tool, die de data migratie packages moet maken, is nog een vervolg project dat moet worden gestart. Om op een efficiënte manier de data flows aan te passen wordt de data flow visualisatie tool gemaakt in dit project.

Om een data flow visualisatie tool te maken moet er een applicatie worden gemaakt, die kan samenwerken met de database van ISMetadata. De opdrachtgever gaf aan de voorkeur te hebben naar een WPF applicatie. Met dit verzoek ben ik de haalbaarheid van het project gaan analyseren. Het gebruik van WPF zou voor mij de eerste keer zijn en kan daardoor voor vertraging zorgen in het ontwikkelproces.

Bij deze analyse kwam naar voren dat het gebruik van WPF de haalbaarheid van het project in het geding brengt. Mijn kennis en gebruik van het het .Net framework is beperkt. Om de verschillende onderdelen van het .Net

framework te kunnen gebruiken en combineren zal naar mijn inzicht te veel tijd kosten. Deze tijd zal ten kosten gaan van ontwikkeltijd in het project. Met deze argumenten heb ik de opdrachtgever overtuigd om een Java applicatie te ontwikkelen.

Met dit besluit kan er gericht worden gezocht naar code library's. Deze code library's zijn van belang om de data flow effectief te kunnen visualiseren. Binnen Java is JavaFX de opvolger van Swing om grafische applicaties te maken. [14] Om een data flow effectief te kunnen visualiseren kan het gebruik van een code library de data flow visualisatie module ondersteunen. Een code library kan het ontwikkelen van de data flow visualisatie ondersteunen.

Een code library moet voldoen aan een aantal eisen om gebruikt te kunnen worden. Allereerst is er binnen Info Support een aantal project eisen. Deze eisen zijn als volgt:

- Er moet gebruik gemaakt worden van een stabiele release
- Er moet actief worden ontwikkeld aan de library

Ook zijn er een aantal eisen die ik stel aan een code library. Een data flow wordt in de bron bewerking bestemming vorm gepresenteerd. Deze vorm is een grafische 2d presentatie van een data flow. De code library moet dus een 2d visualisatie kunnen geven van een data flow. Het gebruik van een 3d engine is dus overbodig en zal buiten de scope van dit project vallen. Verder moeten de code libraries een mogelijkheid hebben om een layout te kunnen plaatsen. Deze layout moet de drie elementen(bron, bewerking en bestemming) van het model logisch kunnen plaatsen.

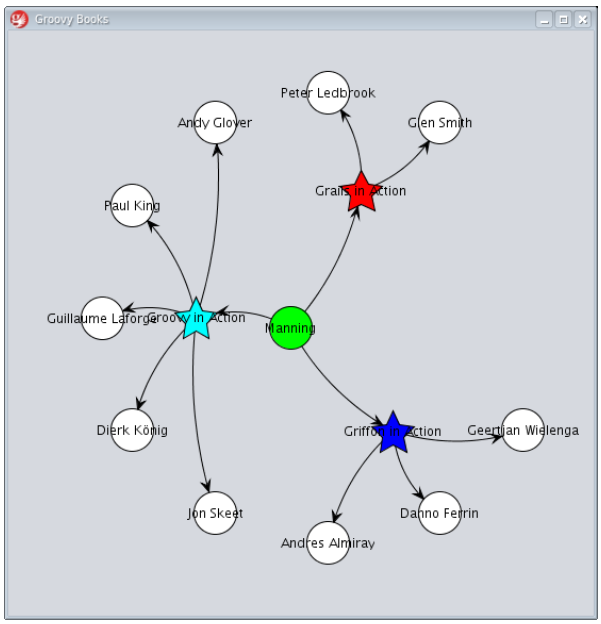
De criteria waaraan een code library moet voldoen zijn als volgt:

- Er moet een stabiele release zijn
- Er moet actief aan de library worden doorontwikkeld
- De library moet voor Java zijn
- De library moet geen 3d engine bevatten
- De library moet een visualisatie kunnen maken met het bron bewerking bestemmingsmodel

Tijdens het onderzoeken zijn een aantal code library's afgevallen op de bovenstaande criteria. Zo is JGraphT een code library waarbij er nog geen stabiele release is. Er is alleen een bèta versie te verkrijgen. De project eisen sluiten deze library uit en daarom is deze library niet meegenomen in de resultaten.

Ook is de code library BFG(Big Faceless Graph) afgevallen voor dit onderzoek. Een 2d weergave is voor de bron bewerking bestemming vorm voldoende. De BFG library is ontworpen om door middel van een 3d engine een weergave te geven. Op de voorgaande criteria valt deze code library af voor dit onderzoek.

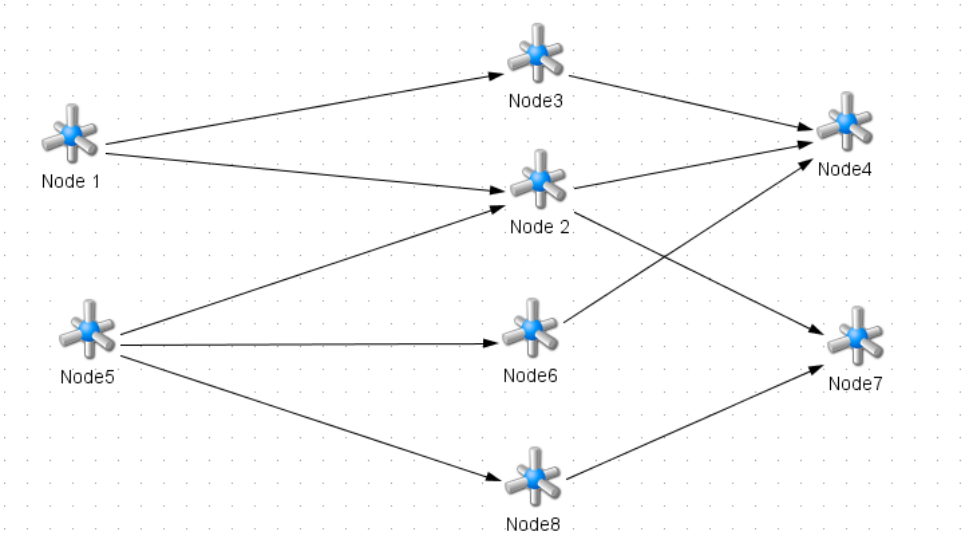
Een code library die wel voldoet aan de bovenstaande criteria is JUNG. Deze library wordt op dit moment actief onderhouden door de ontwikkelaars en heeft een stabiele release. JUNG is ervoor bedoelt om complexe datasets te visualiseren. Er is een groot aantal algoritmes aanwezig om op een logische wijze te kunnen presenteren. Een paar voorbeelden van algoritme namen zijn: tree-layout, radial-layout en balloon-layout. Verder kunnen er door gegevens dynamisch in te laden een voor gedefinieerde grafieken worden getoond. Deze grafieken zijn na het plotten statisch en kunnen alleen opnieuw worden geplot met nieuwe gegevens. De onderstaande afbeelding geeft een voorbeeld van een grafiek gemaakt met de hulp van JUNG.



Afbeelding 10, voorbeeld van een grafiek gemaakt met JUNG

Na het analyseren van de JUNG library is er verder gezocht naar andere mogelijke library's. Hierbij kwam Netbeans Visual naar voren. Deze library wordt gemaakt en ondersteund door de makers van Netbeans. Er is op dit moment een stabiele release te verkrijgen en er wordt nog actief doorontwikkeld aan deze library.

Netbeans Visual maakt gebruik van componenten uit JavaFx. Verder bevat deze library het zogenoemde Widget component. Dit is een grafisch element, dat zichzelf kan tekenen. Het widget element wordt in onderstaand afbeelding gebruikt.



Afbeelding 11, voorbeeld van een visualisatie gemaakt met Netbeans visual

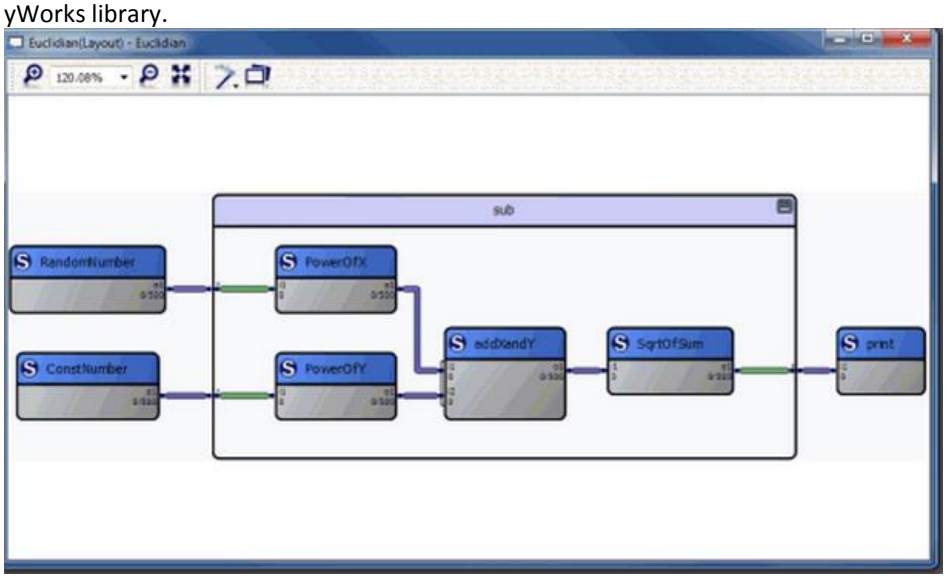
In de afbeelding zijn de nodes gemaakt door gebruik van de Widget objecten. Verder is duidelijk te zien dat Widgets door middel van pijlen aan elkaar verbonden kunnen worden. Ook hebben Widget objecten actionListeners. Dit

betekent dat er op geklikt, aanpassen, versleept en verwijderd kunnen worden. Netbeans Visual heeft in vergelijking met JUNG een beperkt aantal layouts om de data logisch te plaatsen.

De laatste geanalyseerde library is de yWorks library. Deze library heeft een actieve ondersteuning en een stabiele release. Om gebruik te mogen maken van de yWorks library moet er jaarlijks licentie kosten worden betaald. Door dit te doen kunnen volgens de makers de volgende functionaliteiten worden gebruikt:

- Netwerk analyse en visualisatie
- Business proces modeling
- Data mining functions (analyse van log files)
- Database management en modellering
- Sociale netwerk plug-ins
- UML diagrammen genereren en aanpassen
- Flow chart generen en aanpassen
- 3D visualisaties
- Visueel programmeren

De volgende afbeelding illustreert een applicatie die een diagram toont, deze is gemaakt met de hulp van de yWorks library.



Afbeelding 12, voorbeeld van een data flow diagram gemaakt met yWorks

Om een duidelijk overzicht te geven van de onderzochte code library's is er een tabel met de voor- en nadelen gemaakt. Deze tabel geeft alleen de voor- en nadelen van de library's met een huidige ondersteuning en een stabiele release.

Library naam	Voordelen	Nadelen
Netbeans Visual [15][16]	-Stabiele release beschikbaar -Actieve doorontwikkeling -Bevat geen 3d engine -Gebruikt JavaFX componenten - Widget objecten met dynamische eigenschappen, zoals children objecten, listeners en pijlen	- Beperkt aantal layout mogelijkheden, grid en decision tree (layout is de wijze waarop de elementen worden gestructureerd bij de weergave) - bron bewerking bestemming model niet direct ondersteund (deze moet zelf worden gemaakt)
Jung [17]	-Stabiele release beschikbaar -Actieve doorontwikkeling -Bevat geen 3d engine - Grote diversiteit aan layouts beschikbaar	- Beperkte grafische elementen (alleen basis vormen zoals cirkels, vierkanten sterren) -Grafisch niet aantrekkelijk
yWorks [18]	-Stabiele release beschikbaar -Actieve doorontwikkeling	- Jaarlijkse betaalde licentie - Onduidelijke ondersteuning bron,bewerking,bestemming -Bevat een 3d engine(hoeft overigens niet te worden gebruikt)

6.5 Conclusie

De data flow visualisatie tool moet samenwerken met de database van ISMetadata. De ISMetadata tool slaat de metadata van verschillende systemen op. Uit de documentatie blijkt de volgende metadata te worden opgeslagen in de ISMetadata database:

- Beschrijving van de structuur van bronsystemen, staging area's en datawarehouse(tabellen en kolommen)
- Model beschrijvingen van de datavaults (tabellen, kolommen en relaties)
- Bewerkingen (oftewel Mappings) tussen de bronsystemen, staging area's en de datavault (op attribuutniveau)
- Bewerkingen (oftewel Mappings) tussen de datavaults en het datawarehouse (op attribuutniveau)

Uit de requirements analyse komt naar voren dat een data flow uit de volgende onderdelen bestaat:

1. bronattributen met naam, tabelnaam en databasenaam
2. bestemmingsattributen met naam, tabelnaam en databasenaam
3. bewerking die plaatsvindt op het bronattribuut alvorens deze het bestemmingsattribuut wordt

Deze onderdelen staan volledig beschreven in de database van ISMetadata. Daarom wordt de data flow visualisatie module aangesloten op deze database. De data flow wordt in het bron bewerking bestemmingsmodel getoond. Dit model is bevat de drie onderdelen en hun relaties tot elkaar.

Om een conceptuele data flow te visualiseren wordt er een applicatie in Java geschreven. Deze applicatie kan gebruik maken van de Netbeans Visual library en de JUNG library om de visualisatie te ondersteunen. Deze libraries kunnen het ontwikkelproces een positieve invloed geven.

7. Elaboration 1

Nadat het onderzoek is afgerond is er begonnen aan het ontwikkelen van de data flow visualisatie tool. In de onderzoeksfase zijn al een aantal eisen en wensen naar voren gekomen. Deze zullen in de eisen en wensen (requirements) fase worden meegenomen. Verder zal het advies uit het onderzoek worden meegenomen tijdens het ontwerpen van de data flow visualisatie tool.

Om een beeld te krijgen van de eisen en wensen van de stakeholder, in dit geval de opdrachtgever, is er een requirements analyse uitgevoerd. In dit hoofdstuk wordt begonnen met het beschrijven van dit proces. Deze eisen en wensen dienen als basis voor het ontwerp. Het ontwerp van de eerste versie van de data flow visualisatie tool wordt hierna naar voren gebracht. Hierbij zullen gemaakte keuzes en het proces in kaart worden gebracht. Dit zodat er een ontwerp ontstaat dat in de construction fase kan worden uitgewerkt.

7.1 Eisen en wensen

Aan het begin van het project is het huidige probleem geschetst. Hierbij is naar voren gekomen dat het gebrek aan grafische weergave van een data flow belemmering geeft aan de werkzaamheden van een ontwerper of ontwikkelaar geeft. Om dit probleem op te lossen wordt in dit project een data flow visualisatie tool ontwikkeld. Met deze tool moet het overzicht van een data flow grafisch worden weergegeven.

De data flow visualisatie moet dus duidelijkheid geven aan de ontwerpers en ontwikkelaars. Om de exacte functionaliteiten te kunnen vaststellen worden de eisen en wensen, vanaf nu requirements genoemd, achterhaald. Dit wordt gedaan door contact op te nemen met de stakeholder van dit project, de opdrachtgever.

Tijdens deze requirements analyse is er een risico tot uiting gekomen. De opdrachtgever had een vol werkschema. Contact met de opdrachtgever was erg minimaal, zowel telefonisch als via de email. Het wachten op reactie en een contact moment inplannen kostte hierdoor meer tijd. Om dit risico niet verder te laten escaleren zijn er twee partijen op de hoogte gebracht van de situatie.

Hierbij is de procesbegeleider, Pascalle Hijl, ingeschakeld binnen Info Support. Zij heeft mij toegang verschaft tot het werkschema van de opdrachtgever. Deze heb ik daarna onder de loep genomen en dit heeft een aantal momenten naar voren gebracht waarbij contact mogelijk was. Op deze momenten heb ik contactmomenten ingepland om ervoor te kunnen zorgen de vertraging te beperken. Ook is de begeleidend examinator, Arno Nederend, ingelicht van de situatie. Op deze manier zijn de twee betrokken partijen, Info Support en de Haagse Hogeschool, op de hoogte van de situatie en kon er een passende oplossing worden gezocht.

Om het probleem, zeer beperkte tijd, op te lossen is er gewerkt met een brainstormfase en een prioriteringsfase. Hieronder is de inrichting van deze twee fasen beschreven.

7.1.1 Brainstorm

De brainstormfase wordt van start gegaan door de requirements uit de opdracht omschrijving en het onderzoek te noteren. Dit zijn de onderstaande requirements:

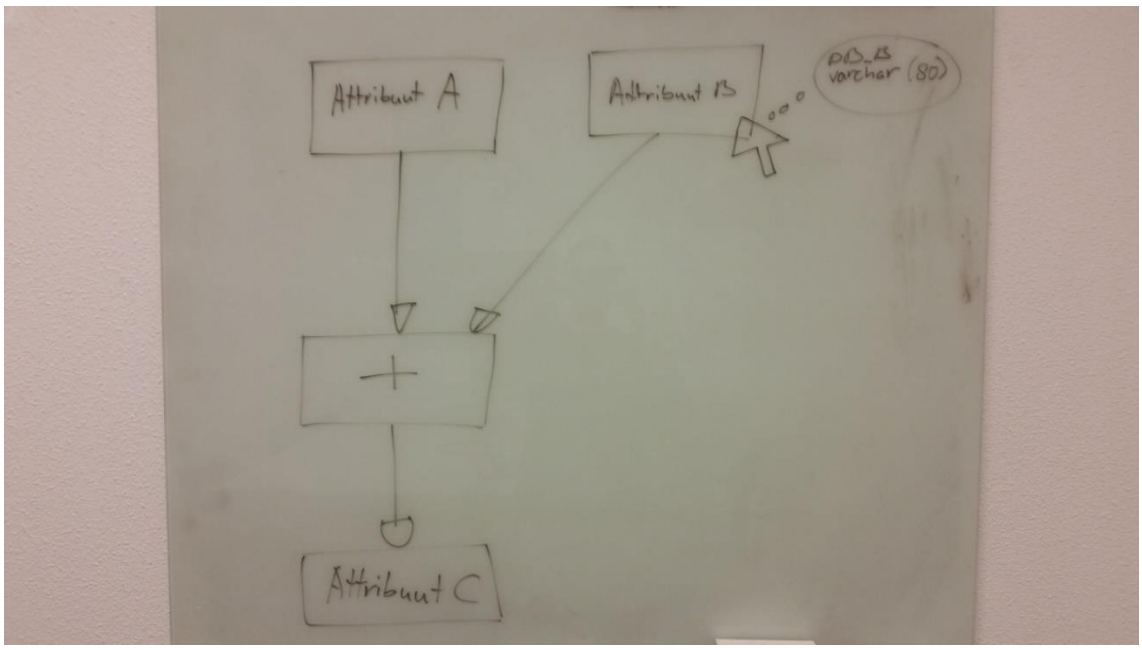
Requirement ID	Requirement beschrijving
UR 1	De data flow moet op attribuut niveau worden gevisualiseerd
UR 2	De brontabellen moeten worden kunnen herleid uit de visualisatie
UR 3	De bestemmingstabellen moeten worden kunnen herleid uit de visualisatie
UR 4	Attributen worden weergegeven in de data flow met een symbool
UR 5	De bewerkingen worden weergegeven met unieke symbolen
UR 6	De bewerkingen moeten kunnen worden aangepast
UR 7	De bewerkingen moeten kunnen worden verwijderd uit de visualisatie
UR 8	Er moeten bewerkingen worden kunnen toegevoegd aan de data flow
UR 9	De flow van een attribuut moet kunnen worden aangepast
UR 10	Attributen moeten kunnen worden verwijderd uit de data flow. Zodat deze niet worden meegenomen in de volgende bron

Ter voorbereiding van deze sessie worden de reeds gevonden requirements teruggekoppeld. De hierboven genoemde requirements worden teruggekoppeld zodat de opdrachtgever kan beslissen of deze requirements nog steeds van toepassing zijn. Daarna wordt kort verwezen naar het advies uit het onderzoek. Hiermee doelend op de vorm van de visualisatie. Zodat de opdrachtgever eventueel op ideeën gebracht kan worden over de vorm van de visualisatie. Als laatste onderdeel van de sessie is brainstormen. Het programma voor de brainstorm sessie is het bespreken van de reeds bekende requirements en de geldigheid daarvan. Daarna wordt er een brainstormfase gehouden waarin alle (nieuwe) ideeën over het systeem naar voren kunnen worden gebracht.

Tijdens de brainstormfase worden de volgende vragen gesteld om de requirements te kunnen specificeren:

1. Wat voor soorten bewerkingen zijn er in een data flow?
2. Met welk symbool moet een attribuut worden weergegeven?
3. Moeten bewerkingen kunnen worden aangepast, toegevoegd en verwijderd worden uit een Data Flow?
4. Moet een attribuut (bron of bestemming) worden verwijderd uit de dataflow?
5. Zijn er nog functionaliteiten vergeten, die wel van belang zijn om te uiten?

Hierbij komt naar voren dat de reeds gevonden requirements nog steeds van toepassing zijn, maar specifieker geformuleerd moeten worden. De laatste vraag start de brainstormfase. Tijdens deze fase wordt een kleine vorm van data flow visualisatie gedaan door op het bord een voorbeeld van de opdrachtgever te tekenen. Aan de opdrachtgever wordt gevraagd: hoe zie jij een simpele data flow visueel voor je, denkende aan de resultaten van het onderzoek? Daarbij heeft de opdrachtgever de volgende tekening gemaakt:



Afbeelding 13, requirements in een schets

Uit de tekening van de opdrachtgever zijn de drie onderdelen wederom duidelijk te onderscheiden. Ook de vorm van de data flow is door deze tekening duidelijk naar voren gebracht. Een belangrijk punt is dat een pijl de richting van de data flow aangeeft.

Nadat de sessie heeft plaatsgevonden worden de gegevens op het kladblok uitgewerkt in een requirements lijst. Onderstaande lijst is het resultaat van de brainstormfase:

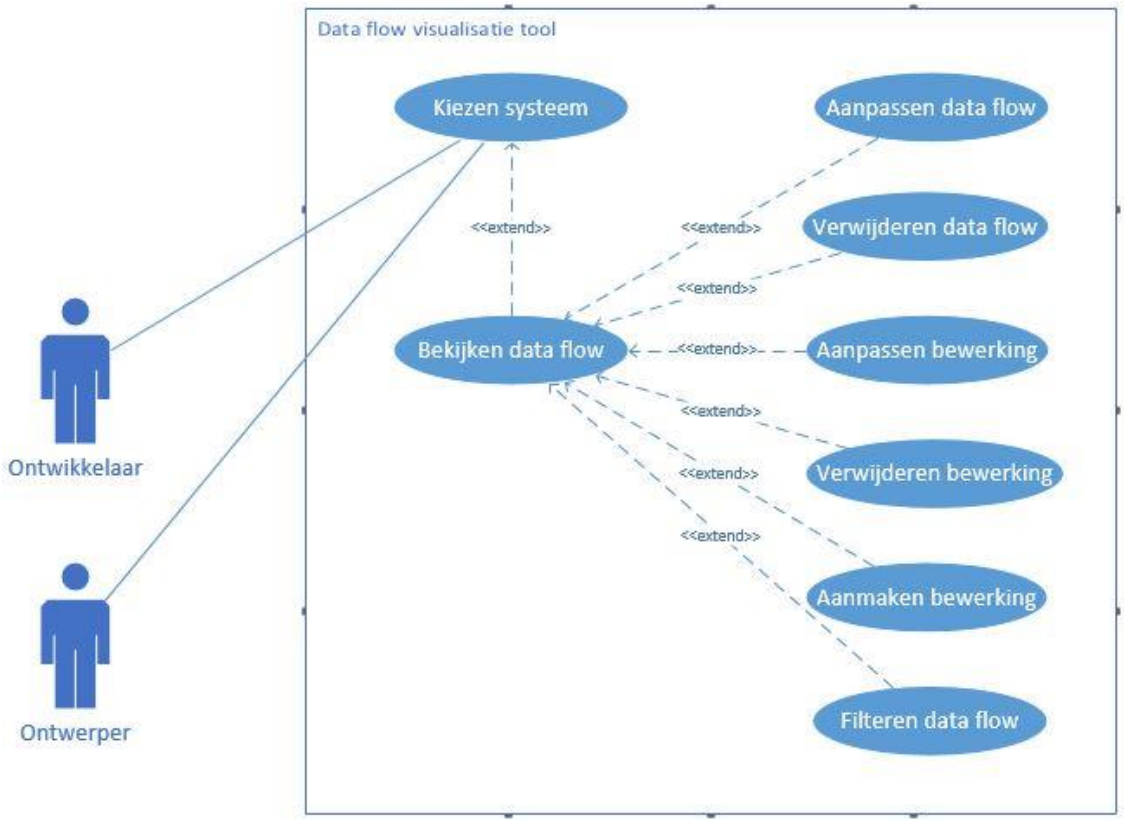
Requirement ID	Requirement beschrijving
UR 1	Er kan een systeem worden gekozen uit de ISMetadata database om te visualiseren
UR 2	De data flow moet op attribuut niveau worden gevisualiseerd
UR 3	De brontabellen en databasenaam moeten worden kunnen herleid uit de visualisatie
UR 4	De bestemmingstabellen en databasenaam moeten worden kunnen herleid uit de visualisatie
UR 5	Attributen worden weergegeven in de data flow met een rechthoek
UR 6	De flow van een attribuut moet kunnen worden aangepast
UR 7	De data flow van een attribuut moet kunnen worden verwijderd
UR 8	De aggregatie bewerkingen kunnen worden getoond
UR 9	De aggregatie bewerkingen kunnen worden aanpast
UR 10	De aggregatie bewerkingen kunnen worden verwijderd uit de visualisatie
UR 11	De aggregatie bewerkingen kunnen worden toegevoegd aan de data flow
UR 12	De scalar bewerkingen kunnen worden getoond
UR 13	De scalar bewerkingen kunnen worden aangepast
UR 14	De scalar bewerkingen kunnen worden verwijderd uit de visualisatie
UR 15	De scalar bewerkingen kunnen worden toegevoegd aan de data flow
UR 16	De attributen in de visualisatie kunnen worden gefilterd op schemanaam en tabelnaam
UR 17	De data flow visualisatie wordt in een layout geplaatst. De attributen en transformaties worden op een nader te specificeren wijze op het scherm geplaatst
UR 18	De door de gebruiker gemaakte layout kan worden opgeslagen

7.1.2 Prioritering

Voordat de prioritering van start gaat worden de niet functionele requirements opgesteld. De niet functionele eisen beschrijven de eisen aan het systeem, die gebruikt kunnen worden om het gedrag te kunnen beoordelen. Uit het onderzoek en afspraken uit het plan van aanpak zijn de volgende niet functionele eisen naar voren gekomen:

Requirement ID	Requirement beschrijving
NFR 1	De data flow visualisatie tool wordt geschreven in het Java platform
NFR 2	De data flow visualisatie tool moet gebruik kunnen maken van Microsoft SQL server
NFR 3	De data flow visualisatie tool moet metadata interpreteren van de ISMetadata database
NFR 4	De data flow visualisatie tool moet binnen 2 seconden een data flow tonen
NFR 5	Bij code libraries moeten gebruik maken van een stabiele release
NFR 6	Bij code libraries moet er een actieve ontwikkelaar zijn voor de library

Na het opstellen van de niet functionele requirements wordt op basis van de functionele requirements een use case diagram opgesteld. In dit use case diagram worden de gebruikers scenario's opgesteld. Deze ontstaan door het categoriseren van de requirements. In het use case diagram wordt verder de verhoudingen tussen de gebruikers scenario's aangegeven.



Afbeelding 14, use case diagram

Uit het use case diagram wordt duidelijk dat er twee use cases zijn die leiden tot de overige use cases. Tijdens het prioriterings gesprek is het belangrijk om aan te geven dat deze gebruikersscenario's van essentieel belang zijn. De use cases verbonden met deze scenario's worden door mij aan de opdrachtgever aangegeven als belangrijkste use cases. Aangezien deze als basis dienen voor de andere use cases.

De requirements worden geprioriteerd aan de hand van de MoSCoW methode. Aan de opdrachtgever wordt duidelijk gemaakt dat het project een tijd beperkende factor heeft. Hierdoor is niet mogelijk om alle requirements binnen de scope van het project af te werken. Door prioriteit te geven aan de requirements kunnen de meest essentiële functionaliteiten worden vastgesteld. Zodat er een product ontstaat dat bruikbaar is. In de MoSCoW methode worden vier niveaus van prioriteit gehanteerd. Dit zijn de volgende niveaus:

- M - must haves: deze requirements moeten in het eindresultaat terugkomen, zonder deze requirements is het product niet bruikbaar;
- S - should haves: deze requirements zijn zeer gewenst, maar zonder is het product wel bruikbaar;
- C - could haves: deze requirements zullen alleen aan bod komen als er tijd genoeg is;
- W – won't haves: deze requirements zullen in dit project waarschijnlijk niet aan bod komen maar kunnen in de toekomst, bij een vervolgproject, interessant zijn.

Uit de prioritering sessie komen de volgende prioriteiten naar voren(voor de prioriteit per requirement zie bijlage D):

Use Case ID	Use case naam	Beschrijving	Requirement	Prioriteit
UC 1	Kiezen systeem	Het kiezen van het gewenste systeem om te visualiseren uit een lijst die wordt gegenereerd uit ISMetadata database	UR 1	M
UC 2	Bekijken data flow	Het bekijken van de volledige data flow visualisatie	UR 2, UR 3, UR 4, UR 5, UR 8, UR 12, UR 17	M
UC 3	Aanpassen data flow	Het aanpassen van de flow van een attribuut in de visualisatie	UR 6	S
UC 4	Verwijderen data flow	Het verwijderen van de flow van een attribuut in de visualisatie	UR 7	C
UC 5	Aanpassen bewerking	Het veranderen van een bestaande bewerking	UR 9, UR 13	C
UC 6	Verwijderen bewerking	Het verwijderen van een bewerking op attribuut/attributen	UR 10, UR 14	C
UC 7	Aanmaken bewerking	Toevoegen van een nieuwe bewerking op een attribuut of attributen	UR 11, UR 15	C
UC 8	Filteren data flow	De attributen van een data flow filteren op schemanaam of tabelnaam	UR 16	W

Van de use cases, waarvan de requirements met een must have prioriteit hebben, worden uitgewerkt in een use case beschrijving. Dit om overeenstemming te krijgen met de opdrachtgever over de gebruikers scenario's. In deze use case beschrijvingen wordt de interactie tussen actor en systeem in kaart gebracht. De use cases laten verder zien op welke wijze de requirements tot uiting komen in het systeem.

Onderstaand zijn de use case beschrijvingen van de use cases met de requirements van de hoogste prioriteit:

Naam	Kiezen systeem
ID	UC 1
Beschrijving	Het kiezen van het gewenste systeem om te visualiseren uit een lijst die wordt gegenereerd uit ISMetadata database
Primaire actor(s)	Ontwikkelaar of ontwerper (Actor)
Secondaire actor(s)	-
Pre-conditie(s)	-
Scenario beschrijving	1. Actor start de visualisatie tool 2. Systeem haalt de lijst met beschikbare systemen op[1] 3. Systeem toont de lijst met beschikbare systemen 4. Actor selecteert het gewenste systeem 5. Systeem haalt de attributen met bijbehorende bewerkingen en bestemmingsattributen van het geselecteerde systeem op
Postconditie(s)	De actor heeft het gewenste systeem geselecteerd ter visualisatie
Alternatieve flow	[1] Geen beschikbare systemen 3.Systeem toont een lege lijst van systemen 4.Actor sluit de visualisatie tool af

Naam	Bekijken data flow
ID	UC 2
Beschrijving	Het bekijken van de volledige data flow visualisatie
Primaire actor(s)	Ontwikkelaar of ontwerper
Secondaire actor(s)	-
Pre-conditie(s)	Er is een beschikbaar systeem geselecteerd
Scenario beschrijving	1. Systeem haalt de gegevens van het geselecteerde systeem op 2. Systeem toont de data flow van het systeem van de geselecteerde systeem. Waarbij de attributen uit de brontabellen, de attributen uit het bestemmingsmodel, aggregatie bewerkingen en de flow van de data tussen bron en bestemming[1]
Postconditie(s)	De actor kan de gewenste data flow visualisatie bekijken
Alternatieve flow	[1] Geselecteerde systeem heeft nog geen data flow 2. Er wordt een lege data flow weergegeven

Deze use case beschrijvingen zijn toegevoegd aan het requirements document. Dit document is naar de opdrachtgever gestuurd. Hierbij de bovenstaande uitleg van een use case. Zodat de opdrachtgever kan aangeven of hij de interactie met het systeem ook op deze manier voor zich ziet. Hiermee wordt het requirements document afgesloten en ontwerpen gestart. Voor een volledig inzicht in het requirements document zie: bijlage D

7.2 Ontwerpen

Na het opstellen van de requirements ben ik begonnen met het ontwikkelproces. Door het vaststellen van de belangrijkste requirements komen de uit te werken functionaliteiten naar voren. Deze functionaliteiten zijn beschreven in use cases. De use cases worden in dit hoofdstuk uitgewerkt in een ontwerp. Dit ontwerp bevat de architectuur van de te maken data flow visualisatie.

Dit hoofdstuk geeft inzicht in de gemaakte keuzes bij het ontwerpen van de data flow visualisatie tool. Eerst wordt aangegeven op welke wijze de data flow visualisatie tool wordt opgebouwd. Hierna wordt er per onderdeel van de opbouw een implementatie model weergegeven en toegepast.

7.2.1 Packages

De data flow visualisatie tool bestaat uit een aantal onderdelen. Om een duidelijk onderscheid te maken tussen deze onderdelen is een package diagram opgesteld. In dit package diagram worden de verschillende onderdelen zo los mogelijk gekoppeld.

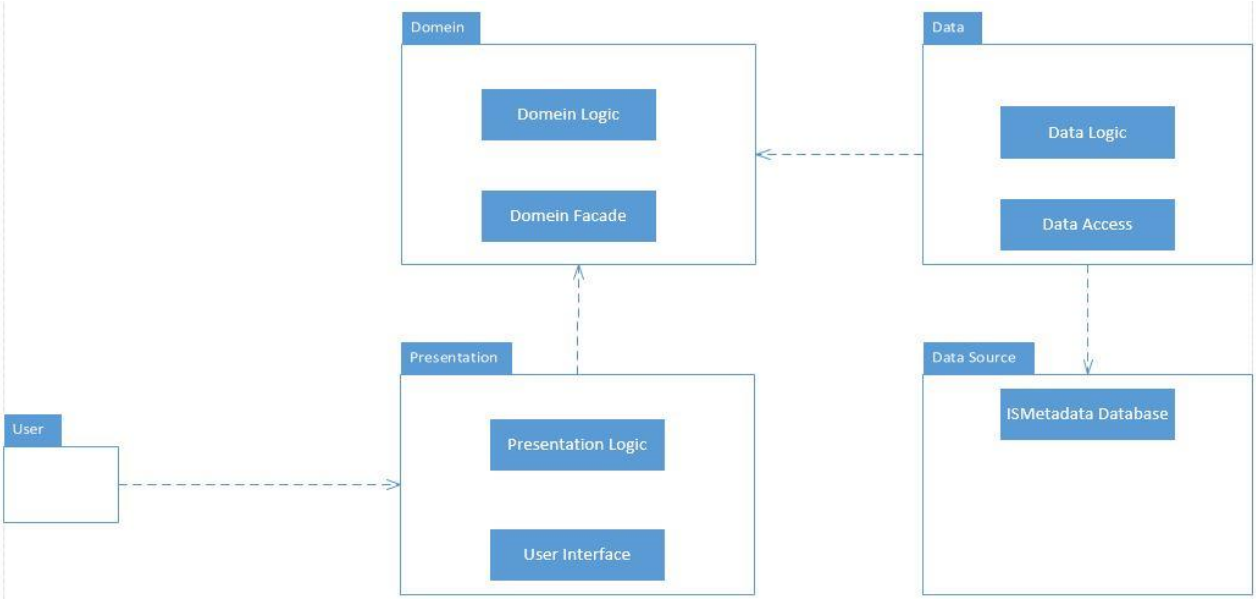
Het eerste onderdeel van de data flow visualisatie tool is een database. Deze database is onderdeel van de ISMetadata tool. Het aanpassen van deze database ligt buiten de scope van dit project. Dit omdat deze beslissingen zullen moeten worden overlegd met het team dat de ISMetadata tool heeft gemaakt. Deze zijn niet beschikbaar voor dit project en daarom is er gekozen om de database te gebruiken zoals deze nu is. De structuur van deze database wordt verderop in dit hoofdstuk naar voren gebracht.

Het tweede onderdeel is de front end. Deze front end is het gedeelte van de applicatie, die verantwoordelijk is voor het tonen van de gemaakte data flow. Hierbij wordt aan de gebruiker een visuele representatie getoond van de data flow.

Het laatste onderdeel heet het domein. Deze is verantwoordelijk voor het creëren van een data flow op basis van de verkregen data uit de database. Deze data wordt dan omgezet aan de hand van bepaalde business logica. De business logica bepaalt de vorm van de data flow. Om de vorm te bepalen wordt het resultaat van het onderzoek meegenomen. Waarbij de vorm bron, bewerking, bestemming wordt aangeraden. De opdrachtgever heeft aangegeven deze vorm overzichtelijk en duidelijk te vinden. Deze vormgeving hangt samen met de presentatie in de front end, maar ook in het domein moet de data op de gewenste wijze worden geïnterpreteerd om tot de visualisatievorm te komen.

Het onderstaande package diagram geeft de drie onderdelen en hun verhoudingen grafisch weer. De packages bevatten ieder een aantal klassen. Deze zullen worden uitgewerkt in het hoofdstuk implementatie. Deze reden dat gekozen is voor de packages is onderhoudbaarheid en uitbreidbaarheid. Door de verschillende onderdelen te scheiden is het mogelijk om een van de onderdelen te vervangen zonder dat de overige onderdelen te veel moeten worden aangepast. Er kan een andere front end (of meerdere front ends) worden opgezet, deze kunnen ieder gebruik maken van het domein. Zonder dat het domein hiervoor hoeft te worden aangepast. Ook kan de database veranderen of worden vervangen zonder dat het domein hoeft worden aangepast.

Onder het package diagram wordt per package uitgelegd uit welke onderdelen deze bestaat en de verantwoordelijkheden van ieder van de package en onderdelen.



Afbeelding 15, package diagram

Om een extra scheiding aan te brengen in het onderdeel database is gekozen voor een data source package. In die package zit de ISMetadata database. Dit is een extern onderdeel van de data flow visualisatie tool.

De data uit de database wordt gebruikt om een data flow te kunnen visualiseren. Het data package is binnen de applicatie verantwoordelijk voor het maken van een connectie met de database en het ophalen van de data. De connectie met de database wordt gedaan door de Data Access. In de Data Logic van dit package staat alle logica omtrent het ophalen van de data. Hierin worden dus de benodigde query's opgebouwd. Deze worden dan door de database connectie uitgevoerd op de database. Het resultaat wordt teruggegeven aan de Data Logic. Bij het veranderen van de database dient deze package te woorden aangepast. De connectie zal alleen moeten worden aangepast als deze gebruik gaat maken van een andere DBMS. Door de query logica te scheiden van het domein is de verantwoordelijkheid van het domein, dat het opstellen van een object georiënteerde data flow is, gescheiden van de logica van de database. Zo ontstaat er een losse koppeling tussen de database en de interne logica van de data flow visualisatie module.

De Data package is verbonden aan het Domein. Binnen het domein wordt de opgehaalde data uit de Data package geïnterpreteerd. Dit betekent dat de relationele data die wordt gegeven vanuit de Data Logic wordt omgezet naar objecten met behulp van de business logica. Het domein is alleen verantwoordelijk voor het maken van een data flow uit gegeven data. Door de data flow logica los te koppelen van de visualisatie wordt het domein alleen verantwoordelijk gemaakt voor het maken van een data flow van objecten. De manier waarop de data flow wordt opgebouwd komt voor uit business logica van de requirements analyse.

Er is eerder vastgesteld dat er drie verschillende onderdelen zijn in een data flow, namelijk: bronattributen, bewerkingen en bestemmingsattributen. Verder blijkt uit de getekende data flow bij de brainstorm sessie een aantal regels van een data flow. Deze regels zijn dus aannames uit de getekende data flow visualisatie.

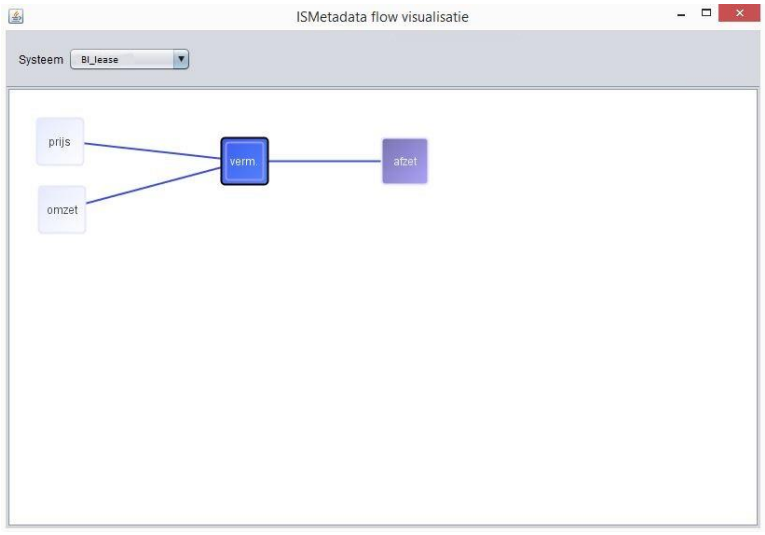
Regel Nr	Regel
1	Een bestemmingsattribuut heeft altijd een bewerking, ook al is deze bewerking leeg
2	Een bestemmingsattribuut kan ontstaan uit meerdere bronattributen. Voor een bronattribuut moet de naam, tabel en database en bijbehorende bewerking overeenkomen om dit vast te stellen

Regel Nr	Regel
3	Een bestemmingsattribuut kan ontstaan zonder bronattributen te hebben
4	Een bewerking kan meerdere bronattributen bewerken
5	Een bronattribuut kan meerdere bestemmingen hebben, deze worden als een nieuwe data flow gezien
6	Een bronattribuut kan geen data flow hebben naar een ander bronattribuut

De presentation package is verantwoordelijk voor het maken en tonen van de data flow visualisatie. Door deze logica te scheiden kan er op een later moment eventueel worden gekozen voor een andere visualisatie front end. Het domein is namelijk verantwoordelijk voor een data flow gemaakt van objecten. De logica van presenteren staat dus los van het domein en kan door deze loskoppeling mogelijk worden vervangen door een andere presentatie logica. De presentatie package is dan verantwoordelijk voor de visuele presentatie van deze objecten. Uit het domein wordt een lijst met objecten en tussenliggende relaties gegeven. De Presentation Logic gaat van de lijst toonbare objecten, de widget objecten uit de Netbeans Visual library, maken. Deze objecten worden dan door de Presentation Logic, op basis van de structuur van de data flow(gemaakt door het domein), op het scherm geplaatst. De gebruiker(ontwikkelaar of ontwerper) maakt gebruik van de User Interface. Deze wordt getoond bij het starten van de applicaties. In deze interface kan de gebruiker het systeem selecteren, waarvan de data flow moet worden getoond. Verder geeft de tekent de Presentation Logic op de User Interface de, uit het domein opgehaalde data flow.

7.2.2 Schermontwerp

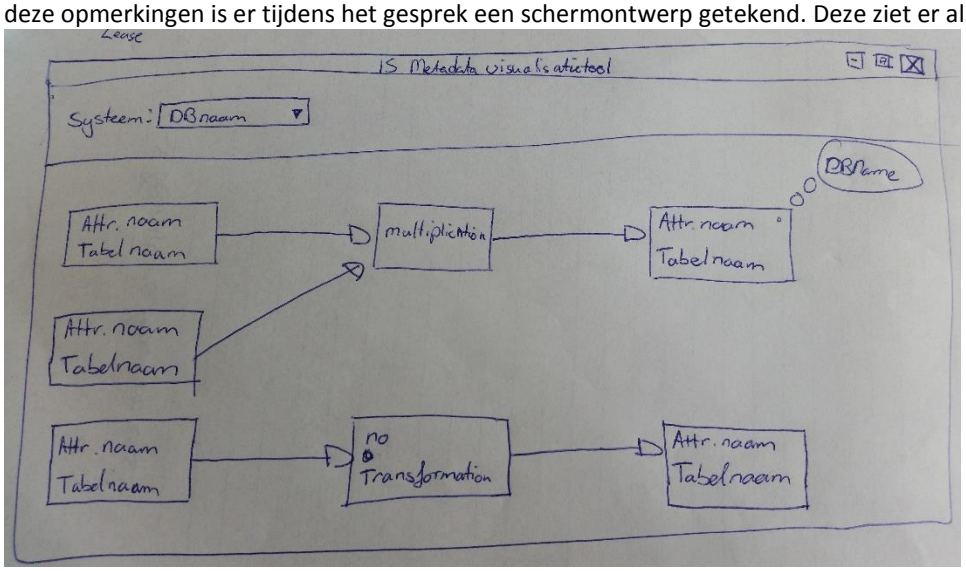
Om de presentatie laag een exacte vorm te geven heb ik gebruik gemaakt van schermontwerpen. Om de stakeholders een duidelijk beeld te geven is dat in dit project ook gedaan. Voor dit project is er een schermontwerp gemaakt met de Netbeans Visual library (voor beschrijving van deze library, zie onderzoek). Dit is gedaan door de getekende afbeelding (afbeelding 13) uit de requirements te analyseren en de gevonden regels hierin toe te passen. In combinatie met de geprioriteerde requirements en de opgestelde regels is daaruit het onderstaande schermontwerp gemaakt.



Afbeelding 16, schermontwerp

In een gesprek met de opdrachtgever is het schermontwerp doorgenomen. Hierbij is uitgelegd dat door het kiezen van een bestemmingssysteem in de dropdownbox de visualisatie wordt getoond van alleen dit systeem. Waarbij van alle bestemmingsattributen de bijbehorende bewerking en bronattributen worden getoond.

De opdrachtgever had een aantal punten, die hij graag uitgewerkt ziet in het scherm. Zo wil hij de richting van de data flow kunnen bepalen door middel van een pijl. Hij wil verder dat bij ieder attribuut een tabelnaam en databasenaam wordt getoond. Hiermee verwijst de opdrachtgever naar de requirement UR3 en UR4. Op basis van deze opmerkingen is er tijdens het gesprek een schermontwerp getekend. Deze ziet er als volgt uit.



Afbeelding 17, schermontwerp tekening

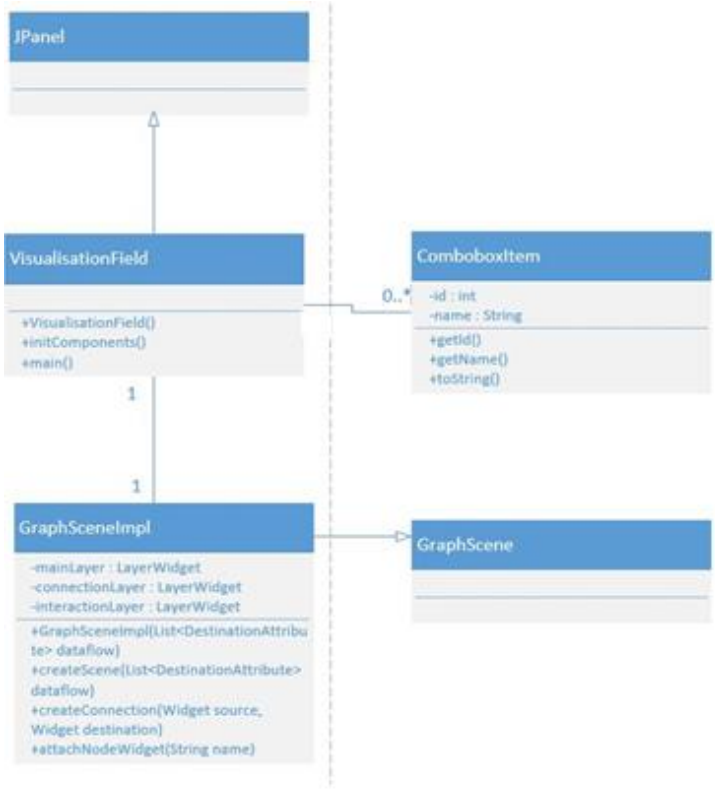
7.2.3 Design

Nadat de vorm en regels van een data flow zijn vastgesteld wordt er een design gemaakt van de beschreven packages. In dit hoofdstuk wordt van iedere package het klassendiagram getoond. Ook zal worden aangegeven op welke wijze de package met de andere packages communiceert. Hierbij wordt van de front end naar de back end gewerkt. Dus er wordt begonnen met de Presentation package en geëindigd met de Data package.

Presentation package
De presentation package heeft te maken met twee onderdelen. Namelijk de logica voor de presentatie en de user interface waarop de presentatie plaatsvindt. Er is gekozen om deze twee onderdelen onder te brengen in verschillende klassen. Dit zodat de logica van de presentatie los wordt gekoppeld van de user interface. Zodat de logica van de presentatie in één klasse is te vinden en er een overzichtelijk situatie ontstaat. De presentatie user interface is ondergebracht in de klassen VisualisationField. Hierop wordt de data flow uiteindelijk getekend. Ook wordt hier de JComboBox getekent in de initComponents(). Deze toont de beschikbare systemen voor visualisatie. Deze systemen moeten wordt opgehaald aan de hand van een uniek id. Daarom heb ik ervoor gekozen om een klasse ComboboxItem te maken. Met deze objecten wordt de combobox gevuld. In dit object wordt de naam van het systeem en de unieke code vastgelegd. Zodat deze bij het selecteren in de combobox kan worden opgehaald en doorgegeven aan het domein.

Verder erft VisualisationField over van het object JPanel. Deze heeft het kenmerk om zichzelf weer te geven en als container te dienen voor de verschillende elementen. (Combobox, visualisatie van de data flow) Deze elementen worden aangemaakt tijdens de start. Er ontstaat zo een scherm dat verantwoordelijk is voor het tekenen van de

elementen. Waarvan de elementen op hun beurt weer verantwoordelijk zijn voor het tekenen van hun componenten.



Afbeelding 18, presentation package

De logica van de presentation package is uitgewerkt in de GraphSceneImpl klasse. Deze klasse erft over van de Netbeans Visual klasse GraphScene. Hiervoor dienen enkele methoden te worden geïmplementeerd, namelijk createConnection() en attachNodewidget(). Bij het starten van de applicatie wordt er vanuit VisualisationField een GraphSceneImpl gemaakt. Om deze te maken wordt een lijst uit het domein opgehaald waarin de data flow wordt beschreven. Op basis van deze lijst wordt widget objecten gemaakt die onderling een connectie hebben door middel van een connectie widget. De attributen en bewerkingen worden op de mainLayer geplaatst en de connectie objecten worden op de connectionLayer geplaatst, dit zijn LayerWidgets. Door een extra layer toe te voegen, (interactionLayer) kan er op de widgets worden geklikt (dit wordt later met code geïllustreerd).

Bij het veranderen van het item in de combobox wordt er een nieuwe GraphSceneImpl gemaakt. Hierdoor wordt de oude data flow van het beeld verwijderd en de nieuwe data flow uit het domein opgehaald.

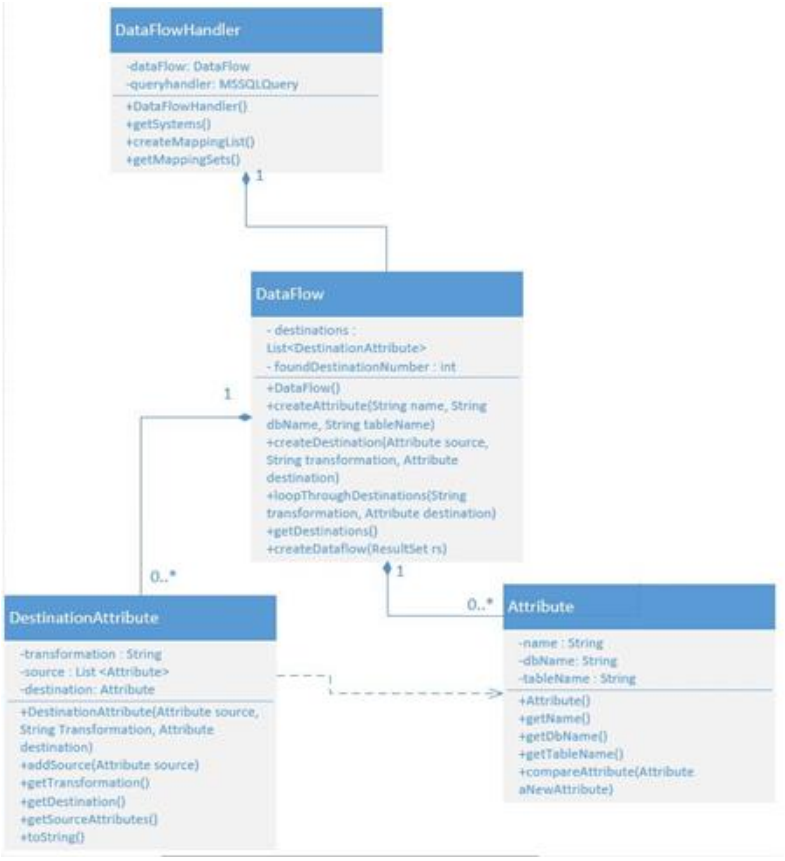
De lijst waarin de data flow staat beschreven wordt opgebouwd aan de hand van de regels van een data flow, zoals hierboven beschreven. De presentatie laag is alleen verantwoordelijk voor het tekenen van de juiste onderdelen van de visualisatie. Op deze manier is de front end alleen verantwoordelijk voor het visualiseren van de data flow en niet de structuur. Door dit te doen kan het domein los van de presentatie laag worden aangepast. Zodat alleen het domein verantwoordelijk blijft voor de structuur van de data flow.

Domein package
De kern van de data flow visualisatie tool is het domein. In het domein wordt de data flow voorbereid voor de visualisatie in de front end. Dit wordt gedaan door de regels van een data flow toe te passen. Het domein bestaat uit vier klassen. De DataFlowHandler is verantwoordelijk voor het doorgeven van de query's. Via de DataFlowHandler wordt er dus gecommuniceerd met de data laag. Verder geeft de DataFlowHandler de data flow door aan de presentation package.

De klasse DataFlow is verantwoordelijk voor het maken van de data flow. Hierbij wordt de relationele data als input gegeven voor de data flow door de DataFlowHandler. De DataFlow zorgt ervoor dat de juiste bronattribu(u)t(en) met de bijbehorende bewerking en bestemmingsattribuut worden gekoppeld.

Ik heb een methode gemaakt die vergelijkt of het bestemmingsattribuut en bewerking overeen komen. Hiermee wordt regel 2 van een data flow gewaarborgd. Om dit verder waar te maken moet een bestemming een lijst van bronattributen bevatten. Daarom heb ik gekozen om een DestinationAttribute te maken. Deze kan dan een bestemmingsattribuut, bewerking en een lijst van bronattributen bevatten. Door dit te doen kan de structuur juist worden bepaald in het domein.

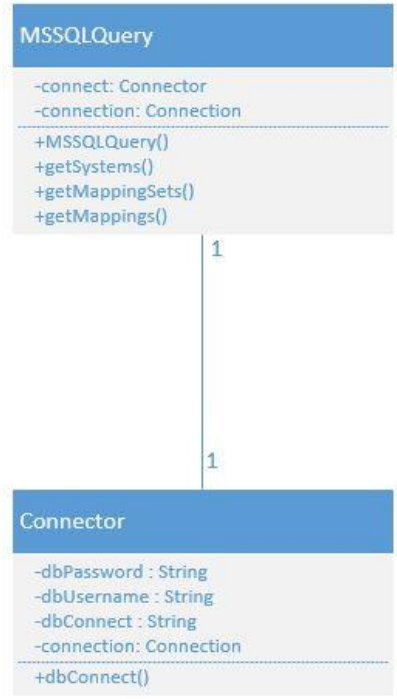
Het onderstaande klassendiagram geeft de uiteindelijke implementatie van het Domein weer.



Afbeelding 19, Domein package

Data Package
De laatste package van de data flow visualisatie tool is de data package. Deze package is verantwoordelijk voor het maken van een connectie met de database en het afvuren en beheren van de query's. Om de connectie met de database te maken is er gebruik gemaakt van JDBC, de release van 2013. Verder ondersteund JDBC veel verschillende DBMS'en waardoor bij het wisselen van DBMS geen nieuwe library hoeft te worden toegevoegd. Verder zorgt de data package ervoor dat de metadata uit de ISMetadata database wordt gehaald.

Er is een klasse gemaakt die ervoor zorgt dat er via JDBC een connectie wordt gemaakt met de ISMetadata database, deze heet connector. Verder is er een klasse die heet MSSQLQuery, deze klasse bevat alle query's die nodig zijn om de data flow te kunnen maken. Deze resultaten geeft MSSQLQuery terug aan de DataFlowHandler, die op zijn beurt aan MSSQLQuery vraagt om de juiste gegevens op te halen. Onderstaand is het bijbehorende klassendiagram van de data package.

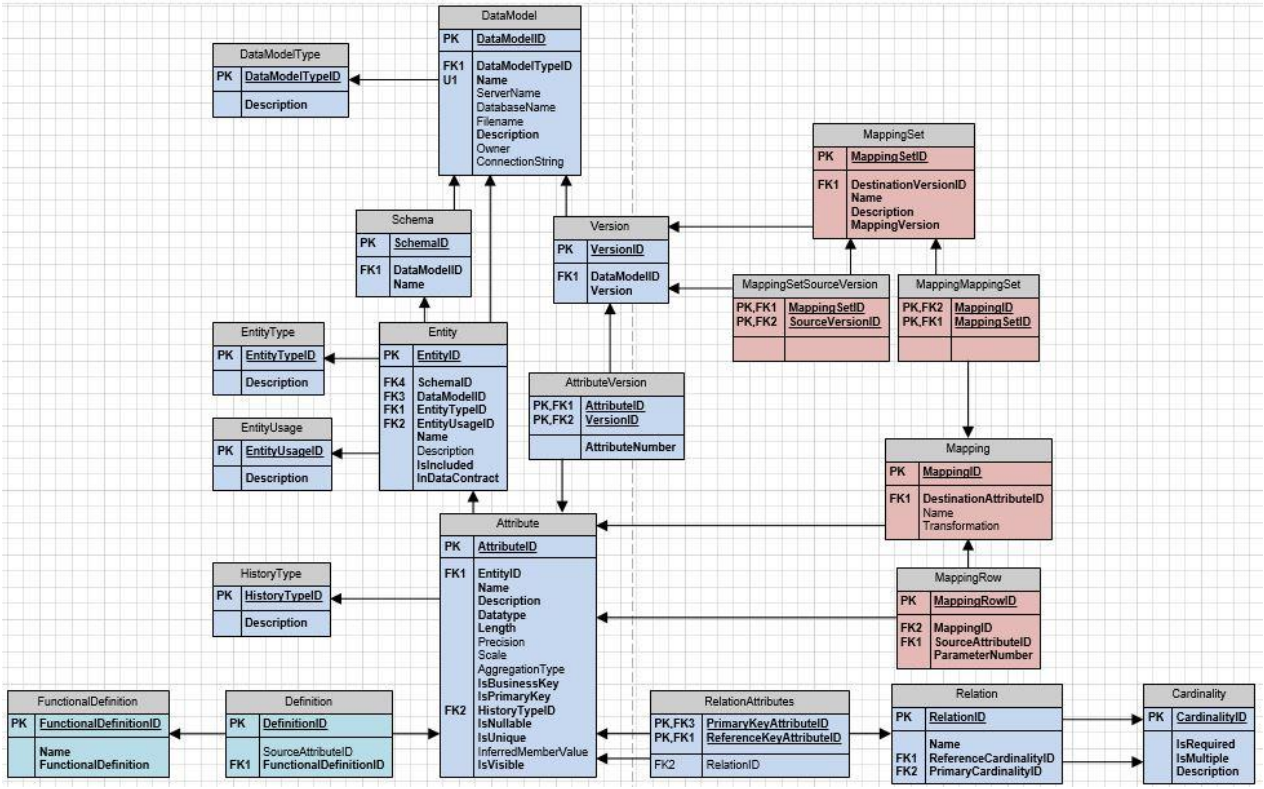


Afbeelding 20, Data package

De data package communiceert met de ISMetadata database. Uit bovenstaand klassendiagram blijkt dat er drie query's zijn opgesteld:

- getSystems() haalt de beschikbare systemen op
- -getMappingSets() haalt de mappingsetten op van een geselecteerd systeem. Een mappingset is een verzameling van mappings van een systeem(er kunnen hier meerdere van zijn)
- getMappings() haalt de mappings op van een mapping set. Hierin staan de bronattributen met bewerking en bestemmingsattribuut in.

Om deze query's op te stellen is er gebruik gemaakt van de documentatie van de ISMetadata tool. Door het onderstaande klassendiagram te analyseren in combinatie met de data dictionary kunnen er query's worden opgesteld. Alle gebruikte query's staan in het technisch ontwerp (bijlage F).



Afbeelding 21, Klassendiagram ISMetadata database

Zodra bekent is welke mappingSet(s) bij het gekozen systeem horen moeten de mappings worden opgehaald. Deze mappings bevatten meerdere onderdelen. In de ISMetadata database worden deze in verschillende tabellen opgeslagen. In de tabel mapping wordt het unieke id van het bestemmingsattribuut en de bijbehorende bewerking (transformation) opgeslagen. Zoals eerder uit de regels van een data flow is gebleken kan een bestemmingsattribuut meerdere bronattributen hebben. Hiervoor is in de database gekozen voor een gescheiden opslag. De bronattributen worden opgeslagen in de tabel mappingRow. Deze zijn gekoppeld aan het bestemmingsattribuut door middel van een mappingID.

Verder heeft de opdrachtgever in de requirementsfase aangegeven, dat er van een bronattribuut de naam, tabel- en databasenaam moet worden getoond. Ook moet van een bestemming een naam, tabel- en databasenaam worden getoond. Om dit te doen moeten de gegevens uit de tabel attribute, entity en datamodel worden gekoppeld. Door een mappingsetID, die bij het bestemmingssysteem hoort, mee te geven wordt door de volgende query de bestemmingsattributen met hun bewerking en bijbehorende bronattributen opgehaald. Hierbij worden de naam, tabel- en databasenaam eraan gekoppeld.

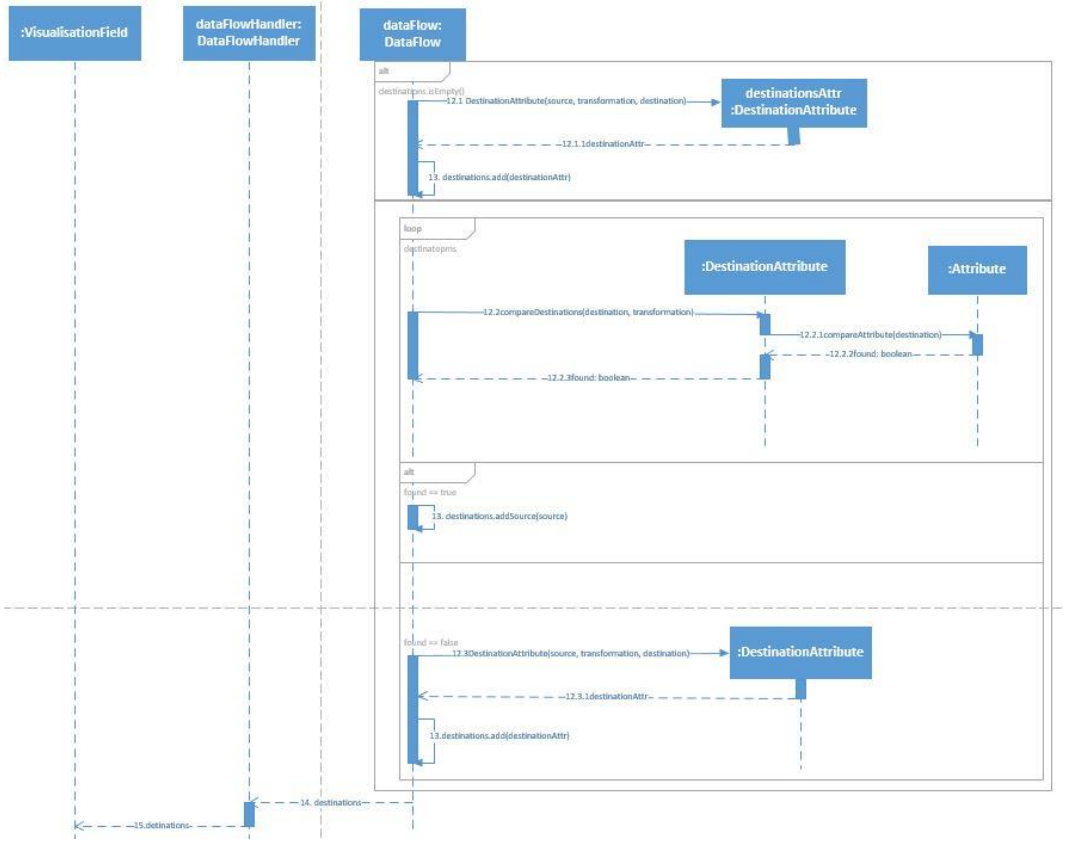
```
SELECT Dest.MappingID, Dest.DestinationAttributeID, DestAttr.Name, DTabel.Name,
DDatabase.Name, Dest.Transformation, Src.MappingID, Src.SourceAttributeID, SrcAttr.Name,
STabel.Name,SDatabase.Name
FROM ISMetadata.ismd.MappingMappingSet MMS
LEFT JOIN ISMetadata.ismd.Mapping Dest
ON MMS.MappingID = Dest.MappingID
LEFT JOIN ISMetadata.ismd.MappingRow Src
ON Src.MappingID = MMS.MappingID
LEFT JOIN ISMetadata.ismd.Attribute DestAttr
ON Dest.DestinationAttributeID = DestAttr.AttributeID
```

```
LEFT JOIN ISMetadata.ismd.Entity DTabel
ON DestAttr.EntityID = DTabel.EntityID
LEFT JOIN ISMetadata.ismd.DataModel DDatabase
ON DTabel.DataModelID = DDatabase.DataModelID
LEFT JOIN ISMetadata.ismd.Attribute SrcAttr
ON Src.SourceAttributeID = SrcAttr.AttributeID
LEFT JOIN ISMetadata.ismd.Entity STabel
ON SrcAttr.EntityID = STabel.EntityID
LEFT JOIN ISMetadata.ismd.DataModel SDatabase
ON STabel.DataModelID = SDatabase.DataModelID
WHERE MappingSetID = GekozenMappingSetID;
```

7.2.4 Data flow algoritme

Als de metadata is opgehaald door de bovenstaande query moet deze nog worden bewerkt. In de metadata staat voor ieder bronattribuut een bewerking en een bestemmingsattribuut. Om de visualisatie te kunnen weergeven moet er voor ieder bestemmingsattribuut in combinatie met de bewerking een lijst van bronattributen worden gemaakt.

Om dit te doen heb ik een controle methode geschreven bij het verwerken van de metadata. Als er een data flow lijst wordt gemaakt door de DataFlow klasse dan wordt deze controle uitgevoerd. Onderstaand sequentie diagram geeft deze controle weer.



Afbeelding 22, Controle op de bronattributen

Als er een DestinationAttribute wordt aangemaakt dan wordt eerst de data flow lijst gecontroleerd. Als deze leeg is hoeft er geen verdere controle plaats te vinden en kan er een nieuwe DestinationAttribute worden gemaakt. Als de lijst niet leeg is wordt de controle uitgevoerd. Om vast te stellen of een bestemmingsattribuut en bewerking overeenkomen heb ik gekozen voor een controle die gescheiden werkt. Ik heb de het object Attribute verantwoordelijk gemaakt voor het controleren van overeenkomende bestemmingsattributen. Door dit te doen blijft en attribuut verantwoordelijk voor zichzelf en wordt de applicatie beter object georienteerd. Ik heb dus gekozen dat een attribuut overeenkomt als de attribuutID, attribuutnaam, tabelnaam en databasenaam overeenkomen. Als de transformatie dan ook overeenkomt dan wordt het bronattribuut toegevoegd aan een bestaande data flow, anders wordt er een nieuwe data flow aangemaakt.

Door in de achterkant van de applicatie deze lijst volgens de regels van een data flow op te bouwen, is de data flow verantwoordelijk voor de structuur. De visualisatie is dan alleen verantwoordelijk om de juiste grafische elementen te tekenen om deze structuur weer te geven.

8. Construction & testing 1

In het vorige hoofdstuk is het ontwerp van de eerste versie van de data flow visualisatie tool besproken. Er is hier volgens een package architectuur beschreven op welke manier de software is opgebouwd. Door middel van schermontwerpen is de gebruikers interface samengesteld. Ook is de bestaande achterliggende database structuur van ISMetadata naar voren gebracht.

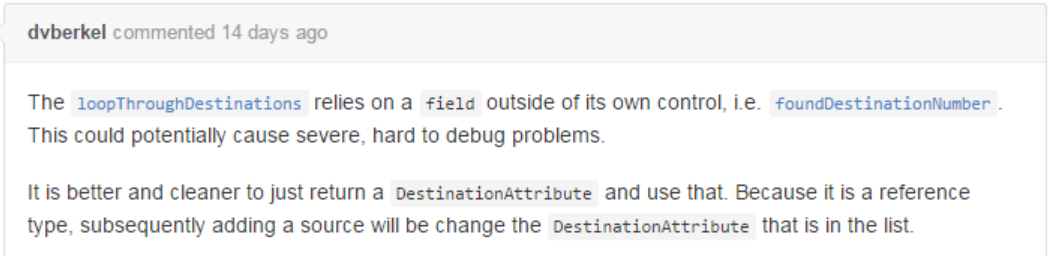
Dit hoofdstuk geeft inzicht in de eerste constructionfase van dit project is doorlopen. Ook worden de gebruikte ontwikkel- en testtechnieken naar voren gebracht. Aan de hand van een code voorbeelden wordt het algoritme uit vorig hoofdstuk verder toegelicht. Als laatste worden er screenshots getoond van de data flow visualisatie tool nadat deze uit de eerste bouwfase is gekomen. Als laatste wordt in dit hoofdstuk het testtraject toegelicht. Hierbij worden de gebruikte testmethode toegelicht en de testsoorten met de resultaten getoond.

8.1 Bouwen

Het ontwerp uit de elaboration 1 fase dient als basis tot de construction 1 fase. Aan de hand van de verschillende diagrammen en de geprioriteerde requirements wordt de data flow visualisatie tool gebouwd. Om het bouwproces te ondersteunen is er gebruik gemaakt van een verscheidenheid aan technieken.

De eerste gebruikte techniek is versiebeheer. In versiebeheer kan de source code van een project worden opgeslagen. Door het gebruik van versiebeheer kan er een duidelijk overzicht worden gegeven van de veranderingen aan de software. Op deze manier kunnen de functionaliteiten, die later worden toegevoegd, onderscheiden worden van de vorige versies. Zodat er bij problemen met de nieuwe functionaliteiten kan worden teruggegaan naar de vorige stabiele versie. Vanwege eerdere positieve ervaringen met GitHub is deze gekozen als repository voor de source code.

Een bijkomend voordeel van het gebruik van GitHub is dat de Java expert, Daan van Berkel, de source code kan bereiken. De tweede techniek, die is toegepast, is code reviewing. Hierbij wordt de kwaliteit van de code beoordeelt door een tweede persoon, in dit geval een Java programmeur van Info Support. Hij doorloopt de code op problemen, onvoorziene uitkomsten en onderhoudbaarheid. Hij heeft via de GitHub website de code kunnen benaderen op afstand. Hierbij maakte hij isssues aan bij de repository om gevonden problemen aan te duiden. Verder zijn er in sessies door de source code heen gelopen waarbij onvoorziene problemen worden opgelost. Onderstaand is een voorbeeld van een aangemaakte issue bij de respository door een code review sessie.[19]



Afbeelding 23, Voorbeeld issue die naar voren kwam bij een code review

De methode die gebruik maakt van deze methode is de createDestination methode. In deze methode wordt het vierde sequentie diagram uitgevoerd. Hierbij wordt de lijst van de dataflow gevuld en bij een overeenkomstige bewerking en bestemming wordt alleen een bronattribuut toegevoegd. Deze methode gebruikt de loopThroughDestinations om te bepalen of de transformatie en de bestemming overeenkomen. Hierbij wordt gewacht op een boolean en wordt een nummer van buiten de methode gebruikt om dit object aan te passen bij een overeenkomst. Uit de code review blijkt dat hiervoor een oplossing is, die geen gebruik maakt van een boolean en een nummer maar een referentie naar het gevonden object.

```
public void createDestination(Attribute source, String transformation, Attribute destination) {

    if (destinations.isEmpty()) {
        DestinationAttribute destinationAttr = new DestinationAttribute(source, transformation, destination);
        destinations.add(destinationAttr);
    } else {
        boolean destinationNumber = this.loopThroughDestinations(transformation, destination);
        if (destinationNumber == true) {
            destinations.get(foundDestinationNumber).AddSource(source);
        } else {
            DestinationAttribute destinationAttr = new DestinationAttribute(source, transformation, destination);
            destinations.add(destinationAttr);
        }
    }
}
```

Afbeelding 24, De createDestination voor de code review

Door een directe referentie te gebruiken naar het object kan er zonder een boolean en nummer buiten de methode worden gewerkt. Het nummer (foundDestinationNumber) is onderdeel van de klassen en wordt telkens overschreven. Deze situatie kan bij toekomstige aanpassingen te komen vervagen, waardoor het gebruik van een nummer buiten de methode voor een onvoorzien probleem kan zorgen. Op deze manier worden er geen onnodige variabelen gebruikt en kan een bronattribuut toch worden toegevoegd aan het juiste bewerking en bestemming. Na het herschrijven van de loopThroughDestintations(hernoemt naar findDestinationAttribute) methode ziet deze er als volgt uit:

```
public DestinationAttribute findDestinationAttribute(String transformation, Attribute destination) {
    for (i = 0; i < destinations.size(); i++) {
        if (destinations.get(i).compareDestinations(destination, transformation)) {
            return destinations.get(i);
        }
    }
  return null;
}
```

Afbeelding 25, De nieuwe methode om bewerking en bestemmingsattribuut te vergelijken

Na het herschrijven wordt er een destinationAttribute teruggegeven als deze wordt gevonden in de bestaande lijst. Als er geen overeenkomstig destinationAttribute is dan wordt er een null teruggegeven. Om deze veranderde methode te kunnen gebruiken moet de createDestination methode ook worden aangepast. Deze ziet er na de aanpassing dan ook als volgt uit:

```
public void createDestination(Attribute source, String transformation, Attribute destination) {

if (destinations.isEmpty()) {
    DestinationAttribute destinationAttr = new DestinationAttribute(source, transformation, destination);
    destinations.add(destinationAttr);
} else {
    DestinationAttribute foundDestination = this.findDestinationAttribute(transformation, destination);
    if (foundDestination != null) {
        foundDestination.AddSource(source);
    } else {
        DestinationAttribute destinationAttr = new DestinationAttribute(source, transformation, destinatio
        destinations.add(destinationAttr);
    }
}
```

Afbeelding 26, De nieuwe createDestination met gebruik van de nieuwe vergelijkmethode

Het bovenstaande is een voorbeeld van het verwerken van een review op de code. Tijdens de gehele construction fase is er wekelijks een code review gedaan. Hierbij zijn de issues toegevoegd op GitHub en daarna verwerkt door mij in de code. Op deze manier wordt de kwaliteit van de code verbeterd en gewaarborgd. Er ontstaat code, die voor Info Support op hun kwaliteitswensen aansluit.

Tijdens het bouwen van de data flow visualisatie tool is gebruik gemaakt van library's. Zoals in vorig hoofdstuk is beschreven is JDBC gebruikt voor de connectie met de ISMetadata database. Verder is uit het onderzoek naar voren gekomen dat er twee libraries zijn die de visualisatie kunnen ondersteunen. Hierbij kwamen JUNG en Netbeans Visual naar voren. Het gebruik van JUNG werd aangeraden als er een layout moest worden bepaald. Echter bleek uit de requirements dat een layout niet van belang is voor de visualisatie. Hierdoor is het gebruik van JUNG overbodig en dus heb ik deze dus niet gebruikt. Het plaatsen van de elementen op het scherm kan hierdoor dan worden gedaan door het aangeven van vaste coördinaten.

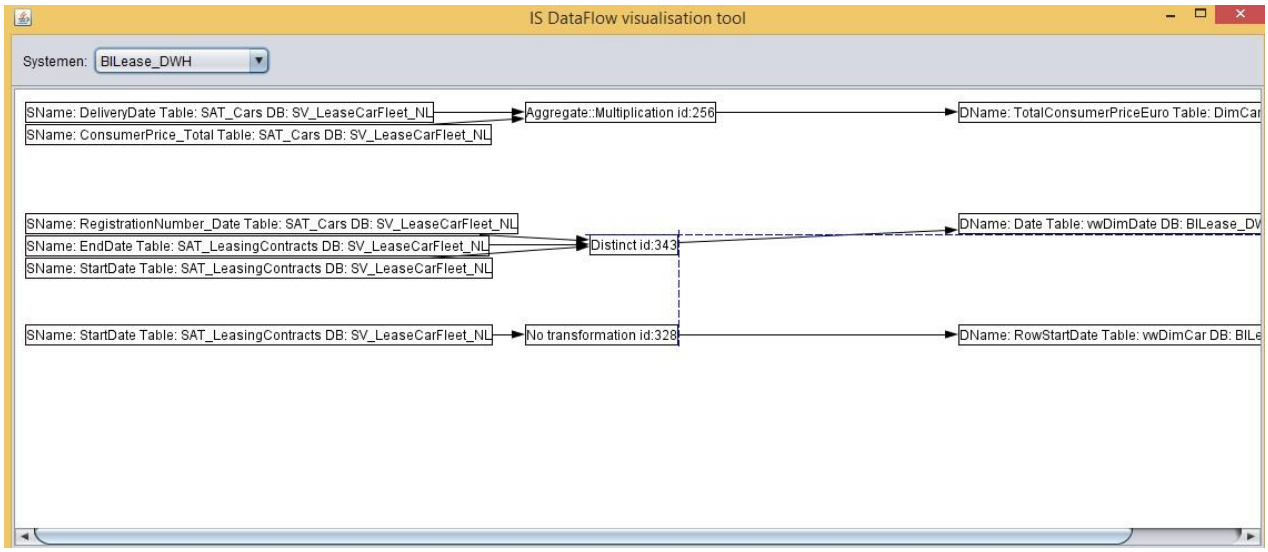
Wel heb ik gebruik gemaakt van de Netbeans Visual library. Uit het schermontwerp bleek dat het gebruik van de widget objecten uit deze library positief werd ontvangen. Er is daarom op basis van het advies uit het onderzoek en de positieve reactie van de opdrachtgever gekozen om voor de visualisatie de Netbeans Visual library te gebruiken.

In de presentation package van het ontwerp wordt de Netbeans Visual library gebruikt. Deze presentation package maakt van de lijst met destinationAttributes widget objecten. Deze objecten worden gemaakt met de methode die een GraphScene object implementeert. Doordat de widget objecten op hun beurt eigen toepassingen hebben kan er via een factory pattern de widget klikbaar worden gemaakt en bij het verslepen van de widget uitlijn lijnen worden getekend. Dit wordt allemaal gedaan door het onderstaande stukje code:

```
@Override
protected Widget attachNodeWidget(String n) {
    LabelWidget widget = new LabelWidget (this);

    widget.getActions().addAction(
            ActionFactory.createAlignWithMoveAction(
                    mainLayer,
                    interactionLayer,
                    ActionFactory.createDefaultAlignWithMoveDecorator()));

    widget.setBorder(BorderFactory.createLineBorder());
    widget.setOpaque(true);

    widget.setLabel(n);
    mainLayer.addChild(widget);

    return widget;
}
```

Afbeelding 27, Methode om een widget te maken die kan worden uitgelijnd

Onderstaande schermafbeelding laat de eerste versie van de data flow visualisatie tool zien. Hierin zijn de drie verschillende onderdelen van een data flow te zien. Aan de linkerkant staan de bronattributen met hun naam, tabel en database. Deze zijn met een pijl verbonden aan hun bewerking, die op zijn beurt weer verbonden is aan het bestemmingsattribuut. Ook is te zien dat bij het verplaatsten de uitlijn lijnen worden getoond van de widget.



Afbeelding 28, Schermafbeelding van de eerste versie IS DataFlow visualisation tool

De data flow wordt gevisualiseerd volgens het algoritme dat hiervoor beschreven is. Verder is de applicatie gebouwd volgens het ontwerp dat is gemaakt in de elaboration fase 1. Hierbij hoort ook de connectie met de database. Onderstaand is een voorbeeld van een query, zoals deze gebruikt wordt in de applicatie.

```
public ResultSet getMappings(int MappingSet) {

    PreparedStatement statement;
    ResultSet rs = null;
    String mappingSetId = Integer.toString(MappingSet);

    try {
        statement = connection.prepareStatement("SELECT Dest.MappingID, "
                + "Dest.DestinationAttributeID, DestAttr.Name, "
                + "DTabel.Name, DDatabase.Name, Dest.Transformation, "
                + "Src.MappingID, Src.SourceAttributeID, SrcAttr.Name, "
                + "STabel.Name,SDatabase.Name, MappingSetID "
                + "FROM ISMetadata.ismd.MappingMappingSet MMS "
                + "LEFT JOIN ISMetadata.ismd.Mapping Dest "
                + "ON MMS.MappingID = Dest.MappingID "
                + "LEFT JOIN ISMetadata.ismd.MappingRow Src "
                + "ON Src.MappingID = MMS.MappingID "
                + "LEFT JOIN ISMetadata.ismd.Attribute DestAttr "
                + "ON Dest.DestinationAttributeID = DestAttr.AttributeID "
                + "LEFT JOIN ISMetadata.ismd.Entity DTabel "
                + "ON DestAttr.EntityID = DTabel.EntityID "
                + "LEFT JOIN ISMetadata.ismd.DataModel DDatabase "
                + "ON DTabel.DataModelID = DDatabase.DataModelID "
                + "LEFT JOIN ISMetadata.ismd.Attribute SrcAttr "
                + "ON Src.SourceAttributeID = SrcAttr.AttributeID "
                + "LEFT JOIN ISMetadata.ismd.Entity STabel "
                + "ON SrcAttr.EntityID = STabel.EntityID "
                + "LEFT JOIN ISMetadata.ismd.DataModel SDatabase "
                + "ON STabel.DataModelID = SDatabase.DataModelID "
                + "WHERE MappingSetID =?");

        statement.setString(1, mappingSetId);

        rs = statement.executeQuery();

    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, e.toString());
        System.out.println("Query mappings faalt");
    }

    return rs;
}
```

Afbeelding 29, Prepared statement query voor het ophalen van de mappings

Ik heb hier gekozen om gebruik te maken van preparedstatements[20]. Hierbij wordt de variabele losgekoppeld van de query. Hierdoor wordt voorkomen dat speciale tekens in de variabelen de query veranderen. Dit is gedaan met het oog op de toekomst. In de toekomst moet het mogelijk worden om data in te voeren om een data flow aan te passen of aan te maken. Door het gebruik van preparedstatements kunnen gebruikers niet per ongeluk met het gebruik van een speciaal teken de query veranderen. Zo wordt foutieve invoer op het database niveau voorkomen.

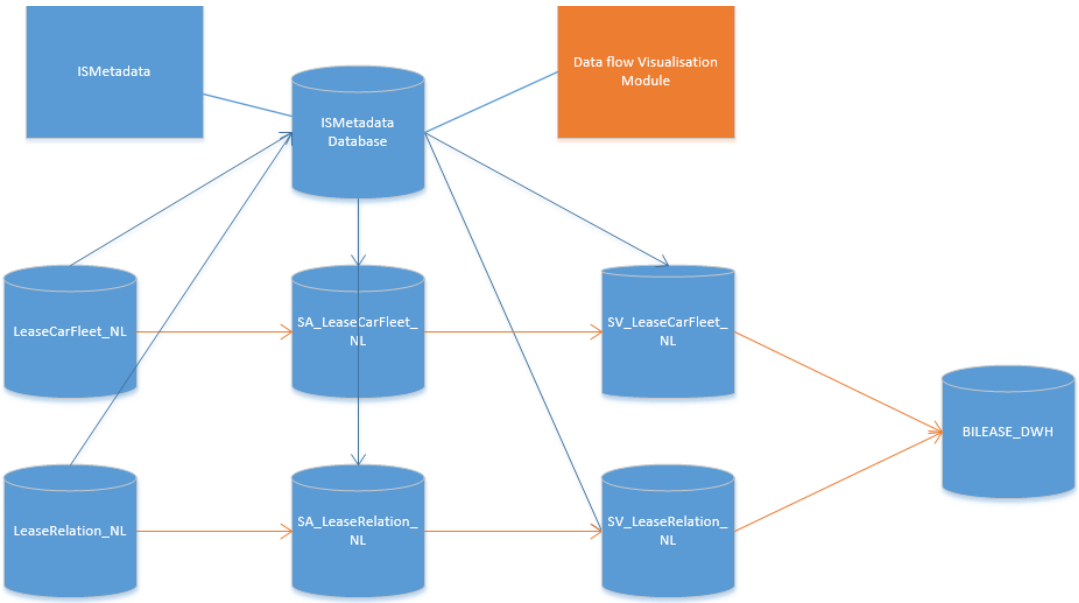
8.2 Testen

In dit hoofdstuk wordt inzicht gegeven in de gebruikte testtechnieken om de data flow visualisatie tool te testen. Allereest zal er worden begonnen met uitleggen van de testopzet. Daarna worden de technieken, die de data flow visualisatie module testen naar voren gebracht.

Om de data flow visualisatie module te kunnen testen heb ik gebruik gemaakt van testdata. Toen ik de opdracht van te voren aannam zou deze module worden ontwikkeld voor een klant van Info Support. Echter heeft de klant aangegeven niet langer de behoefte te hebben aan deze module. Daarom zou de testdata eigenlijk zijn geëxtraheerd uit de klanten productie omgeving. Dit was echter niet meer mogelijk. Om toch testdata te hebben die overeenkomt met de werkelijke data heb ik gebruik gemaakt van testdata van Info Support. Binnen Info Support is deze test data opgezet met het doel om een klantsituatie te simuleren. Hierdoor kan ik er vanuit gaan dat de test data een realistisch beeld geeft van de werkelijke data, waarmee de data flow visualisatie tool te maken gaat krijgen.

De test data van Info Support bestaat uit drie databases. Er zijn twee bron databases genaamd: LeaseCarFleet_NL LeaseRelation_NL. Ook is het bijbehorende datawarehouse van deze bronsystemen aanwezig in de test data van Info Support. Dit datawarehouse heet: BILEASE_DWH.

Om een realistische testomgeving te simuleren wordt de architectuur, zoals te zien is op afbeelding 8, nagebouwd. Hierbij wordt de controle omgeving van een BI systeem gesimuleerd. Ik heb dit gedaan door op een virtuele machine de ISMetadata tool te installeren, dit heb ik gedaan door de bijgeleverde handleiding te volgen. Hiervoor is een Microsoft SQL server 2012 vereist, deze moet de ISMetadata database hosten. Door de bronsystemen in te laden in ISMetadata is er voor beide bronsystemen een Staging area structuur en een Datavault structuur gegenereerd. Ook worden hierbij de mappingschema's tussen de bronsystemen, staging area en datavault gegenereerd. Hierna is ook de metadata van het datawarehouse ingeladen. Hiervoor wordt er later met de hand de mappings van de datavaults naar het datawarehouse toegevoegd(hier meer over verderop in het hoofdstuk). Naast de ISMetadata tool wordt de eerste versie van de data flow visualisatie tool in deze omgeving geplaatst. Zodat deze kan communiceren met de ISMetadata database. Onderstaande afbeelding geeft de testomgeving weer. De oranje pijlen zijn de mappingschema's die zijn opgeslagen in de ISMetadata database. De blauwe pijlen geven aan dat de metadata van deze systemen is opgeslagen in de ISMetadata database, deze structuren worden niet gevisualiseerd.



Afbeelding 30, Testomgeving voor de data flow visualisatie tool

Nadat de architectuur is nagebootst kan er worden achterhaald welke mappings horen bij welk systeem. Op deze manier kan ik verwachte resultaten opstellen. Zo heb ik voor ieder systeem het aantal mappings achterhaald (met een count query). Daarna heb ik geautomatiseerd getest of het verwachte aantal mappings overeen komt met de door de data flow. Dit heb ik gedaan door onderstaande unit tests te schrijven voor de mappingsets.

```
@Test
public void testGetMappings() throws SQLException {
    MSSQLQuery mssqlQuery = new MSSQLQuery(connect, username, password);

    ResultSet rs = mssqlQuery.getMappings(mappingSetId);
    int size = 0;

    try {
        while (rs.next()) {
            size++;
                System.out.println(rs.getString(9) + " " + rs.getString(6) + " " + rs.getString(3));
        }
    } catch (Exception e) {
        Assert.fail("Query faalt");
    }
    assertTrue(size == expectedSize);

}
```

Afbeelding 31, Unit test om het verwachte aantal mappings te testen

Door de inhoud van de resultset te printen kan ik zien dat de opgehaalde data overeenkomt met de mappings in de database. Zodat niet alleen het aantal mappings overeenkomt maar ook de inhoud van deze mappings. Hierna heb ik de data flow visualisatie module de visualisatie laten maken. Hierbij controleer ik of de gegevens uit de database overeenkomen met het aantal en inhoudt van de getoonde data flows. Verder wordt de connectie met de database getest door een test connectie te maken met een unit test.

Verder heb ik tijdens het bouwen van de applicatie gebruik gemaakt van JUnit tests om belangrijke methode op verwachte uitvoer te testen. Zo zorgt de onderstaande unit test ervoor dat de methode wordt getest waarmee er wordt bepaalt of een bronattribuut bij een al bestaand bestemmingsattribuut hoort. De onderstaande methode test

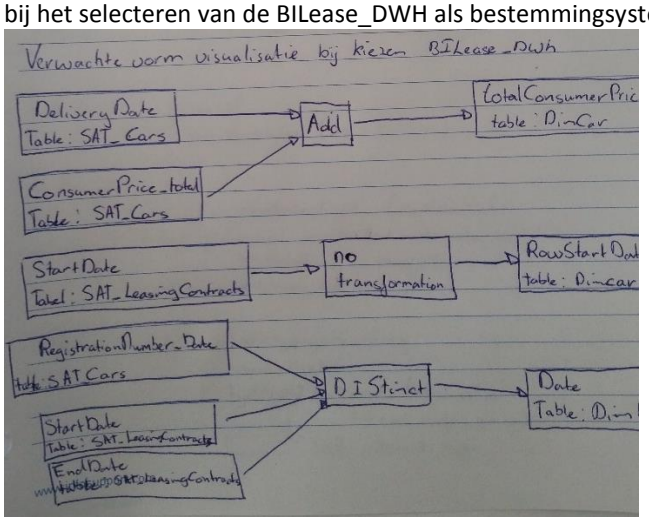
met een situatie waar er true uitkomt, maar een andere unit test veroorzaakt een niet waar status. Om te kunnen vaststellen dat de methode naar verwachtingen werkt. Deze unit tests blijven onderdeel van de applicatie en kunnen bij een nieuwe release opnieuw worden uitgevoerd. Hiermee wordt de functionaliteit van bestaande source code gewaarborgd.

```
@Test
public void compareShouldBeTrue() {
    DataFlow dataFlow = new DataFlow();
    Attribute sourceAttribute = dataFlow.createAttribute("carName", "Database1", "Car", 1, 1);
    Attribute destinationAttribute = dataFlow.createAttribute("VehicleName", "DatabWarehouse", "Vehicle",2,2);
    DestinationAttribute destination = new DestinationAttribute(sourceAttribute, "something",destinationAttribute);


    assertTrue(destination.compareDestinations(destinationAttribute, "something"));

}
```

Afbeelding 32, Module test voor de vergelijk functie van bewerking en bestemming

De interne werking van de data flow visualisatie tool wordt getest door de JUnit tests (zoals hierboven beschreven). Het aanmaken van objecten, vergelijkingsmethoden, loops worden alle ook getest met JUnit tests. Op deze manier heb ik zo veel mogelijk van de code op verwachte uitkomsten getest.

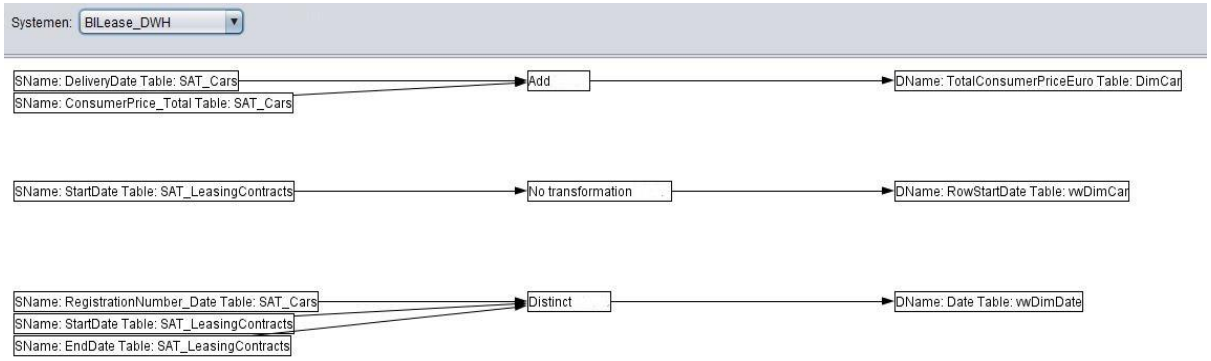
Om de vorm van de visualisatie te testen heb ik met de hand mappings toegevoegd aan de metadata in de ISMetadata database. Dit heb ik gedaan door middel van een transactie script, waarbij er een nieuwe mappingset wordt aangemaakt en daarna mappings worden toegevoegd. Van te voren heb ik bronattributen met een bewerking en een bestemmingattribuut uitgezocht, zodat ik een verwachte uitvoer kan definiëren. Deze attributen en bewerkingen zijn uit het testdatawarehouse (BILease_DWH) gekoppeld aan attributen uit de datavaults. Ik heb bewust gekozen voor bewerkingen met een, twee en drie bronattributen om zo de interne werking nogmaals te verifiëren. Maar ook zodat de data representatief is voor de werkelijkheid. Onderstaand is de verwachte uitkomst bij het selecteren van de BILease_DWH als bestemmingsysteem. (Zie voor de exacte testinvoer het testrapport)



Afbeelding 33, Verwachte data flow uitvoer

Nadat ik de gecontroleerde metadata had ingevoerd en de het verwachte eindresultaat heb gedefinieerd, heb ik de data flow visualisatie tool gestart. Hierbij heb ik het systeem gekozen BILease_DWH, zoals beschreven in de use case beschrijving. Hierna toonde de applicatie de onderstaande visualisatie. Deze exact overeenstemmend met de

verwachte uitkomst. Door deze test een aantal keer te herhalen, met steeds andere handmatige metadata invoer en verwachte uitvoer, heb ik vastgesteld dat de data flow visualisatie tool op basis van de data de juiste visualisatie vorm toont.



Afbeelding 34, Werkelijke uitvoer

Verder heb ik met de opdrachtgever een functionele acceptatie test gedaan en een gebruikers acceptatie test. Hiermee toets ik of de gestelde requirements zijn terug te vinden in de applicatie. Verder controleer ik samen met de opdrachtgever of het gebruik van het systeem gebruiksvriendelijk is. Deze eis kwam voort uit de opdracht omschrijving. De opdrachtgever gaf aan het systeem duidelijk en gebruiksvriendelijk te kunnen gebruiken. Verder heb ik aan de hand van de use cases logische en fysieke testcases opgezet. Deze tests maken dan gebruik van de test metadata die ik heb ingevoerd en heb laten genereren. Hierbij wordt ook gecontroleerde invoer en uitvoer met elkaar vergeleken. Hieruit kwam naar voren dat de data flow visualisatie tool bij alle use cases en de alternative flows het gewenste resultaat toonde.

9. Elaboration 2

Nadat de eerste construction fase is afgerond start de tweede elaboration fase. Tijdens het uitvoeren van de testen kwamen er een aantal nieuwe eisen en wensen naar voren. In deze tweede elaboration fase worden deze nieuwe eisen en wensen meegenomen. In dit hoofdstuk wordt het proces van de tweede elaboration fase naar voren gebracht. Dit gebeurt door eerst de nieuwe eisen en wensen naar voren te brengen. Hierna wordt laten zien op welke wijze de should have requirements zijn verwerkt in het ontwerp voor de data flow visualisatie tool.

9.1 Ontwerpen

Voor de tweede elaboration fase wordt de use case met de should have prioritering uitgewerkt. Uit de functionele acceptatie test kwamen nog twee nieuwe requirements naar voren. Deze requirements waren een extra aanvulling en ook gaf hij aan dat deze graag wilde meenemen in de aankomende versie. Ze zijn daarom toegevoegd aan de should have prioriteit.

Requirement ID	Requirement beschrijving
UR 19	Per tabelnaam moet het attribuut een unieke kleur krijgen
UR 20	Van het attribuut moet de databasenaam naar voren komen in een wolkje als er met de muis over het attribuut wordt gegaan

In de tweede elaboration fase is gepland om de should have requirements uit te voeren. Onderstaande requirements vallen onder deze prioritering.

Should Haves

Requirement ID	Requirement beschrijving
UR 6	De flow van een attribuut moet kunnen worden aangepast
UR 19	Per tabelnaam moet het attribuut een unieke kleur krijgen
UR 20	Van het attribuut moet de databasenaam naar voren komen in een wolkje als er met de muis over het attribuut wordt gegaan

Bij de requirement UR6 hoort een nieuwe use case. Deze use case heet aanpassen van een data flow. Om overeenstemming te krijgen tussen mij en de opdrachtgever is deze use case gemodelleerd in een use case beschrijving. Onderstaand is de overeengekomen interactie tussen actor en het systeem.

Naam	Aanpassen data flow
ID	UC 3
Beschrijving	Het aanpassen van de flow van een attribuut in de visualisatie
Primaire actor(s)	Ontwikkelaar of ontwerper
Secondaire actor(s)	-
Pre-conditie(s)	Er is een systeem met een data flow geselecteerd
Scenario beschrijving	1. Actor selecteert de data flow pijl van het attribuut 2. Systeem geeft de versleeppunten van de pijl weer 3. Actor selecteert het pijluiteinde dat versleept dient te worden en versleept deze naar de nieuwe transformatie 4. Actor zet de pijl op de nieuwe transformatie[1][2] 5. Actor klikt op opslaan

	6. Het systeem slaat de wijziging op
Postconditie(s)	De actor heeft de data flow van een attribuut naar wens gewijzigd
Alternatieve flow	[1] geen transformatie geselecteerd 5. Systeem slaat wijziging niet op [2] Actor selecteert een ander bronattribuut of bestemmingsattribuut 5. Systeem zet de pijl terug op de transformatie

Deze use case heb ik hierna verwerkt in het bestaande ontwerp van de data flow visualisatie module. Doordat ik een architectuur heb gebruikt met duidelijke scheiding tussen de onderdelen kan het aanpassen van een data flow hier in worden opgenomen. De logica achter het opslaan van een data flow wordt meegenomen in de data package (Zie afbeelding 15). Verder wordt het omzetten van de veranderingen in relationele data in het domein geplaatst. In de presentation package plaats ik de visuele mogelijkheden van het aanpassen van een data flow. Op deze manier zijn de functionaliteiten van het aanpassen van een data flow gescheiden over de bijbehorende packages en heeft ieder onderdeel zijn eigen verantwoordelijkheden behouden.

Om veranderingen in de data flow te registreren voordat er wordt opgeslagen heb ik gekozen om in het domein drie klassen en een interface te maken. Door drie nieuwe klassen te definiëren breng ik onderscheid aan op basis van de functionaliteiten van de klassen. Zo is de dataflowhandler verantwoordelijk voor het maken van de data flow en is de changehandler verantwoordelijk voor het bijhouden en opslaan van aanpassing in een data flow.

Bij het aanpassen van een data flow zijn er namelijk twee soorten aanpassingen de pijl van een bronattribuut naar een andere bewerking zetten en de pijl van een bewerking naar een andere bestemmingsattribuut zetten. Om deze bewerkingen te registreren heb ik twee objecten gedefinieerd: ChangeSource en ChangeDestination. In de changehandler wordt de lijst van gemaakte veranderingen voor het opslaan geregistreerd. Hierna zorgt de changehandler ervoor bij het opslaan dat de juiste checks worden uitgevoerd en dat de object data wordt doorgegeven aan de query.

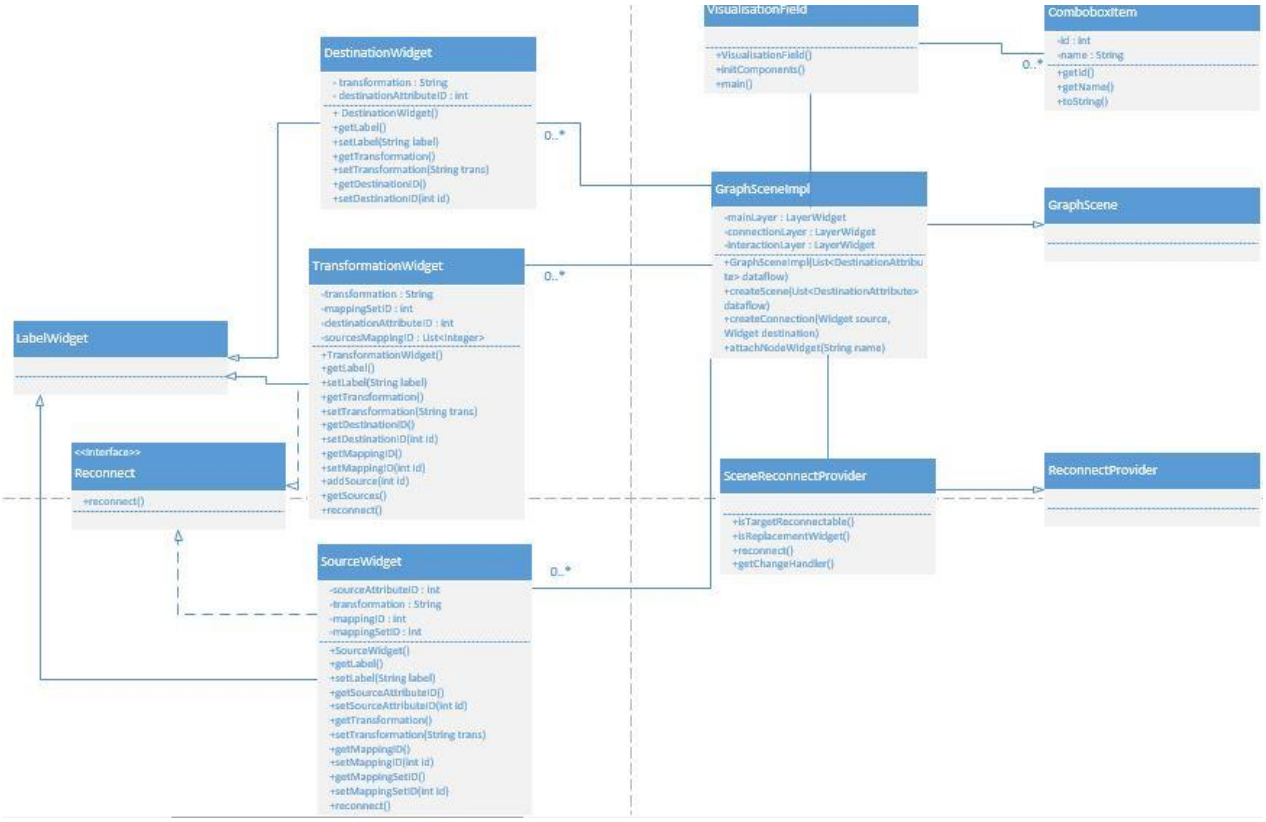
Bij het registreren van veranderingen kan het zo zijn dat een gebruiker een verandering ongedaan maakt. Ik heb ervoor gekozen om bij iedere verandering een object te laten maken die de interface change implementeert. Als de gebruiker ervoor kiest om op te slaan worden de veranderingen met hetzelfde object type vergeleken. Hierbij wordt gekeken of de gebruiker een verandering heeft gemaakt en die later weer heeft terug gedraaid. Als dit het geval is worden de veranderingen uit de veranderingenlijst gehaald, die wordt bijgehouden door de changehandler. Dit zodat de database niet een verandering opslaat en daarna de verandering weer terug draait met een daarop volgende verandering. Door dit te doen worden er niet twee queries uitgevoerd die elkaar weer ongedaan maken. Hierdoor wordt de database zo minimaal mogelijk belast. Ook door de change interface te hebben is het mogelijk om in de toekomst meerdere soorten change objecten te definiëren, zoals het aanpassen van een bewerking, die op hun beurt een eigen inverse() methode hebben om te controleren of de bewerking ongedaan is gemaakt. Onderstaand klassendiagram geeft weer op welke manier de veranderingen worden bijgehouden in het domein.

Afbeelding 35, Toevoeging domein klassen voor het registreren en opslaan van een aanpassing

Om veranderingen te registreren in de changehandler moet in de presentatie logica de aanpassingen met bijbehorende data worden vastgelegd en opgehaald. Ik heb er daarom voor gekozen om bij het aanmaken van de data flow de data op te slaan in de widgets. Dit heb ik gedaan door zelf van de drie onderdelen van een data flow een eigen widget te maken. Deze widget erft over van de labelwidget, zodat deze zichzelf kunnen tekenen en alle andere standaard eigenschappen van een widget kunnen gebruiken.

Door de Netbeans Visual library documentatie door te nemen kwam ik erachter dat er door middel van een factory pattern, die al gebruikt wordt om de widgets te kunnen laten klikken en verplaatsen, ook een ReconnectProvider kan worden toegevoegd aan de widgets. Door deze toe te voegen kunnen de bestaande pijlen opnieuw worden verbonden. Hiermee kan een data flow dus visueel worden aangepast. Doordat deze interface een aantal methoden implementeert kan de data van de oude connectie en de nieuwe connectie worden opgevraagd. Doordat de reconnectprovider de widgets de aanpassing meegeeft kan de data uit deze custom widgets worden opgevraagd. Ook kan ik door een reconnect interface te definiëren de business regels van een data flow hanteren. Zo kan ik er al voor zorgen dat een bronattribuut nooit aan een bestemmingsattribuut of andere bestemmingsattributen kan worden gekoppeld. Door dit te doen kan ik foutieve aanpassingen in de presentatie logica weren. Dit zodat er alleen aanpassingen kunnen worden gedaan, die volgens de syntax regels van een data flow kloppen. Door dit te doen

wordt het risico van foutieve invoer op de database afgedekt. Onderstaande afbeelding geeft weer op welke wijze de presentatie package nu is ingericht.



Afbeelding 36, Nieuwe structuur presentation package met de nieuwe klassen voor het registreren van een aanpassing

In de data package heb ik de query's toegevoegd die een aanpassing aan de data flow registreren. Om ervoor te zorgen dat het opslaan van een aanpassing volledig wordt uitgevoerd heb ik gebruik gemaakt van transacties. Door dit te doen wordt bij een fout in de query teruggedraaid. Op deze manier wordt foute invoer voorkomen, dit zodat de ISMetadata tool geen foutieve data in de database zal hebben staan. Dit is vooral belangrijk omdat er in de toekomst een tool zal worden ontwikkeld, die van de data flows (gemaakt door de data flow visualisatie module) een etl schema zal genereren. Doordat ik de data op verschillende punten in de applicatie kan controleren kunnen de eisen van de etl generator tool worden ingepast in de data flow visualisatie tool.

10. Construction & testing 2

Nadat de nieuwe use case is verwerkt in het ontwerp wordt deze ingebouwd in de data flow visualisatie module. In dit hoofdstuk wordt het bouwproces naar voren gebracht. Hierna wordt verteld op welke wijze de data flow visualisatie module is getest op de bestaande en nieuwe functionaliteiten. Het hoofdstuk eindigt met het beschrijven op welke wijze de data flow visualisatie module wordt overgedragen aan Info Support, de transition fase van RUP.

10.1 Bouwen

De tweede construction fase bouwt voort op de bestaande code van de data flow visualisatie module. Deze code staat in versiebeheer en daarom kon ik direct verder gaan tot het programmeren van de nieuwe functionaliteiten. Ik ben begonnen met het aanmaken van nieuwe widget objecten. Deze moesten de id's van hun aanliggende widgets opslaan. Dit zodat bij het aanmaken van een verandering de benodigde data kan worden doorgegeven aan het verander object.

Hierna heb ik de reconnectProvider toegevoegd. Deze wordt gekoppeld aan de pijlen, dit zijn connectionwidgets. Hierdoor kan er bij het aanpassen van de data flow een aantal controles worden uitgevoerd. Om zelf deze controles te kunnen definiëren heb ik een eigen implementatie gemaakt van de reconnectprovider. Hierdoor kan ik forceren dat de regels van een data flow behouden blijven. Dus zodat een bronattribuut alleen gekoppeld mag worden als deze aan een andere transformationwidget wordt vastgemaakt.

Tijdens de tweede construction fase heb ik wederom gebruik gemaakt van code reviewing. Daarbij heb ik met Daan van Berkel gesproken. Hij heeft de code beoordeeld op kwaliteit zoals dat ook gegaan is tijdens de eerste construction fase.

Ik had voor het verwijderen van de ongedane wijzigingen in de veranderingen lijst de object op null gezet en hierna gekozen om alle null objecten te verwijderen uit de veranderingen lijst. Zie de onderstaande code voordat de code was gereviewd.

```
public void removeInverses() {

    for (int i = 0; i < changesMade.size(); i++) {
        for (int j = 0; j < changesMade.size(); j++) {
            if ( changesMade.get(i) != changesMade.get(j) && changesMade.get(i).inverse(changesMade.get(j))) {
                if(changesMade.get(i) != null && changesMade.get(j) != null)
                changesMade.set(i, null);
                changesMade.set(j,null);
            }
        }
    }
    changesList.removeAll(Collections.singleton(null));
}
```

Afbeelding 37, code van verwijderde omgekeerde verandering voor code review

Tijdens de code review kwam hiervoor en nettere oplossing naar voren. Het probleem met de bovenstaande code is dat als er twee objecten op null worden gezet er op een gegeven moment de null objecten met elkaar vergeleken zullen worden. Dit kan zorgen voor onverwachte problemen in de methode. Een nettere oplossing zou zijn het bijhouden van een lijst met objecten die elkaars omgekeerde zijn en deze aan het einde van de methode uit de

originele veranderingen lijst te verwijderen. Nadat ik de code heb herschrijven is de onderstaande afbeelding het resultaat. Door deze verbetering hoeft er geen controle worden gedaan op het mogelijk voorkomen van null objecten. Er wordt een lijst bijgehouden van alle objecten die elkaars omgekeerde zijn. Deze lijst van omgekeerde veranderingen worden aan het einde van de loop uit de lijst met veranderingen verwijderd.

```
public void removeInverses() {

    List<Change> inverses = new ArrayList<Change>();

    for (Change change : changesMade) {
        for (Change candidate : changesMade) {
            if (change != candidate && change.inverse(candidate)) {
                if (!inverses.contains(change)) {
                    inverses.add(change);
                }
            }
        }
    }
    changesMade.removeAll(inverses);
}
```

Afbeelding 38, code van verwijderde omgekeerde verandering na code review

10.2 Testen

Tijdens en nadat ik de nieuwe functionaliteit heb toegevoegd aan de data flow visualisatie module heb ik getest. In dit hoofdstuk wordt inzicht gegeven in de technieken die ik heb gebruikt om de module te testen.

Terwijl ik de data flow visualisatie tool aan het programmeren was heb ik gebruik gemaakt van module tests. Allereerst heb ik voor de nieuwe methoden en klassen module tests geschreven. Maar bij het builden van de applicatie voer ik de eerder gemaakte, uit de eerste iteratie, module tests opnieuw uit. Door deze regressie test te doen kan ik vaststellen dat de bestaande en nieuwe methoden en klassen functioneren naar verwachting.

Zo heb ik onderstaande module test geschreven om te testen of het verwijderen van ongedane wijzigingen naar verwachting werkt. In de test maak ik drie veranderingen aan waarvan er twee elkaars omgekeerde zijn. Na het uitvoeren van de removeInverses() methode moeten deze twee omgekeerde zijn verwijderd uit de veranderingen lijst. Het resultaat dat ik verwacht is dan ook een array met veranderingen met de lengte van één. Van deze module test heb ik een aantal varianten gemaakt. Hierbij test ik of bij het invoeren van alleen echte (niet ongedane) veranderingen de verwachte lengte van de array kloppend blijft. Op deze manier zijn de mogelijke uitkomsten van de removeInverses() methode getest en kan ik aan de hand van de verwachte resultaten zien dat de methode werkt.

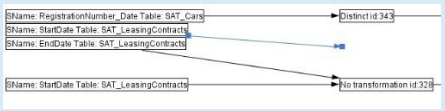
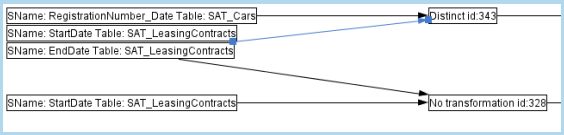
```
@Test
public void checkDoubleDestinationEntriesShouldBeTrue() {

    int sourceMap1 = 20;
    int destID1 = 200;
    String newTrans = "no transformation";
    String oldTrans = "verm";
    int oldDest = 100;

    ChangeDestination firstDestinationEntry = new ChangeDestination(sourceMap1, destID1, newTrans, oldTrans, oldDest);
    changehandler.addChange(firstDestinationEntry);
    ChangeDestination changeBack = new ChangeDestination(sourceMap1, oldDest, oldTrans, newTrans, destID1);
    changehandler.addChange(changeBack);
    changehandler.addChange(realChangeDestination);

    changehandler.removeInverses();

    assertTrue(changehandler.getChangesMade().size() == 1);

}
```

Afbeelding 39, module test van de removeInverses() methode

Naast de module tests is het belangrijk om de invoer van de metadata in de ISMetadata database te testen. Met deze metadata moet het in de toekomst mogelijk zijn om ETL schema's te genereren. Hierdoor is het belangrijk dat de aanpassingen in de bestaande data de juiste resultaten leveren.

Om dit te doen heb ik een aantal tests uitgevoerd. Allereerst heb ik testscenario's opgesteld. Deze testscenario's heb ik uitgevoerd in de testomgeving, zoals deze is omschreven in de vorige iteratie. In deze testomgeving is de architectuur uit de controle omgeving van de BI architectuur nagebootst. Door dit te doen kan er een representatieve situatie en data worden gecreëerd. De resultaten uit de testomgeving geven dan een goed beeld van de werking van de data flow visualisatie module in de werkelijkheid.

Allereerst heb ik de invoer van de veranderingen getest. Deze testscenario's testen de regels van een data flow, zoals deze staan beschreven in hoofdstuk 7.2.1. Alle testscenario's staan beschreven in het testrapport. Onderstaand is het voorbeeld waarbij ik test of het mogelijk is om een bronattribuut direct te koppelen aan een bestemmingsattribuut, volgens de regels van een data flow is dit niet mogelijk. Tijdens het programmeren heb ik door de reconnect() methoden deze regels toegepast. Hierbij wordt in de presentatie laag het aanbrengen van deze niet toegestane bewerking geweigerd. Het testscenario test deze regels of deze daadwerkelijk worden toegepast.

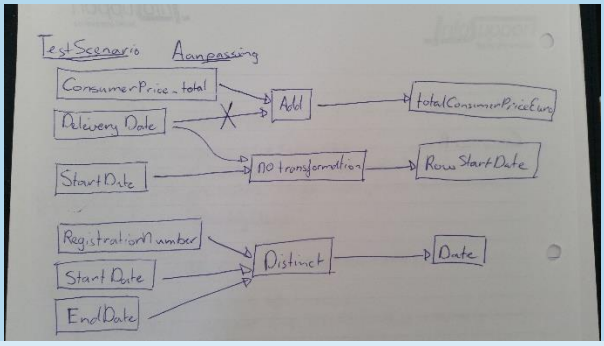
Fysieke testcase 7:	Aanpassen data flow		
Naam tester:	**Maarten Koene**		
Pre-conditie:	Er is een systeem met een data flow gekozen (BILease_DWH)		
Requirements:	UR6		
Main Flow	Verwacht resultaat	Werkelijk resultaat	Opmerkingen
1.Actor selecteert de data flow pijl van StartDate		✓	
3. Actor selecteert het pijluiteinde van de transformatie "Distinct"		✓	
4.Actor zet de pijl op de bestemmingsattribuut "Date"		✓	De pijl zit vast aan het oorspronkelijke transformatie
5. Actor klikt op opslaan	Het systeem geeft de melding dat er geen aanpassingen zijn om op te slaan	✓	

Met dit soort testscenario's heb ik aan de voorkant van de applicatie getest of er mogelijke aanpassingen worden gedaan die een tot verkeerde of niet mogelijke data zorgen. Verder heb ik terwijl ik deze testscenario's uitvoerde in de database gekeken of niet toegestane aanpassingen ook niet worden opgeslagen. De applicatie geeft hier ook een melding van als er niets kan/hoeft te worden opgeslagen. Maar om er zeker van te zijn dat de data niet alsnog wordt opgeslagen in de database heb ik in de ISMetadata database in de testomgeving gekeken. Bij scenario's

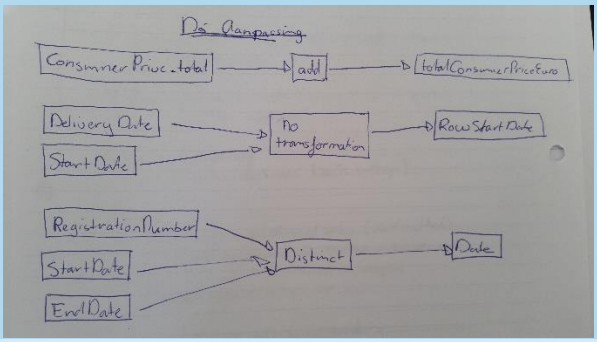
waarbij de veranderingen wel mag/kan worden opgeslagen heb ik in de database gekeken of de data corresponderend met de aanpassing in de visualisatie tool is gemaakt en opgeslagen.

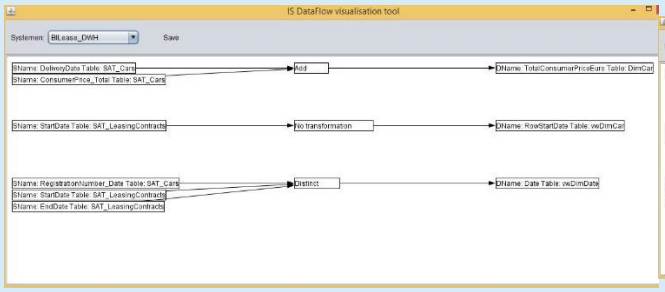
Verder heb ik testscenario's gemaakt waarin ik bij een opgeslagen aanpassing in de data flow een verwachte nieuwe data flow visualisatie heb gedefinieerd. Op deze manier test ik of de aanpassing in de data flow wordt opgeslagen en bij het bekijken van de aangepaste data flow de aanpassing ook wordt getoond. Hierbij schets ik de data flow visualisatie van te voren. Ik geef aan welke data flow wordt aangepast en wat het verwachte eindresultaat is. Onderstaand is een voorbeeld van zo'n test. Hierbij wordt de data flow van delivery data aangepast van transformatie en bestemmingsattribuut. Nadat de data flow is opgeslagen ziet de data flow eruit zoals de tweede tekening aangeeft. De data flow visualisatie module laat in de scenario zien dat de aanpassing van de data flow wordt opgeslagen en daarna op de juiste manier wordt getoond.

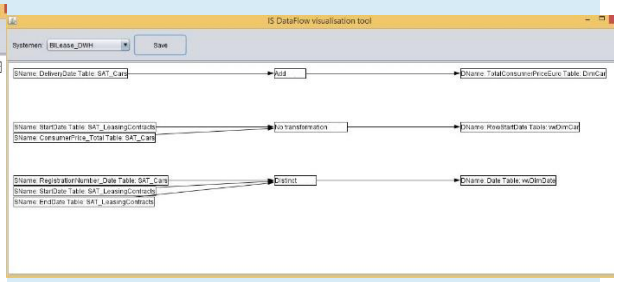
Visualisatie voor aanpassing



Situatie na aanpassing







De laatste tests die ik heb uitgevoerd zijn de regressietests van de vorige iteratie. Daarnaast is er met de opdrachtgever een GAT en een FAT uitgevoerd. Hiermee heb ik de requirements en de gebruiksvriendelijkheid van de applicatie vastgesteld. Er kwam hieruit naar voren dat de opdrachtgever niet op een opslaan knop wil drukken maar dat hij bij het aanpassen van de data flow direct de verandering wil opgeslagen hebben. De gevraagde veranderingen zijn meegenomen in het vrijgaveadvies. Alle testresultaten, behalve de module tests, zijn te vinden in het testrapport. Ook is daar het vrijgaveadvies te lezen. (bijlage G)

10.3 Transition

Na het afronden van de construction fase uit de tweede iteratie nadert het project zijn einde. Het laatste fase van de RUP methodiek is de transition fase. Dit hoofdstuk beschrijft op welke wijze de transition fase is uitgewerkt.

Om het project af te ronden wordt de transition fase uitgevoerd. Om een duidelijke overdracht te maken van het project aan Info Support heb ik een aantal stappen genomen. Allereerst heb ik een gebruikershandleiding geschreven. De gebruikershandleiding geeft een stappenplan weer. Deze stappen helpen de gebruikers van de applicatie om de data flow visualisatie module te integreren in de controle omgeving van hun BI systeem. Verder legt de handleiding uit wat de data flow visualisatie module toont en wat de gebruiker hiermee kan doen.

Verder heb ik een overdrachtsdocument gemaakt. In dit document staan de resultaten van het testen. Verder staan hier de eisen en wensen, die nog niet zijn verwerkt in de data flow visualisatie module. Op deze manier is er een duidelijk overzicht voor een eventueel vervolg project. Verder worden de eisen en wensen uit de laatste gebruikers acceptatie test toegevoegd aan de lijst van openstaande eisen en wensen.

Bij de overdracht van de data flow visualisatie module wordt de gemaakte documentatie gebundeld. Hierbij wordt het plan van aanpak, requirements document, functioneel-, technisch ontwerp, testrapportage, gebruikershandleiding en het overdrachtsdocument gebundeld en overgedragen aan de opdrachtgever. Verder heb ik de source code van Github gehaald en overgedragen aan de opdrachtgever. Dit omdat de klant, voor wie dit project was opgezet, heeft afgezien van de data flow visualisatie module. Hierdoor is het niet nodig om de data flow visualisatie module te implementeren bij deze klant.

Het resultaat van de transition fase is dat de documenten en de source code aan Info Support zijn overgedragen. Met de handleiding zorg ik ervoor dat de eindgebruikers een duidelijk beeld hebben van de installatie en de werkwijze van de data flow visualisatie module. Verder zorgt de documentatie ervoor dat Info Support eventueel een vervolg project kan opzetten om de overgebleven functionaliteiten toe te voegen. Ook geeft deze documentatie een duidelijk beeld van de huidige werking.

11. Evaluatie

In dit hoofdstuk evalueer ik het afstudeerproject. Dit doe ik door in hoofdstuk 11.1 eerst het proces te evalueren en de leermomenten daarin aan te stippen. Hierna evalueer ik de geleverde producten en de leerprocessen die ik daarbij heb gehad. Daarna evalueer ik de producten in hoofdstuk 11.2

11.1 Proces evaluatie

Het opzetten van dit project is goed verlopen. Door een aantal gesprekken te voeren met de opdrachtgever en de technisch begeleider kon ik een duidelijk plan van aanpak opstellen. Het onderzoek dat hierop volgde heeft voor vertraging gezorgd. Dit omdat ik laat in het project de ISMetadata tool tot mijn beschikking had. De ISMetadata tool had ik nodig om de structuur van de metadata te beoordelen. Door de verlate inzage in de metadata sloten de resultaten van het onderzoek niet goed aan bij deze tool en moesten deze worden herschreven. Voor een volgend project zou ik door een proactievere houding ervoor kunnen zorgen zo snel mogelijk toegang te krijgen tot applicaties en data. Door bij het begin van het project de afhankelijke data en applicaties te identificeren en direct aan te vragen bij de juiste personen.

De opdrachtgever had tijdens de requirements analyse een drukke periode met zijn klant. Er was maximaal een uur contact per week. Dit risico had ik van te voren niet aangemerkt bij het opstellen van het plan van aanpak. Toen dit risico zich voordeed, heb ik door middel van een inzage in het rooster een strak requirements proces opgezet. Door goed voor te bereiden op de brainstormsessie en prioriteringssessie heb ik een optimaal resultaat weten te boeken. Doordat het risico zich voordeed is er wel tijd verloren gegaan maar door mijn ingrijpen is dit niet verder geëscaleerd. Voor vervolg projecten zal ik het risico van een drukke opdrachtgever meenemen in de risico analyse. Ook kunnen duidelijke afspraken omtrent contacturen dit risico voorkomen.

Om de verloren tijd in het begin van het project te corrigeren, ben ik met mijn opdrachtgever tot een bepaalde requirements prioritering gekomen. Omdat ik aangaf het onmogelijk te achten binnen de resterende tijd alle functionaliteiten toe te voegen, hebben de opdrachtgever en ik gekozen om het tonen van de data flow en het aanpassen van de data flow uit te werken.

Het ontwerpen en bouwen van de applicatie is op een gecontroleerde manier door mij uitgevoerd. Het ontwerpen heb ik gedaan aan de hand van artefacten binnen RUP en deze boden een duidelijke houvast. Voor een vervolg project zou ik deze zelfde aanpak gebruiken. Dit omdat de ontwerpen een duidelijk basis hebben gevormd voor de construction fase. Ook het bouwen van de applicatie is soepel verlopen. Ook heeft code reviewing ervoor gezorgd dat de code van goede kwaliteit is. Voor een volgend project zou ik zeker weer gebruik maken van code reviewing om de kwaliteit van de code te waarborgen.

Bij het testen heb ik een testomgeving ingericht. Deze was gebaseerd op een gecontroleerde testcase van Info Support. Hierdoor ontstond er een representatieve testomgeving. Voor een ander project zou ik de test omgeving wederom opzetten op de manier waarbij er een representatieve omgeving ontstaat. Hierdoor geven de testresultaten een goed beeld van de werkelijkheid. Echter zou in een vervolg project een beter alternatief zijn om echte klantdata te gebruiken. In klantdata kan er de mogelijkheid zijn dat de data fouten bevat, door bijvoorbeeld menselijke fouten. Op deze manier kan je dan beter testen of de applicatie blijft werken, zelfs onder onverwachte omstandigheden. Hierdoor wordt de applicatie robuuster.

De algemene leerpunten van het bovenstaande proces van het afstuderen zijn dan ook als volgt:

- Bij aanvang van het project een proactievere houding aannemen om toegang te krijgen tot de benodigde tools en data
- Tijdens het project is het belangrijk om de uitloop van fasen of stappen te controleren. Dit kan worden gedaan door steeds het tijdspad in de gaten te houden. Op deze manier kan vertraging snel worden geïdentificeerd
- Voordat het project van start gaat is het belangrijk om de werksituatie van de opdrachtgever duidelijk te hebben. Zodat het risico van weinig contact kan worden geïdentificeerd. Verder kunnen duidelijke contact afspraken aan het begin van het project dit risico verminderen

11.2 Product evaluatie

Plan van aanpak
Het plan van aanpak is in de eerste weken van het project opgesteld. Ik ben zelf tevreden over het plan van aanpak. De fasering en planning gaven mij houvast in de voortgang van het project. De huidige situatie beschrijving heeft echter nog een herziening nodig gehad. Dit kwam doordat de toegang tot de ISMetadata tool voor nieuwe en duidelijkere inzichten zorgde. Een risico, dat ik over het hoofd heb gezien, was de beschikbaarheid van de opdrachtgever. Dit risico is tijdens het project wel tot uiting gekomen. Voor een vervolg project zal ik dit risico zeker meenemen. Ook wil ik in het plan van aanpak de contact frequentie vastleggen met de opdrachtgever. Op deze manier is zowel voor mij als de opdrachtgever een duidelijke overeenkomst over de contact uren. Verder heeft het plan van aanpak geholpen om de organisatie omtrent het project vanaf de start duidelijk te hebben. Het plan van aanpak was dus een goede overeenkomst tussen mij en de opdrachtgever.

Onderzoekrapport
Het onderzoek is een document waar ik minder tevreden over ben. Met het onderzoek bracht ik de visualisatie vorm naar voren. Ook heb ik bestaande software en code library's bekeken. Doordat ik later in het project de afhankelijkheid met ISMetadata ontdekte had ik pas laat toegang had tot de tool en waren de resultaten niet passend bij de metadata in deze tool. Hierom heb ik heb onderzoek moeten aanpassen om wel een passend advies te kunnen geven.

Naar mijn mening had de visualisatie vorm bepaald kunnen worden met een requirements analyse. Door met de stakeholders brainstorm sessies te houden kan de visualisatie vorm worden bepaald. Dit zeker omdat door de toegang tot ISMetadata de mogelijkheden van de metadata duidelijk worden. Ook heeft het uitzoeken van bestaande software geen resultaat geboden. Tijdens de eerste resultaten had ik gezocht naar software die data flow in een diagram konden plaatsen, dit gaf geen resultaat. Nadat ik de ISMetadata metadata had geanalyseerd was het duidelijk dat het zoeken naar de wijze waarop ETL tools de data flow weergeven een beter idee had geweest. De metadata beschreef op conceptueel niveau de data flow. Later moet het mogelijk zijn om deze conceptuele data flows om te zetten met een tool zodat er een SSIS package wordt gegenereerd. Wel komt er uit het onderzoek duidelijk naar voren dat er geen bestaande oplossing is voor het visualiseren van de ISMetdata metdata. Het uitzoeken van een geschikte code library had zonder onderzoek kunnen worden gedaan. Zodra de requirements duidelijk zijn kan er een overwogen keuze worden gemaakt over bruikbare code library's.

Mijn conclusie is dat een uitgebreidere requirements analyse had kunnen zorgen voor een bruikbare visualisatie vorm, hierbij had geleerd kunnen worden van bestaande ETL tools en de wijze waarop deze de data flow visualiseren. Verder had na de requirements op te stellen er een duidelijk beeld geweest van de software waardoor er een overwogen code library keuze had kunnen worden gemaakt.

Requirementsrapport
Het requirementsrapport is een document waar ik tevreden over ben. De requirements zijn verkregen tijdens het onderzoek, de brainstormsessie en de opdracht omschrijving. Hierna zijn de requirements geprioriteerd aan de hand van de MoSCoW methode. Hierdoor ontstaan er duidelijke afspraken over de te maken software. De sessies waarbij de eisen naar voren komen waren goed voorbereid en gaven daarom een nuttig resultaat. Voor een volgend project zou ik enkel meer stakeholders willen spreken over de requirements, zodat er een beeld ontstaat van de applicatie geschetst door meerdere stakeholders.

Functioneel ontwerp
Over het functioneel ontwerp ben ik tevreden. De schermontwerpen zijn gebaseerd op de tekeningen van de brainstormsessie gecombineerd met de gekozen visualisatie vorm. Met deze scherm ontwerpen heb ik voor de opdrachtgever een gebruiksvriendelijke data flow visualisatie kunnen maken. Verder hebben de use case beschrijvingen mij ondersteund in het uitwerken van de requirements. Bovendien hebben de beschrijvingen als goede basis voor de logische en fysieke testcases kunnen bieden.

Technisch ontwerp
Het technisch ontwerp ben ik ook tevreden over. De gemaakte klassendiagrammen hebben mij geholpen tijdens de constructionfasen. Hierbij had ik een duidelijke houvast aan de architectuur, die ik had ontworpen. Ook ben ik blij dat ik de documentatie van de ISMetadata database tot mijn beschikking had. Het ontwerpen van de verschillende query's was hierdoor versimpeld. Dit kwam omdat de structuur was beschreven in het klassendiagram voor de ISMetadata database. Ik vond het nuttig om deze voor de volledigheid ook op te nemen in het technisch ontwerp. Ik heb hier later veel aan gehad omdat ik hier snel kon bekijken op welke manier de database in elkaar zat. Verder hebben de sequentiediagrammen ervoor gezorgd dat de belangrijke volgordelijkheid van methode duidelijk wordt. Dit was vooral handig tijdens het debuggen in de construction fase. Ook kon ik met de klassendiagrammen en sequentie diagrammen ingewikkelde concepten uitleggen aan de code reviewer.

Data flow visualisatie module
Ik ben gedeeltelijk tevreden over de data flow visualisatie module. Ik ben tevreden over de kwaliteit van de code achter de module. Dit omdat door herhaaldelijk te code reviewen zijn de standaarden toegepast en konden onverwachte uitkomsten worden opgelost, maar ook door een goed ontwerp dat als basis heeft kunnen dienen. Op deze manier heb ik een object georiënteerde applicatie kunnen programmeren. Hierdoor is de code duidelijk leesbaar en onderhoudbaar geworden. Ik vind het jammer dat ik niet meer functionaliteiten heb kunnen toevoegen. Door tijdsgebrek en uitloop, de afhankelijkheid van ISMetadata en tijdsgebrek bij de requirements, in het begin van het project heb ik overeengestemd met de opdrachtgever om minder functionaliteiten te bouwen. Ik had graag een helemaal afgewerkte module achtergelaten. De module is nu nog niet klaar voor gebruik en dit vindt ik jammer. Echter is er in de toekomst misschien nog mogelijkheid tot het uitbouwen van de applicatie, zodat alle gewenste functionaliteiten erin verwerkt zijn.

Testrapport
Over het testrapport ben ik redelijk tevreden. Door het gebruik van een realistische testomgeving geven de resultaten een betrouwbaar beeld van de applicatie. Ook hebben de unit tests eraan bijgedragen dat de kwaliteit van de code gewaarborgd is. Verder heb ik met gebruikersscenario's de requirements kunnen testen. Ook verwachte in- en uitvoer tests zorgen ervoor dat de data flow visualisatie kloppend is met de achterliggende data. Het verbeterpunt, dat ik zie voor het testen, is het uitvoeren van een FAT en GAT met meerdere stakeholders. Hiermee kunnen de kwaliteitsattributen worden vastgesteld door meerdere stakeholders. Op deze manier kunnen de uitspraken over deze kwaliteitsattributen geldig worden beschouwd voor alle stakeholders. Dit omdat bijvoorbeeld gebruiksvriendelijkheid door iedereen anders kan worden ervaren. Met de hulp van meerdere stakeholders kan je dan beter vaststellen of de gebruiksvriendelijkheid van de applicatie echt goed werkt.

12. Beroepstaken

In dit hoofdstuk worden de beroepstaken, die ik vooraf aan mijn afstudeeropdracht heb opgesteld om te behalen, beschreven. Hierbij geef ik per beroepstaak aan welke werkzaamheden ervoor hebben gezorgd dat ik deze heb uitgevoerd. Per beroepstaak geef ik aan welk hoofdstuk deze werkzaamheden beschrijven en welke bijlagen de resultaten van deze werkzaamheden bevatten.

12.1 Beroepstaak: 1.3 Selecteren van standaardsoftware

Tijdens dit project heb ik een onderzoek gedaan. Hierbij heb ik bestaande software, code library's en visualisatie vormen onderzocht. Uit dit onderzoek komt naar voren dat er geen bestaande software pakketten beschikbaar zijn om een conceptuele data flow te visualiseren. Daarom ben ik code library's gaan onderzoeken. Deze code library's moeten aan een aantal eisen voldoen. Allereerst moeten de code library's voldoen aan project eisen van Info Support:

- Er moet gebruik gemaakt worden van een stabiele release
- Er moet actief worden ontwikkeld aan de library

Daarnaast heb ik een aantal eisen opgesteld op basis van de kenmerken van het project. Dit zodat de code library's passen bij het project en dat er niet te veel overhead in zit.

- De library moet voor Java zijn
- De library moet geen 3d engine bevatten
- De library moet een visualisatie kunnen maken met het bron bewerking bestemmingsmodel

Uit dit onderzoek komt naar voren dat er een drietal code library's zijn: Netbeans Visual, JUNG en yWorks. In het onderzoek adviseer ik om yWorks niet te gebruiken. Dit omdat er jaarlijkse licentie kosten aan verbonden zijn. Het onderzoek geeft dus een advies over de te gebruiken code library's en raad hierbij het gebruik van bestaande software af.

12.2 Beroepstaak: 1.4 Uitvoeren analyse door definitie van requirement

Tijdens dit project heb ik requirements opgesteld. Er zijn requirements in verschillende onderdelen van het project naar voren gekomen. In het onderzoek heb ik een visualisatie vorm voor een data flow aangeraden. Om tot passende visualisatie vormen te komen heb ik een aantal requirements nodig, die niet waren beschreven in de case, zoals waar bestaat een data flow uit. Door een aantal gerichte vragen te stellen over data flows kwamen er bij het onderzoek een aantal requirements naar voren.

Tijdens een brainstorm sessie heb ik de requirements uit de case teruggekoppeld aan de opdrachtgever. Verder heb ik tijdens deze sessie met de opdrachtgever gebrainstormd. Hieruit komt naar voren dat het advies uit het onderzoek zal worden opgevolgd. Echter komen er ook een aantal nieuwe requirements naar voren. Zoals het kunnen filteren van de data, het opslaan van layouts en meerdere bewerkingssoorten. Deze requirements zijn bij de eerder gevonden requirements gezet in het requirements rapport.

Verder heb ik een aantal niet functionele requirements opgesteld. Hierbij maak ik een scheiding in de requirements op functionele- en niet functionele requirements. Op basis van de requirements heb ik een use case diagram opgesteld. Hierbij heb ik de requirements logisch ondergebracht in functionaliteiten van de applicatie.

Nadat de requirements bekend waren heb ik met de stakeholder de requirements geprioriteerd. Dit heb ik gedaan door de MoSCoW methode toe te passen. Hierbij heb ik van de belangrijkste requirements een use case beschrijving gemaakt. Dit alles om met de stakeholders te valideren dat de requirements kloppen en dat de interactie tussen actor en systeem duidelijk is.

De elementen van dit proces zijn terug te vinden in hoofdstuk: 7.1 en het begin van hoofdstuk 9. Verder is het requirements document en het functioneel ontwerp terug te vinden in de bijlagen (bijlagen F). Hierin kan het geheel worden nagelezen.

12.3 Beroepstaak: 2.3 Uitvoeren gegevensconversie

De originele opdracht zou werken met een klanten case. Echter heeft de klant van Info Support afgezien van dit project. Hierdoor kan/mag de data niet worden geëxtraheerd uit de productie omgeving van de klant. Ik heb daarom gewerkt met een bestaande testcase van Info Support. Deze data hoefde niet worden verkregen door middel van een gegevensconversie. Deze beroepstaak is daarom komen te vervallen.

12.4 Beroepstaak: 3.2 Ontwerpen systeemdeel

Tijdens dit project heb ik een data flow visualisatie module gemaakt voor het bestaande metadata platform van Info Support.

Tijdens het ontwerpen heb ik gebruik gemaakt van schermontwerpen om de data flow visualisatie overeen te komen met de wensen van de opdrachtgever. De schermontwerpen zijn gebaseerd op de tekeningen van de opdrachtgever tijdens de brainstormsessie. Verder heb ik door middel van use case beschrijvingen de interactie tussen actor en systeem ontworpen. Hierbij worden de functionaliteiten van de applicatie schematisch in beeld gebracht.

Verder heb ik tijdens het ontwerpen een duidelijke scheiding tussen de componenten aangebracht. Zo heb ik de view losgekoppeld van de model. De GUI heb ik zo simpel mogelijk gehouden om de gebruiker een duidelijke interface te kunnen tonen.

Verder heb ik een object georiënteerd ontwerp gemaakt. Hierbij houdt ik rekening met eventuele aanpassingen in de toekomst. Omdat ik gebruik maak van een bestaande database is de logica van deze database gescheiden van de rest van de applicatie. Zodat eventuele aanpassingen aan de database verminderde impact hebben op het aanpassen van de data flow visualisatie module. Verder maak ik gebruik van een factory pattern om de widgets aan te maken en de juiste informatie in op te slaan. Dit door verschillende soorten widget objecten te maken, die hun eigen informatie vasthoudt.

De ontwerpen zijn gemaakt met UML diagrammen. Hierbij heb ik een klassendiagram gemaakt die de verschillende klassen van de applicatie specificeert. Verder heb ik gebruik gemaakt van sequentie diagrammen om het gedrag van

de applicatie te beschrijven bij verschillende situaties. Deze ontwerpen zijn terug te vinden in dit document hoofdstuk 7.2 en 9.1. Verder zijn de gehele ontwerpen terug te vinden in het functioneel- en technisch ontwerp.

12.5 Beroepstaak: 3.3 Bouwen applicatie

De ontwerpen van de data flow visualisatie tool heb ik gebruikt om de applicatie te programmeren. Door deze ontwerpen te gebruiken heb ik een object georiënteerde applicatie gemaakt. Waarbij ik een aantal functies heb gemaakt die ervoor zorgen dat de database zo min mogelijk hoeft te worden benaderd. Ook heb ik rekening gehouden (ook in het ontwerp) op toekomstige aanpassingen. Door een objectgeorienteerde applicatie te bouwen is de code gescheiden en onderhoudbaar. Door dit in de eerste bouwfase toe te passen kon ik in de tweede bouwfase met toevoegingen en zo min mogelijk aanpassingen de applicatie uitbreiden.

Verder heb ik de applicatie geschreven in Java. Hierbij heb ik gebruik gemaakt van een aantal library's. Namelijk JDBC voor de connectie met de database. JUnit om unit tests te kunnen maken en Netbeans Visual om de data flow te visualiseren.

De source code heb ik in Git gezet zodat ik gebruik kan maken van versiebeheer. Dit heb ik ook gedaan zodat Daan van Berkel code reviewing kon doen op de code. Hij beoordeelde de code op kwaliteit en gaf aan waar deze kon worden verbeterd (door middel van Issues op Git).

De beschrijving van het proces van het bouwen van de applicatie wordt beschreven in hoofdstuk 8.1 en 10.1

12.6 Beroepstaak: 3.5 Uitvoeren van en rapporteren over het testproces

Tijdens het bouwen van de applicatie heb ik gebruik gemaakt van JUnit tests. Met deze tests heb ik de verwachte in en uitvoer van specifieke methoden in de applicatie getest. Op deze manier heb ik de kwaliteit van de code verantwoord. Daarnaast heeft de code review er ook voor gezorgd dat de kwaliteit van de code wordt gewaarborgd.

Naast de unittests heb ik een testomgeving ingericht. Deze testomgeving heeft gebruik gemaakt van een goedgekeurde testdata van Info Support. Hierbij is de productieomgeving van klanten gesimuleerd. In deze testomgeving heb ik logische en fysieke testgevallen uitgevoerd. Deze zijn gebaseerd op de use cases zoals deze zijn beschreven in de ontwerpen. Verder heb ik gecontroleerde in en uitvoer gebruikt om de vorm van de data flow te kunnen testen. Hierbij stel ik een data flow visualisatie op, op basis van de test data. Daarna kijk ik of de tool de verwachte uitvoer toont. Al deze testtechnieken zijn in detail beschreven en zijn daarom herbruikbaar. Verder heb ik een performance test gedaan. Hierbij test ik de data flow module als er een grote hoeveelheid data moet worden getoond. De resultaten toonde aan dat er geen merkbare vertraging ontstaat als de data flow visualisatie module een grote data set moet tonen. Als laatste heb ik gebruik gemaakt van exploratory testing hierbij heb ik op basis van mij inhoudelijke kennis van de source code op zoek gegaan naar mogelijke fouten.

De resultaten zijn te vinden in het testrapport in de bijlagen (bijlagen G). Verder wordt het proces van testen besproken in hoofdstuk 8.2 en 10.2.

13. Bijlagen

13.1 Verklarende woordenlijst

Woord	Verklaring
Aggregatie bewerking	Data aggregatie is een proces waarin informatie wordt verzameld en uitgedrukt in een beknopte vorm, voor doeleinden zoals statistisch analyseren
Code library	Een verzameling van routines, methoden, objecten met als doel om software te ontwikkelen.
Data flow	Het verplaatsen van data tussen twee data bronnen
ISMetadata	Een programma dat in de controle omgeving staat. Met als doel de metadata te beheren van de verschillende onderdelen van het systeem
Mapping	Elementen van verschillende databronnen, die aan elkaar zijn gekoppeld
Metadata	Data die andere data beschrijft
Repository	Een opslagplaats voor de source code van het project. Met als doel het kunnen toepassen van versiebeheer
Scalar bewerking	Een scalaire bewerking haalt een enkel datapunt uit een verzameling van data

Afkorting	Betekenis
CRUD	Create, Read, Update en Delete
DBMS	Database Management Systems
ETL	Extract Transform Load
FAT	Functionele Acceptatie test
GAT	Gebruikers Acceptatie test
MoSCoW	Must have, Should have, Could have, Won't have (zie ook hoofdstuk 7.1.2

13.2 Bronnenlijst

[1] http://www.infosupport.com/OverInfoSupport
[2] http://www.infosupport.com/MissieEnKernwaarden
[3] http://carriere.infosupport.com/afstuderen-bi-datavisualtisatie-bi-metadata
[4] http://www.techopedia.com/definition/26204/enterprise-data-warehouse
[5] http://en.wikipedia.org/wiki/OLAP_cube
[6] http://nl.wikipedia.org/wiki/Watervalmethode
[7] http://www.executivebrief.com/software-development/waterfall-rup-agile/
[8] http://nl.wikipedia.org/wiki/Scrum_(softwareontwikkelmethode)
[9] https://www.cprime.com/2012/09/when-to-use-scrum/
[10] http://nl.wikipedia.org/wiki/Rational_Unified_Process
[11] http://www.slideshare.net/maheshpanchal1/rup-1226744

[12] http://en.wikipedia.org/wiki/Software_development_process
[13] http://blog.softelegance.com/rup/rup-artefacts-in-a-context-of-phases/
[14] http://www.oracle.com/technetwork/java/javafx/overview/faq-1446554.html#6
[15] http://bits.netbeans.org/dev/javadoc/org-netbeans-api-visual/overview-summary.html
[16] https://platform.netbeans.org/tutorials/nbm-quick-start-visual.html
[17] http://jung.sourceforge.net/
[18] http://www.yworks.com/en/products_yfiles_about.html
[19] http://smartbear.com/all-resources/articles/what-is-code-review/
[20] http://thecodist.com/article/sql-injections-how-not-to-get

13.3 Bijlagenlijst

Bijlage	Naam
A	Afstudeerplan
B	Plan van aanpak
C	Onderzoeksrapport
D	Requirementsrapport
E	Functioneel ontwerp
F	Technisch ontwerp
G	Testrapport

Afstudeerplan

Informatie afstudeerder en gastbedrijf (*structuur niet wijzigen*)

Afstudeerblok: 2015-1.1 (start uiterlijk 9 februari 2015)
Startdatum uitvoering afstudeeropdracht: 9 februari 2015
Inleverdatum afstudeerdossier volgens jaarrooster: 5 juni 2015

Studentnummer: 11112131
Achternaam: dhr Koene (*) *weghalen niet van toepassing*
Voorletters: M.W.
Roepnaam: Maarten
Adres: Florens van Brederodelaan 228
Postcode: 2722XN
Woonplaats: Zoetermeer
Telefoonnummer: 0611043029
Mobiel nummer: 0611043029
Privé emailadres: maarten.koene@gmail.com

Opleiding: Informatica
Locatie: Zoetermeer (*) *weghalen niet van toepassing*
Variant: voltijd

Naam studieloopbaanbegeleider: R. Ruijsenaars
Naam begeleidend examinator: Arno Nederend
Naam tweede examinator: Tim Cocx

Naam bedrijf: Info Support
Afdeling bedrijf: Business Intellegence
Bezoekadres bedrijf: Kruisboog 42
Postcode bezoekadres: 3905TG
Postbusnummer: -
Postcode postbusnummer: -
Plaats: Veenendaal
Telefoon bedrijf: +31 318552020
Telefax bedrijf: +31 318552355
Internetsite bedrijf: www.infosupport.com

Achternaam opdrachtgever: dhr van Bilsen (*) *weghalen niet van toepassing*
Voorletters opdrachtgever: T.
Titulatuur opdrachtgever: -
Functie opdrachtgever: Consultant Business Intelligence
Doorkiesnummer opdrachtgever: -
Email opdrachtgever: Thomas.vanBilsen@infosupport.com

Achternaam bedrijfsmentor: dhr Hijl (*) *weghalen niet van toepassing*
Voorletters bedrijfsmentor: P.
Titulatuur bedrijfsmentor: -
Functie bedrijfsmentor: Afstudeerbegeleider
Doorkiesnummer bedrijfsmentor: -
Email bedrijfsmentor: Pascalle.Hijl@infosupport.com

Doorkiesnummer afstudeerder: -
Functie afstudeerder (deeltijd/duaal): -

Titel afstudeeropdracht:
Dataflowvisualisatie voor BI metadata platform

Opdrachtomschrijving *(toelichtende tekst verwijderen)*

1. Bedrijf
Info Support is een bedrijf dat zich specialiseert in ontwikkelen, beheren en hosten van software op maat. De producten, die worden gemaakt voor klanten, is software. Deze worden speciaal ontwikkeld voor de klant met de hulp van standaarden. Op deze wijze worden er stukken software opgeleverd die goed onderhoudbaar en uitbreidbaar zijn.

Het bedrijf telt meer dan 300 werknemers. Deze zijn verdeelt over de verschillende locaties in Nederland en Belgie. Verder wil Info Support zich onderscheiden van andere bedrijven door trainingen te verzorgen over een grote verscheidenheid van onderwerp. Ook wordt er binnen het bedrijf gebruik gemaakt van een wekelijkse bijeenkomst. Hierbij worden de verschillende trends en ontwikkelingen onder de loep genomen.

2. Probleemstelling
Bij Business Intelligence (BI) en data migratie projecten komt het vaak voor dat er gegevens van de ene database naar de andere database verplaatst moeten worden.

Dit verplaatsen van gegevens wordt ook wel dataflow genoemd. Alhoewel het implementeren van deze dataflows vaak niet erg complex is neemt het doorgaans toch veel tijd in beslag. Tevens zijn de werkzaamheden vaak nogal repeterend van aard.

Bij BI projecten wordt tegenwoordig steeds vaker gebruik gemaakt van metadata. Er worden dan zo veel mogelijk gegevens over de structuur en het proces van het BI systeem vastgelegd. Met deze gegevens kan vervolgens een groot deel van de code worden gegenereerd. Dit heeft als voordeel dat de kans op fouten afneemt, de ontwikkeltijd afneemt en dat er door de ontwikkelaars minder repetitief werk uitgevoerd hoeft te worden.

Info Support heeft voor Business Intelligence projecten een platform ontwikkeld om metadata vast te leggen en hieruit code te generen. Echter is het probleem dat de dataflow niet meer duidelijk te herleiden is met alle tussen databronnen. Zo wordt uit de externe bronsystemen een exacte kopie gemaakt. Die kopie wordt vervolgens via allerlei andere databronnen aangepast en geaggregeerd.

Het veranderen van het datawarehouse(en dus de metadata) geeft problemen. Door de structuur tussen de verschillende databronnen grafisch weer te geven kan dit probleem worden verminderd. Hiermee wordt bedoelt de huidige metadata in een grafische interface te laten zien. Op die manier moet de source, bewerkingen/logica en destination van de data in beeld gebracht.

3. Doelstelling van de afstudeeropdracht
De afstudeeropdracht zal uit drie hoofdonderdelen bestaan. Het eerste onderdeel is het onderzoeken van de verschillende bestaande manieren om dataflows te visualiseren. Denk hierbij bijvoorbeeld aan UML programma's of andere database structuur beschrijvende data. Nadat dit voltooid is kan het tweede deel van de opdracht worden gestart. Hierbij zal er onderzoek worden gedaan naar de verschillende code library's en beschikbare software voor dataflowvisualisatie. Het laatste deel van de opdracht zal het ontwerpen en implementeren van de module om dataflowvisualisatie te bewerkstelligen zijn.

4. Resultaat
Als de afstudeeropdracht is voltooid zal er een dataflowvisualisatie module zijn ontwikkeld. Verder is de module gedocumenteerd opgezet, waarbij er onderzoek is gedaan naar de mogelijkheden tot dataflowvisualisatie. Hierbij zijn bestaande oplossingen overwogen en is waar mogelijk gebruik gemaakt van bestaande software of library's.

Dit moet voornamelijk bij de Business Intelligence tak, het inzien van en aanpassen van gegenereerde metadata versimpelen. Er is dus een grafische omgeving ontstaan die de ontwikkelaars en ontwerpers inzicht geeft in de dataflow van metadata. Hierbij kunnen de ontwerpers aanpassingen doen de opbouw van het datawarehouse op een grafische achtergrond. Verder kunnen de ontwikkelaars op basis van deze ontwerpen, uit de module, de bronsystemen en tussenbronnen(de metadata) aanpassen zodat het datawarehouse conform het vernieuwde ontwerp zal zijn. De module geeft inzicht in de huidige metadata zodat deze makkelijker kan worden aangepast.

5. Uit te voeren werkzaamheden, inclusief een globale fasering, mijlpalen en bijbehorende activiteiten

- Onderzoek naar verschillende mogelijkheden van dataflow visualisatie (7dagen)
- Onderzoek naar beschikbare software en code library's voor het visualiseren van dataflows (5dagen)
- Een plan van aanpak schrijven(2dagen)
- Requirements opstellen aan de hand van eisen en wensen van de opdrachtgever, eventueel aan de hand van interviews(5dagen)
- Ontwerpen van de dataflow visualisatie module, beschreven in een functioneel en technisch ontwerp(10dagen)
- Ontwikkelen van de dataflow visualisatie module(30dagen)
- Testen (5dagen)
- Implementeren en overdracht(6dagen)

6. Op te leveren (tussen)producten

- Onderzoeksrapport verschillende mogelijkheden datavisualisatie
- Onderzoekrapport beschikbare software en code library's
- Plan van aanpak
- Requirements rapport
- Functioneel en technisch ontwerp
- Dataflow module
- Testrapportage

7. Te demonstreren competenties en wijze waarop

1.3 Selecteren van standaardsoftware
In deze opdracht worden een onderzoek gedaan naar de verschillende mogelijkheden van dataflow visualisatie. Verder wordt er een onderzoek gedaan naar de bestaande standaardsoftware en code library's. De onderzoeken worden door mij uitgevoerd. De uitkomsten worden meegenomen in het ontwerpen van de dataflow module.

1.4 Uitvoeren analyse door definitie van requirements
De exacte requirements staan nog niet vast. Deze zullen door interview, bestaande wensen en eisen worden vastgesteld. Hierbij wordt er kritisch met de opdrachtgever nagedacht over de toekomstige mogelijkheden van de module.

2.3 Uitvoeren gegevensconversie
Tijdens het bouwen van de module wordt er gewerkt met gegevensconversies. Hierbij wordt de data uit de live omgeving verplaatst naar de staging database. Deze gegevens stromen moeten visueel laten zien worden. Hierbij zal de data via gegenereerde scripts de data worden verplaatst. Deze scripts zullen ook in het grafische interface worden verwerkt. Hiervoor zullen de scripts eventueel moeten worden aangepast.

3.2 Ontwerpen systeemdeel
Tijdens deze opdracht wordt een dataflow module ontworpen voor MS-Access. Dit wordt gebaseerd op eerder genoemde onderzoeken en vastgestelde requirements. Dit zal ik als afstudeerder zelfstandig tot stand laten komen.

3.3 Bouwen applicatie

Na het onderzoeken en ontwerpen zal de dataflow module worden gebouwd. De standaardsoftware wordt eventueel ingebouwd. Verder wordt de module volgens de requirements en ontwerpen opgebouwd door mij.

3.5 Uitvoeren van en rapporteren over het testproces

Op basis van de, door mij opgestelde, testplannen zal de module worden getest op requirements en integratie met de andere systemen. Zoals eerder genoemd zal dit een testomgeving plaatsvinden. Het testproces zal worden gedocumenteerd en als onderdeel van de documentatie dienen van de dataflow module.

Plan Van Aanpak

Dataflowvisualisatie voor BI metadata platform

Plan van aanpak

Titel	Plan van aanpak
Project/Onderwerp	Dataflowvisualisatie voor BI platform
Versie	1.3
Status	Definitief
Datum	19-02-2015
Bestand	Plan van aanpak 2015
Bedrijf	Info Support

Info Support B.V. Kruisboog 42, 3905 TG, Veenendaal, Tel. +31 (0) 318 55 20 20 - Fax +31 (0) 318 55 23 55
Kenniscentrum De Smalle Zijde 39, 3903 LM Veenendaal, Tel. +31 (0) 318 50 11 19 - Fax +31 (0) 318 51 83 59

Historie

Versie	Status	Datum	Auteur	Verandering
1.0	Concept	16-2-2015	Maarten Koene	Creatie
1.1	Eerste definitieve versie	19-2-2015	Maarten Koene	Feedback opdrachtgever verwerkt
1.2	Definitieve versie	12-3-2015	Maarten Koene	Bijstellen van planning en verwerkte feedback
1.3	Definitieve versie	23-3-2015	Maarten Koene	Spelling en feedback

Management samenvatting

In de Business Intelligence wordt er gebruik gemaakt van datawarehouses. In een datawarehouse worden gegevens uit meerdere bronsystemen gecombineerd. Bij Info Support is er een tool ontwikkelt, de ISMetadata, die het ontwerpen en creëren van Datawarehouses voor een groot gedeelte automatiseert. De ISMetadata tool wordt gebruikt om voor klanten van Info Support datawarehouses te ontwikkelen. Om zo een werkend BI systeem neer te kunnen zetten.

Deze tool geeft ontwikkelaars en ontwerpers een mogelijkheid datavault(s) te genereren (en uiteindelijk datamarts). Daarnaast kunnen deze databases worden aanpast. Echter is het overzicht houden bij het aanpassen van een of meerdere delen een probleem. In dit project zal er in samenwerking met de opdrachtgever een visualisatietool worden ontwikkeld. Dit zal gebeuren volgens de RUP methode, zodat er ruimte is voor het inwerken van de nog onbekende requirements. Verder wordt er voor het afstuderen van de opdrachtnemer een afstudeerverslag gemaakt. Deze zal naar de tweede opdrachtgever gaan, de Haagse Hogeschool.

Het uiteindelijke resultaat zal een serie documenten zijn die een beschrijving geven van de visualisatietool. Verder wordt er een data flow visualisatie tool opgeleverd. Deze zal gebruik maken van een demo database tijdens het ontwikkelen. Uiteindelijk kan deze visualisatie tool gebruikt worden bij klanten om dynamisch een weergave te maken van de metadata, die is gegenereerd door de ISMetdata.

Om de exacte eisen aan de software in kaart te brengen wordt Thomas van Bilsen geraadpleegd. Verder worden er requirements opgesteld op basis van de resultaten van het onderzoek dat plaats zal vinden in dit project. Hierbij wordt een iteratieve werkwijze gebruikt op het opstellen van de requirements en het ontwikkelen. Hierbij kan de opdrachtgever continu feedback geven. Onderstaande lijst geeft de op te leveren documenten op:

- Plan van aanpak
- Requirementsrapport
- Onderzoeksrapport
- Functioneel ontwerp
- Technisch ontwerp
- Dataflowvisualisatie module
- Testrapport
- Afstudeerverslag

Vanuit Info Support en de Haagse Hogeschool zijn er een aantal betrokkenen. In onderstaande lijst de namen van de betrokkenen en hun functie tijdens dit project:

-	Maarten Koene	Opdrachtnemer
-	Thomas van Bilsen	Opdrachtgever Info Support
-	Arno Nederend	Opdrachtgever Haagse Hogeschool
-	Tim Cocx	Opdrachtgever Haagse Hogeschool
-	Orhan Uyan	Technisch begeleider Info Support
-	Pascalle Hijl	Procesbegeleider Info Support
-	Henk Brands	Unit Manager BI Info Support

Inhoudsopgave

MANAGEMENT SAMENVATTING 4

1 OPDRACHT 6

1.1 Opdrachtgever en opdrachtnemer 6
1.2 Opdrachtdefinitie 7
1.3 Afbakening 7
1.4 Afhankelijkheden 9
1.5 Kwaliteitseisen 9
1.6 Uitgangspunten 9
1.7 Randvoorwaarden 10

2 RISICOMANAGEMENT 11

3 AANPAK 12

4 BEHEERSASPECTEN 15

4.1 Organisatie 15
4.2 Informatie 16
4.3 Tijd 17
4.4 Kwaliteit 17
4.5 Overlegvormen 18

5 OPLEVERING 20

1 Opdracht

1.1 Opdrachtgever en opdrachtnemer

De opdrachtnemer voor de in dit plan van aanpak beschreven delen is:

Maarten Koene

De opdrachtgever voor de in dit plan van aanpak beschreven delen is:

Thomas van Bilsen	Info Support
Arno Nederend	Haagse Hogeschool
Tim Cocx	Haagse Hogeschool

1.2 Opdrachtdefinitie

Business Intelligence(BI) systemen worden steeds meer ingezet. Hierbij worden uit meerdere databronnen(databases) informatie verzamelt. Om de gegevens te kunnen gebruiken in het datawarehouse moeten deze worden aangepast. Om dit te doen wordt er gebruik gemaakt van ETL schema's. Hiermee wordt de data uit de bronnen gehaald en bewerkt voor het datawarehouse. Bij bewerkingen kan worden gedacht aan het combineren van een of meerdere velden, wiskundige berekeningen of veranderen van datatype. In het hoofdstuk afbakening wordt ingegaan op de architectuur die gebruikt wordt binnen Info Support.

Het verplaatsen van data van een bron naar een bestemming heeft een data flow. Het implementeren van data flows is repetitief werk. Hiervoor is een geautomatiseerde werkwijze ontwikkeld. Bij deze geautomatiseerde wijze wordt de structuur en de bewerkingen opgeslagen in een metadatabron. Deze structuur beschrijft de databronnen, de bewerkingen op de data van databronnen en datawarehouse.

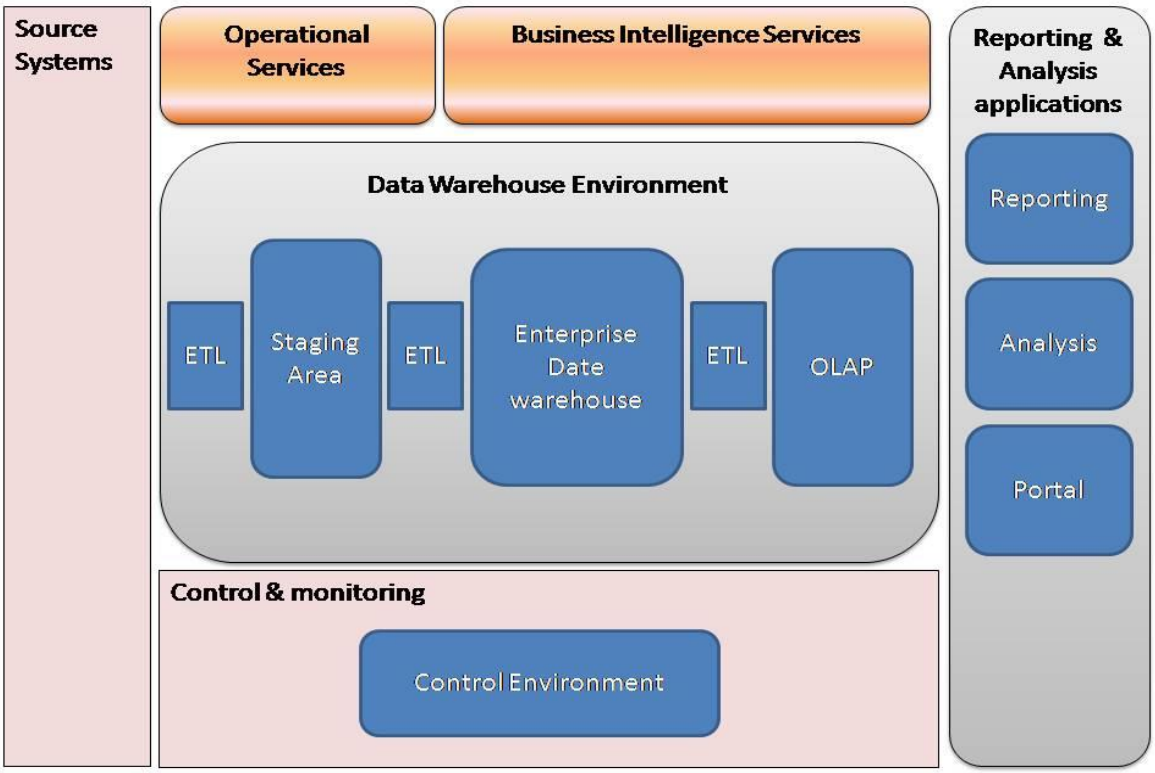
Het probleem van de metadata is dat deze op een beperkte wijze kan worden gevisualiseerd. Hierbij wordt het aanpassen van de metadata voor de ontwerpers een tijdrovende en ingewikkelde taak. Ook wordt er door de beperkte visualisatie mogelijkheden de kans op fouten in de code en dus metadata vergroot. De ontwikkelaars kunnen het overzicht van de verschillende systemen kwijt raken. Door dit probleem kunnen er onvoorziene fouten ontstaan bij het aanpassen van een bestaand datawarehouse.

Dit project is gefocust op het ontwerpen en ontwikkelen van een visualisatie tool voor de metadata. Met de visualisatietool moet de gegenereerde metadata worden gevisualiseerd. Verder moet de metadata grafisch kunnen worden aangepast. Door de tool zal het genereren van code en het aanpassen van de metadata versimpelen en wordt de kans op het maken van fouten in de code geminimaliseerd.

Verder is er voor de opdrachtnemer een tweede doelstelling. Deze doelstelling is het maken van een afstudeerverslag. Dit afstudeerverslag zal worden geleverd aan de opdrachtgevers van de Haagse Hogeschool.

1.3 Afbakening

Onderstaand figuur illustreert de huidige referentie architectuur bij Info Support. Dit figuur is gehaald uit het BI architectuur beschrijvingsdocument van de BI unit van Info Support.



De "source systems" van Info Support zijn niet geschikt voor analyse. Deze systemen draaien in verschillende omgevingen (test-, acceptatie-, ontwikkel- of productieomgeving). Deze omgevingen zijn voor een ander doeleinde opgezet dan Business Intelligence doelen. Om de gegevens te gebruiken worden deze gekopieerd naar de staging area. Ook kunnen er gegevens worden gehaald uit losse bestanden, denk hierbij aan excel sheets of csv files. Nadat de data uit de systemen en bestanden is geëxtraheerd, wordt er bewerkingen (ETL) uitgevoerd op de data. Dit kan variëren van nul tot meerdere bewerkingen. De structuur van het datawarehouse en de bewerkingen(bijvoorbeeld de formules van een berekening) op de data worden opgeslagen in de metadata database. De structuur en vorm van de metadata zal tijdens het onderzoek in dit project worden vastgesteld.

Als de data uiteindelijk bij het datawarehouse aankomt wordt er soms een geheel andere structuur gegeven aan de data. De structuur staat beschreven in de metadata. De metadata in het bovenstaande diagram is weergegeven in de "control environment". Het gebruik van het datawarehouse wordt gedaan door de reporting & analysis applications.

Door Info Support is een ISMetadata tool gemaakt. Deze maakt op basis van een aantal datawarehouse standaarden en brongegevens een conceptueelmodel van een datawarehouse. De ISMetadata tool genereerd de metadata van het datawarehouse.

Tijdens de ontwikkeling van de dataflowvisualisatie tool wordt gefocust op de metadata die wordt gegenereerd door de ISMetadata tool. Deze staat in bovenstaande architectuur in de control & monitoring.

De volgende onderwerpen zijn geen onderdeel van dit project:

1. Bronsystemen ontwerpen of aanpassen
2. ETL schema's ontwerpen of aanpassen

3. Het datawarehouse ontwerpen of aanpassen
4. Ontwerpen of aanpassen van een database

De volgende onderwerpen zijn wel onderdeel van dit project:

1. Dataflow tussen de verschillende systemen visualiseren
2. Grafische weergave maken van de gegenereerde metadata uit de ISMetadata tool
3. Eisen stellen aan de metadata die wordt gegenereerd door de ISMetadata tool
4. Aanpassingen in een grafische interface op de metadata

1.4 Afhankelijkheden

Het project heeft een aantal afhankelijkheden. In dit hoofdstuk zullen de afhankelijkheden in kaart worden gebracht. Hierbij moet gedacht worden aan externe systemen of organisatorische problemen.

1. De metadata moet voldoende informatie bevatten om de gewenste visualisatie te produceren. Dit zal worden beoordeelt door de opdrachtnemer.
2. De software, die benodigd is om het visualiseren te bewerkstelligen, moet compatible zijn met de gebruikte MSSQL software
3. De metadata moet een beeld geven van het datawarehouse, hierbij moeten alle onderdelen worden weergegeven. Dus de stappen die data heeft doorgemaakt en de uiteindelijke samenkomst in het datawarehouse.

1.5 Kwaliteitseisen

Er zijn een aantal eisen aan het eindproduct van dit project. In dit hoofdstuk worden de kwaliteitseisen opgesomd.

1. De metadata wordt in een grafische interface weergegeven.
2. De grafische interface is ontworpen naar de wensen van de ontwikkelaars en ontwerpers. Deze zullen worden vertegenwoordigd worden door de opdrachtgever van Info Support.
3. De metadata kan in de grafische interface worden bewerkt.
4. De wijzigingen wordt door de visualisatietool opgeslagen in de metadata database
5. De metadata moet worden geïnterpreteerd door de visualisatietool
6. Er zal gebruik gemaakt worden van Java. Dit zodat er Operating systeem onafhankelijk gewerkt kan worden. Ook heeft de opdrachtnemer ervaring met het gebruik van Java.

1.6 Uitgangspunten

Voor uitvoering van de in dit plan van aanpak beschreven delen zijn de onderstaande uitgangspunten van toepassing:

1. Ontwikkeling van de module wordt gedaan bij Info Support op de locaties Zoetermeer en Veenendaal.
2. Hard- en software zal worden verzorgd door Info Support
3. Benodigde trainingen en (interne) opleidingen worden verzorgd door Info Support
4. De benodigde trainingen zullen worden vastgesteld tijdens het project
5. De benodigde trainingen worden in overleg met de technische begeleider (Orhan Uyan) aangevraagd

6. De metadata tool (ISMetadata) wordt verzorgd door Info Support
7. Er wordt gewerkt met een demo datawarehouse van Info Support

1.7 Randvoorwaarden

Aan uitvoering van de in dit plan van aanpak beschreven delen zijn de volgende randvoorwaarden gesteld:

1. Er wordt een onderzoek verricht naar de verschillende technieken van dataflowvisualisatie. Er wordt gekeken welke manieren er bestaan. Welke voor- en nadelen er zijn aan de bestaande manieren. Dit zal uiteindelijk uitsluitsel geven over het gebruik van bestaande software voor de dataflowvisualisatie.
2. Op basis van code library's naar voren gekomen uit het onderzoek zal een advies worden gegeven omtrent het gebruik hiervan.
3. Er wordt toegang gegeven tot de metadata tool en deze werkt samen met de demo case van Info Support
4. Info Support zal de opdrachtnemer voorzien van de benodigde hard- en software
5. Er moet tijd zijn bij de technisch begeleider, opdrachtgever, unit manager en proces begeleider voor het stellen van vragen en het krijgen van feedback
6. De opdrachtnemer werkt 40 uur per week voor een periode van 17 weken

2 Risicomanagement

Onderkende risico's met maatregelen ter beheersing
Voor uitvoering van de in dit plan van aanpak beschreven delen zijn de onderstaande risico's onderkend. Bij de risico's zijn de oorzaken en bijbehorende maatregelen ter beheersing opgenomen.

Nr.	Risico omschrijving				
	Oorzaak	Maatregel	P/S	Wie	Wanneer
1	De metadata bevat niet voldoende informatie om de juiste visualisatie te produceren.				
	Missende metadata	Gebruik maken van testdata	S	Info Support	Opstellen van het functioneel ontwerp
	Incomplete metadata	Eisen stellen aan de benodigde metadata	P	Opdrachtnemer	Opstellen van het functioneel ontwerp
2	De software, die benodigd is om het visualiseren te bewerkstelligen, is niet compatible met de gebruikte database software				
	Ondersteuning voor bijv. MSSQL is nog niet toegevoegd	Andere software gebruiken, die wel compatible is met MSSQL	S	Opdrachtnemer	Onderzoek
3	De metadata geeft geen volledig beeld van het ontworpen datawarehouse				
	Incomplete metadata	Meta data aanvullen met testdata	S	Info Support	Opstellen van het functioneel ontwerp
	Meta data niet toereikend genoeg door ontbreken informatie	Gebruik maken van een testomgeving die de metadata simuleert	S	Info Support	Opstellen van het functioneel ontwerp
4	De te gebruiken programmeertaal wordt nog niet beheerst door de opdrachtnemer				
	Nog geen gebruik gemaakt hebben van de programmeertaal	Een cursus volgen bij het kenniscentrum van Info Support	P	Opdrachtnemer	Bouwen van de applicatie
5	De tool voldoet niet aan de eisen van de doelgroep				
	De tool heeft de verkeerde functionaliteiten	De functionaliteiten vaststellen met behulp van de opdrachtgever	P	Opdrachtgever	Requirements opstellen
	De tool werkt niet volgens gestelde eisen	Iteraties op het ontwikkelproces invoeren. Om de gebruikersgroep te laten feedbacken	P	Opdrachtnemer	Testfasen

3 Aanpak

Het project zal gebruik maken van een software ontwikkelmethode. Deze methode is het Rational Unified Process (RUP). Bij deze ontwikkelmethode wordt gebruik gemaakt van een fasering. In dit hoofdstuk zullen de voordelen van het gebruik van RUP worden gegeven. Daarna wordt er per fase van het project de deelproducten en werkzaamheden weergegeven.

Om structuur te geven aan de ontwikkeling van de dataflowvisualisatie is er gekozen voor een software ontwikkelmethode. RUP is de ontwikkelmethode die hierbij wordt aangehouden. Bij het starten van het project zijn de exacte functionaliteiten nog niet vastgesteld. De opdrachtgever is nog niet geheel zeker wat de mogelijkheden zijn van de ISMetadata. Het onderstaande schema is dus een schatting van het aantal benodigde iteraties.

Door gebruik te maken van iteraties binnen RUP kunnen de verschuivende requirements worden bewerkstelligd. Binnen RUP is er ruimte om het onderzoek te plaatsen. Met dit onderzoek worden verschillende manieren van visualiseren en beschikbare software en code library's onderzocht. Hierbij zullen ook de eerste requirements naar voren komen.

Ook heeft de opdrachtnemer positieve ervaringen met het gebruik van RUP. Ook wordt er binnen Info Support regelmatig gebruik gemaakt van de fasering van RUP. Door deze software ontwikkelmethode te gebruiken, kan de opdrachtgever op de hoogte worden gesteld over de status van het project.

Inception fase

In de eerste fase van het project zal er een plan van aanpak worden geschreven. Hiermee wordt de structuur van het project vastgesteld. Verder zal hiermee een planning worden geschreven en de op te leveren producten te worden naar voren gebracht. Zodra het plan van aanpak goed gekeurd is door de opdrachtgever, kan het project worden gestart door de opdrachtnemer.

Op te leveren producten aan het einde van Inception fase:

Product naam	**Artefact RUP**
Plan van aanpak	Iteratie plan (planning)
	Risico lijst
	Business model (huidige situatie)
	Project scope

Onderzoeksfase (Onderdeel van inception fase)

Tijdens dit project zal er onderzoek plaatsvinden. Dit onderzoek heeft betrekking op de bestaande vormen van dataflowvisualisatie. Ook moeten hierbij de bestaande software en code library worden meegenomen. Door dit onderzoek te doen wordt achterhaalt of er bestaande oplossingen zijn voor dataflowvisualisatie en of eventueel gebruik gemaakt kan worden van reeds bestaande oplossingen. Om de richting van het onderzoek te bepalen wordt de opdrachtgever benaderd. Hieruit is het mogelijk om requirements op te stellen. Verder wordt in deze fase de te gebruiken databronnen en datawarehouse vastgesteld. De kwaliteit en toepasbaarheid van de bestaande metadata beoordeelt. Op deze manier kan de metadata eventueel, bij ontbrekende of missende data, worden vervangen voor testdata.
Op te leveren producten aan het einde van Onderzoeksfase:

Product naam	Artefact RUP
Onderzoeksrapport	Requirements
Requirements rapport v1.0	Initiële use case diagram
	Requirements

Elaboration fase 1

Om vast te stellen welke functionaliteiten en eisen het eindproduct zal moeten krijgen wordt er gebruik gemaakt van requirements. Deze eisen worden aan de hand van interviews en gesprekken met de opdrachtgever opgesteld. Ook worden de gevonden requirements uit het onderzoek besproken met de opdrachtgever. De resultaten uit het onderzoek worden verder geïmplementeerd in het ontwerp van de data flow visualisatie module. Denk hierbij vooral aan de te gebruiken code library's.

Op te leveren producten aan het einde van Elaboration fase 1:

Product naam	Artefact RUP
Requirements rapport v1.5	Use case diagram
	Requirements (functioneel en niet functioneel)
	Use case beschrijvingen
	Technische beperkingen
Functioneel ontwerp	Analyse klassendiagram
	Schermontwerpen
Technisch ontwerp	Tools
	Design klassendiagrammen
	Sequence diagrammen

Construction & testing fase

Na de ontwerp fasen wordt de bouwfase gestart. Hierbij wordt conform het functioneel- en technisch ontwerp de dataflowvisualisatie module gebouwd. Deze module zal na het bouwen worden getest aan de hand van de beschreven requirements. De testsoorten kunnen op een later stadium worden overeengekomen met de opdrachtgever.

Op te leveren producten aan het einde van fase:

Product naam	Artefact RUP
Data flow visualisatie module	Deployable versie
Testrapport	Fysiek testontwerp
	Testrapport

Elaboration fase 2

Nadat de opdrachtgever de eerste versie van de visualisatie module voor zich heeft gehad kunnen er nieuwe inzichten ontstaan. In deze fase worden de nieuwe inzichten en overgebleven requirements geëvalueerd. Hierbij wordt opnieuw besloten welke van de requirements moeten worden vervuld. Deze zullen daarna worden uitgewerkt in het ontwerp. Zodat deze in de hierna komende bouwfase kunnen worden gebouwd.

Op te leveren producten aan het einde van Elaboration fase 2:

Product naam	Artefact RUP
Requirements rapport v1.6	Use case beschrijvingen
Functioneel ontwerp	Analyse klassendiagram
Technisch ontwerp	Design klassendiagrammen
	Sequence diagrammen

Construction & testing fase

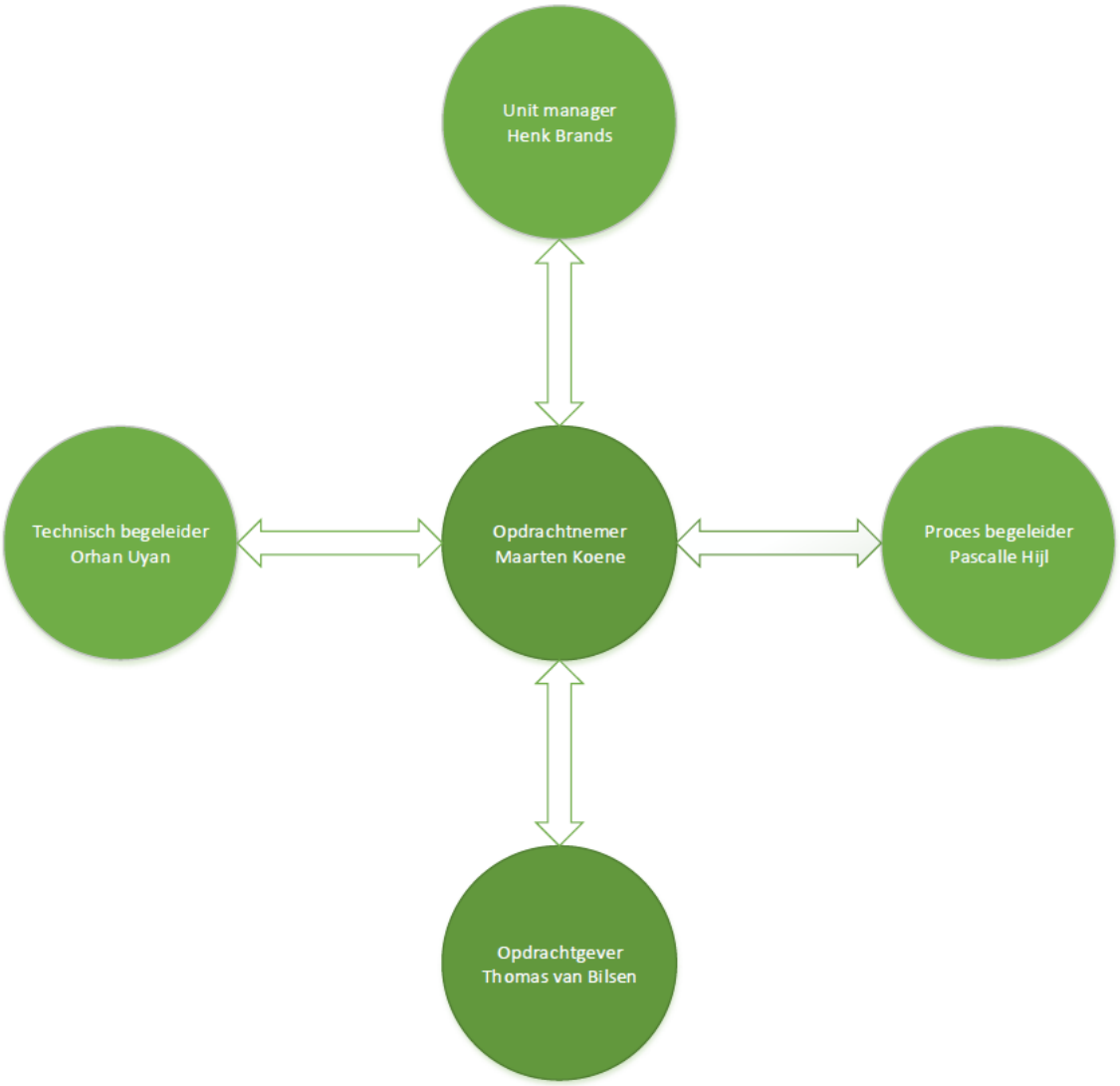
Na de ontwerp fasen wordt de bouwfase gestart. Hierbij wordt conform het functioneel- en technisch ontwerp de dataflowvisualisatie module gebouwd. Hierbij zullen de nieuwe requirements worden toegevoegd aan de bestaande visualisatie module. Deze module zal na het bouwen worden getest aan de hand van de beschreven requirements. Hierbij wordt gebruik gemaakt van de test set van de vorige construction fase.

Op te leveren producten aan het einde van fase:

Product naam	**Artefact RUP**
Data flow visualisatie module	Deployable versie
Testrapport	Fysiek testontwerp
	Testrapport

4 Beheersaspecten

4.1 Organisatie



Functie	Naam	Bevoegdheden
Opdrachtnemer	Maarten Koene	- Overleg plegen met alle verschillende partijen - Maken van de visualisatiemodule - Maken van een afstudeerverslag - Maken van mijlpaalproducten
Unit manager	Henk Brands	- Aanspreekpunt voor problemen binnen het bedrijf - Beoordelen functioneren opdrachtnemer binnen het bedrijf - Bijwonen oefenpresentatie
Technisch begeleider	Orhan Uyan	- Technische inhoud beoordelen - Benodigde trainingen bepalen - Vragen op technisch vlak beantwoorden - Beoordelen en feedback geven mijlpaalproducten
Proces begeleider	Pascalle Hijl	- Overleg plannen met opdrachtnemer - Voortgang van het project proces bewaken - Uitvoerende middelen regelen - Contactpersoon hogeschool
Opdrachtgever	Thomas van Bilsen	- Aangeven functionaliteiten eindproduct - Beoordelen en feedback geven mijlpaalproducten - Beoordelen afstudeerverslag - Eindzitting bijwonen
Proces begeleider	Arno Nederend	- Voortgang van het project proces bewaken - Beoordelen afstudeerverslag - Eindzitting bijwonen - Contact persoon van de hogeschool voor Info Support
Tweede examinator	Tim Cocx	- Beoordelen afstudeerverslag

4.2 Informatie

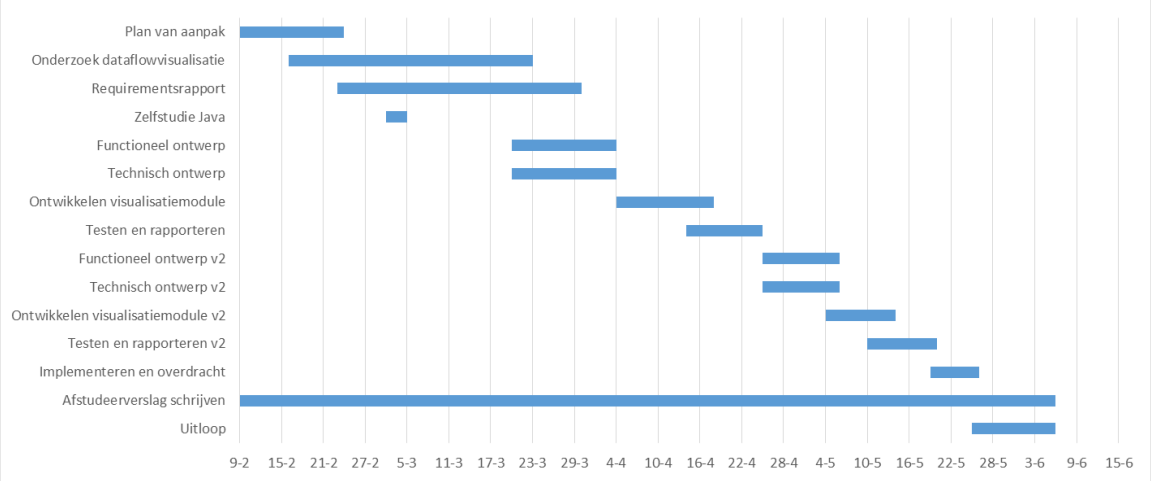
Tijdens dit project worden er een aantal mijlpaalproducten opgeleverd. In dit hoofdstuk wordt de omgang met de inhoudelijke informatie vastgelegd.

De mijlpaalproducten worden geproduceerd voor Info Support. De enige uitzondering op deze regel is het afstudeerverslag. Deze zal worden geproduceerd voor de Haagse Hogeschool. Daardoor zal dit mijlpaalproduct voldoen aan de eisen die de opdrachtgevers van de Haagse Hogeschool opleggen. De andere mijlpaalproducten zullen worden geproduceerd met de richtlijnen van Info Support als richtlijn.

De mijlpaalproducten blijven eigendom van de bijbehorende opdrachtgever. Echter worden de mijlpaalproducten door alle opdrachtgevers ingezien. Alle mijlpaalproducten worden toegevoegd aan het afstudeerverslag als bijlage. Dit wordt gedaan om inzicht in de werkwijze en beslissingen van het project te geven. Op verzoek van Info Support kan worden besloten om het afstudeerverslag niet buiten het bedrijf te laten komen. Als Info Support geen verzoek hiervoor indient, zal er een kopie van het afstudeerverslag (inclusief bijlagen) worden bewaard door de Haagse Hogeschool.

De geproduceerde software(source code) en documentatie blijft eigendom van Info Support. De rechten hierop zijn te verkrijgen bij Info Support. De Haagse Hogeschool behoudt zich het recht om inkijk te verkrijgen in de source code, maar zal deze niet kopiëren of op enige andere wijze ontvreemden.

4.3 Tijd



4.4 Kwaliteit

In het onderstaande overzicht wordt kort ingegaan op de mijlpaalproducten. Hierbij zal de inhoud globaal worden geschetst. Om de mijlpaalproducten per fase te zien wordt u doorverwezen naar het hoofdstuk aanpak.

Plan van aanpak
Het plan van aanpak beschrijft het project in een globale vorm. Hierbij wordt een software ontwikkelmethode gekozen. Alle belanghebbende bij het project worden naar voren gebracht. Op deze manier wordt vastgelegd wat wordt verwacht van de verschillende partijen. Ook worden de mijlpaalproducten beschreven. Voor de opdrachtgever de oplever data en globale planning en volgordelijkheid vastgelegd.

Requirementsrapport
De gewenste dataflowvisualisatie tool is globaal beschreven. Door gesprekken te voeren met de opdrachtgever zullen de gewenste functionaliteiten in beeld worden gebracht. Ook de niet functionele eisen zullen worden vastgelegd in dit document. Verder kunnen er requirements voortvloeien uit het te verrichten onderzoek.

Onderzoeksrapport
Er zijn twee onderwerpen waar duidelijkheid over geschapen dient te worden. Ten eerste moet voor het project bepaalt worden welke vorm van visualisatie het beste werkt voor dit project. Het tweede deel zal zich concentreren op de te gebruiken code library's van Java. Hieruit zal een advies volgen voor het vervolgen van het project.

Functioneel ontwerp
In het functioneel ontwerp worden de gevonden requirements omgezet naar functionaliteiten voor de dataflowvisualisatie tool. Ook de wensen van de

opdrachtgever en de resultaten van het onderzoek zullen gecombineerd in het ontwerp.

Technisch ontwerp
In het technisch ontwerp zullen de technische eisen worden gesteld. Hierbij worden de eisen voor de te gebruiken metadata vastgelegd. Ook wordt hier de te gebruiken technieken en onderhoudbaarheid vastgelegd.

Dataflowvisualisatie module
De dataflowvisualisatietool wordt opgeleverd op basis van de eisen van de opdrachtgever. Deze eisen zijn door het hele ontwikkelproces meegenomen. Door te feedbacken met de opdrachtgever (die garant staat voor de doelgroep) wordt de tool ontwikkeld op deze wijze.

Testrapport
Om de kwaliteit te waarborgen van de dataflowvisualisatie tool wordt er getest volgens officiële methodieken. Deze methodieken zullen de requirements moeten testen op basis van de functionaliteiten van de tool.

Afstudeerverslag
Het afstudeerverslag is voor de opdrachtgevers van de Haagse Hogeschool. Deze zal worden opgesteld aan de hand van het doorlopen proces, die ten gronde ligt aan het ontwikkelen van de dataflowvisualisatietool. Dit document bevat alle geproduceerde documenten en eventuele code als bijlagen.

Om de mijlpaalproducten op te leveren binnen de afgesproken tijd zal de procesbegeleider toezicht houden. Zij zullen met de opdrachtnemer een maal per twee weken de voortgang van het gesprek bespreken. Hierbij kunnen eventuele veranderingen in het proces worden aangebracht. Dit gebeurd in overleg met de opdrachtgever en technisch begeleider. In onderstaand hoofdstuk worden de overlegvormen gedetailleerder uitgelegd.

4.5 Overlegvormen

Om de voortgang van het project te controleren zijn er een aantal personen aangesteld.

Om de voortgang van het project te bespreken is er een maal per week een gesprek met de opdrachtgever. Tijdens deze gesprekken zullen de functionaliteiten en de reeds geboekte resultaten worden besproken. Op de geboekte resultaten zal feedback worden gegeven. Requirements zullen worden opgesteld aan de hand van deze gesprekken. De duur van de gesprekken is ongeveer een uur. Naast deze gesprekken zal er gebruik gemaakt worden van contact via de email. Hierbij zullen urgente vragen en documenten worden uitgewisseld.

De technische kant van het project zal worden gecontroleerd door een technisch begeleider. De technisch begeleider, Orhan Uyan, zal zorgen voor het keuren en beoordelen op technisch gebied. Hierbij worden de mijlpalen beoordeelt op inhoud met betrekking tot het technische aspect. Denk hierbij aan het reviewen van geschreven code, het beoordelen op UML diagrammen en de samenhang van het ontwerp. Met de technisch begeleider wordt een keer per 2 weken gesproken. Verder wordt er per email contact gehouden en overleg gepleegd tussen deze gesprekken door over de technische kant van het project.

De derde begeleider is de procesbegeleider. De procesbegeleider, Pascalle Hijl, zorgt voor de beheersing van het project in het algemeen. De procesbegeleider zal helpen bij het beheersen van de planning. Hierbij wordt de algemene voortgang bewaakt. Verder is de procesbegeleider de contactpersoon voor de opdrachtgevers van de Haagse Hogeschool.

Vanuit de Haagse Hogeschool is er een tweede procesbegeleider aangesteld. Deze begeleider zal tijdens de duur van het project twee maal worden gesproken. Dit gesprek is dan samen met de opdrachtgever en opdrachtnemer. Hierin zal de voortgang van het project worden besproken. De begeleider is ook het aanspreekpunt voor de opdrachtnemer. Deze gesprekken kunnen gaan over eventuele problemen binnen het bedrijf.

De unit manager is de hoofd van de afdeling van de opdrachtnemer. Gesprekken met de unitmanager worden een keer per maand gehouden. Deze gesprekken hebben als onderwerp het functioneren binnen het bedrijf. Enige vorm van escalatie van de opdracht kan ook worden besproken met de unit manager. Hiermee staat vooral het afstuderen centraal in de mogelijke escalatieprocedure.

5 Oplevering

Bij aanvang van dit project wordt een plan van aanpak opgesteld. Hierin worden de (deel)producten en te nemen stappen vastgelegd. De betrokken partijen en begeleiding wordt ook naar voren gebracht. In het plan van aanpak wordt de scope van het project bepaald. Dit alles gebeurd in overleg met de opdrachtgever. In een tijdspad wordt de tijdigheid van het project afgesloten. Dit wordt gedaan aan de hand van een planning. Er wordt aangegeven welk software ontwikkelmethode zal worden aangehouden bij de ontwikkeling van de dataflowvisualisatiemodule.

Het plan van aanpak geeft aan dat de opdrachtgever de gewenste functionaliteiten terug wil zien in het eindproduct. Om structuur en overzicht te geven aan deze functionaliteiten wordt er een requirementsrapport opgesteld. Dit rapport zal worden opgesteld uit resultaten van het onderzoek, maar ook uit de interviews met de opdrachtgever zullen requirements worden gehaald. Het rapport wordt opgebouwd in twee fasen van het project. Hierbij zal er een iteratie plaatsvinden op de eerste versie. Door deze iteratie kan er gezorgd worden dat alleen de essentiële requirements worden meegenomen in het uiteindelijke ontwerp.

Tijdens het onderzoek zullen de eerste requirements naar voren worden gebracht. Het onderzoek focust zich op bestaande manieren van dataflowvisualisatie. Hierdoor biedt het resultaat van dit onderzoek als basis voor het uiteindelijke eindproduct. Uit het onderzoek kan onder andere naar voren komen waar de metadata aan moet voldoen om te fungeren als basis voor de visualisatie.

Na het onderzoek en eerste set requirements wordt er begonnen met ontwerpen van de dataflowvisualisatiemodule. In het functioneel ontwerp worden de requirements omgezet naar functionaliteiten. Terwijl dit document wordt gemaakt zal er een iteratie plaatsvinden op de requirements. Hierbij worden de belangrijkste requirements doorgewerkt in het ontwerp.

Nadat de functionele zijde van het ontwerp voltooid is wordt de technische kant benaderd. Hierbij wordt de structuur van de omliggende systemen en andere technische eisen vastgesteld. Zodat de beschreven functionaliteiten kunnen worden behaald.

Na het vaststellen van de functionaliteiten en de technische structuur wordt de dataflowvisualisatie module geschreven. Ook de resultaten van het onderzoek worden verwerkt in de uiteindelijke module.

Het eindproduct moet voldoen aan de gestelde functionaliteiten en dus ook de requirements. Op basis van het requirementsrapport en ontwerp kan worden getest of de module voldoet. Hiermee wordt de validiteit van het project aangetoond.

Om de tweede genoemde doelstelling te halen van dit project moet er een afstudeerverslag worden geschreven. Dit verslag wordt gemaakt en voortdurend aangevuld met informatie over de verloop van het project. Dit document bevat in de bijlagen alle documentatie omtrent dit project. Ook is de opdrachtgever van deze doelstelling is dan ook de Haagse Hogeschool. Eventuele (delen) source code kan ook overlegd worden aan deze opdrachtgever.

De uiteindelijke mijlpalenlijst is dan ook als volgt:
- Plan van aanpak
    -Planning
- Requirementsrapport
- Onderzoeksrapport
    -Onderzoeksplan
- Functioneel ontwerp
- Technisch ontwerp
- Dataflowvisualisatie module
- Testrapport
- Afstudeerverslag
    -Voorgenoemde documentatie

Onderzoeksrapport

Data flow, hoe te laten zien en met wat?

Titel	Onderzoeksrapport
Project/Onderwerp	Data flow, hoe te laten zien en met wat?
Versie	2.0
Status	Internal Draft
Datum	30-mrt-2015
Bedrijf	Info Support B.V.

Historie

Versie	Status	Datum	Auteur	Verandering
1.0	Concept		Maarten Koene	Creatie
1.5	Verbeterslag, resultaten in een top down benadering		Maarten Koene	Hierzien van resultaten en toevoegen van informatie aan de resultaten voor top down benadering, herformulering van de conclusie
2.0	Herziening van de resultaten		Maarten Koene	Weglaten van de bestaande software en aanpassingen naar aanleiding van requirements

Inhoudsopgave

1. Inleiding 4

1.1 Aanleiding 4

1.2 Doelstelling 4

1.2.1 Probleemomschrijving 4

1.2.2 Scope 5

1.3 Theoretisch kader 5

1.4 Hoofdvraag 6

1.5 Deelvragen 6

1.6 Leeswijzer 6

2. Vooronderzoek 8

3. Onderzoeksontwerp 10

3.1 Interview 10

3.2 Populatie 10

3.3 Steekproef 11

3.4 Overzicht deel- en onderzoeksvragen 11

4. Resultaten 12

4.1 ISMetadata 12

4.2 Vormen van data flow visualisatie 13

4.2.1 Bron, bewerking, bestemming 14

4.2.2 Bron, bron/drop-off, bron/drop-off 17

4.3 Java Code library 18

4.3.1 Netbeans Visual 19

4.3.2 Jung 20

4.3.3 yWorks 22

5. Conclusie 24

5.1 Deelvragen 24

5.2 Hoofdvraag en advies 25

6. Bronnenlijst 27

1. Inleiding

1.1 Aanleiding

Business Intelligence systemen worden steeds meer ingezet. In dit soort systemen worden gegevens uit meerdere databronnen verzamelt. Door de gegevens uit meerdere bronnen te combineren kunnen er nieuwe inzichten ontstaan. Zo kan bijvoorbeeld de kosten van zieke werknemers per maand worden bekeken. De gecombineerde gegevens worden in een datawarehouse(DWH) opgeslagen. Een DWH is erop gericht om relatief snel resultaten te kunnen genereren, zoals in het voorbeeld.

Info Support ontwerpt en ontwikkelt deze Business Intelligence systemen voor haar klanten. Om het proces van het ontwerpen en ontwikkelen van een datawarehouse te versnellen is er een tool gemaakt binnen Info Support. De tool, ISMetadata, kan gedeeltelijk het datawarehouse genereren. De werking van deze tool wordt beschreven in hoofdstuk 4.1.

Het gebruik van de ISMetadata wordt gedaan door ontwerpers en ontwikkelaars. Deze geautomatiseerde manier van werken blijkt echter een nadeel te hebben. Als de datavaults en bronnen moeten worden aangepast, hebben ontwikkelaars en ontwerpers moeilijkheden met de oorsprong en eindstand van de data te visualiseren. De oorsprong van deze data is op tabel niveau te visualiseren door middel van gebruik van SSIS packages. Echter is het van belang voor de ontwerpers en ontwikkelaars om dit op attribuutniveau te visualiseren. Er is echter onduidelijkheid over de wijze waarop dit kan worden gedaan. Ook weet Info Support niet welke software of code library's het visualiseren op attribuut niveau kan ondersteunen.

Daarom is er een project opgezet om de metadata die wordt gegenereerd door ISMetadata te visualiseren. In dit onderzoek wordt onderzocht welke manieren van visualisatie toepasbaar zijn voor de metadata van ISMetadata. Ook wordt bestaande software bekeken die ondersteuning kan bieden voor automatisch genereren van een visualisatie op attribuutniveau. Bij onvoldoende functionaliteit van bestaande software wordt er onderzoek gedaan naar code library's die visualisatie op attribuut niveau mogelijk kunnen maken.

1.2 Doelstelling

1.2.1 Probleemomschrijving

De metadata, geproduceerd door ISMetadata, kan op dit moment niet grafisch worden benaderd. De ontwikkelaars en ontwerpers geven aan dat het aanpassen van de metadata een tijdrovende klus is.

In dit onderzoek wordt er gekeken naar de bestaande vormen van dataflowvisualisatie. Hierbij worden de bestaande software en code library's geanalyseerd. Dit zal gebeuren aan de hand van interviews met experts op het gebied van data(flow)visualisatie. Verder wordt er een literatuur analyse gedaan naar de bestaande varianten van dataflowvisualisatie.

Het onderzoek zal zich verder focussen op de wijze waarop de dataflow gevisualiseerd dient te worden. Hierbij zal de vorm van de grafische weergave centraal staan. In gesprekken met de opdrachtgever en het literatuur onderzoek zal een advies schetsen voor het gebruik tijdens het ontwerpen van de dataflowvisualisatietool.

Het onderzoek zal dus uitwijzen welke Java code library's er eventueel geschikt is voor het visualiseren van de metadata van datawarehouses geproduceerd door ISMetadata. Op deze manier kan er voor het project omtrent het maken van een dataflowvisualisatietool een advies worden uitgebracht.

1.2.2 Scope

Het onderzoek is opgezet voor Info Support. Dit onderzoek werkt ter ondersteuning van het project omtrent data flow visualisatie. Dit project wordt gericht op het in beeld brengen van dataflowmetadata die wordt opgeslagen staat in ISMetadata. Verder wordt er gewerkt binnen het Java platform in dit project. Daarom zal de scope van het onderzoek zich beperkt zijn tot:

- De metadata die wordt opgeslagen in de ISMetadata database.
- De onderzochte code library's moeten binnen het Java platform functioneren.

Verder zal de opdrachtgever een bindende rol hebben omtrent het selecteren manier waarop de metadata uiteindelijk zal worden gevisualiseerd. Dit onderzoek zal een adviserende rol spelen.

1.3 Theoretisch kader

Begrip	Betekenis
Code library	Routines, operaties en objecten voor een platform. In dit onderzoek specifiek voor Java
Data flow	Verplaatsen van data van een systeem (bijv. database) naar een andere bron. In dit onderzoek is de data flow, het verplaatsen van data van de bronsystemen naar het datawarehouse.
Datavault	Een databron gemodelleerd op basis van bronsystemen. In een datavault wordt geen data aangepast of verwijderd.
ERD	Entity Relation Diagram. Dit is een modelleer techniek waarmee de conceptuele structuur van een database wordt beschreven. ERD's kunnen ook worden gebruikt om bedrijfsprocessen te beschrijven.
ETL schema	Extract Transform Load (ETL). Vaak in SQL geschreven script om data uit een databron te halen, te veranderen en daarna in een andere databron op te slaan.
ISMetadata	Een tool van Info Support, gemaakt om op basis van bronsystemen en een bedrijfsmodel. Hieruit wordt een conceptueel datamodel te gegenereerd van een datavault. Dit model wordt opgeslagen in de metadata database in de control environment.
Java	Een programmeer platform, gemaakt om te werken met objecten (object oriëntatie)
Klasse	Een onderdeel van een klassendiagram. Deze beschrijft een onderdeel van het systeem en de bijbehorende attributen van de klasse.
Klassendiagram	Een statisch diagram, die de structuur van een systeem beschrijft (applicatie of database), door middel van klassen en attributen.

Begrip	Betekenis
Layout	Een systematische wijze van het tonen van data. Bijvoorbeeld door het aanbrengen van groeperingen of hiërarchieën.
Metadata	Data die andere data beschrijft.

1.4 Hoofdvraag

De hoofdvraag van dit onderzoek is als volgt:
Welke vorm van data flow visualisatie is het meest geschikt om de data flow informatie uit ISMetadata weer te geven en welke Java code library's kunnen dit ondersteunen?

1.5 Deelvragen

Vooronderzoek:

1. Wat is een dataflow?
2. Waar bestaat een dataflow uit?
3. Wat moet er van een dataflow worden gevisualiseerd?
    a. Welke eisen stelt dit aan de metadata?

Onderzoek verschillende manieren visualisatie:

1. Op welke manieren kan een dataflow worden gevisualiseerd?
    a. Wat zijn de voor- en nadelen van de verschillende manieren?
2. Welke visualisatie is het meest geschikt voor de metadata van ISMetadata?
    a. Sluit dit aan bij de wensen van Infosupport?

Onderzoek beschikbare Java code library's:

1. Welke code library's van Java zijn van toepassing op het visualiseren van data?

1.6 Leeswijzer

Het onderzoeksrapport begint met een beschrijving van de aanleiding, hoofd en deelvragen. Hierbij wordt de scope en theoretisch kader vastgesteld. In het tweede hoofdstuk wordt de benodigde achtergrond informatie over data flows beschreven. In hoofdstuk 3 wordt de structuur en werkwijze van het onderzoek beschreven. Hierbij wordt de populatie en onderzoeksmethode per deelvraag vastgesteld.

In hoofdstuk 4 worden de resultaten van de onderzoeksmethode getoond. Eerst wordt de werking van ISMetadata getoond. Dit concludeert het vooronderzoek. Waarbij aangetoond wordt dat de metadata van ISMetadata de juiste informatie bevat voor een gewenste visualisatie. Gevolgd door de verschillende methoden om data flows te visualiseren. Dit beantwoordt de volgende vragen:

1. Op welke manieren kan een dataflow worden gevisualiseerd?
    a. Wat zijn de voor- en nadelen van de verschillende manieren?

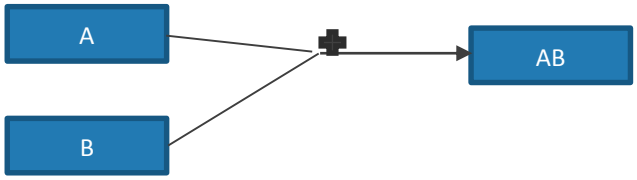
2. Welke visualisatie is het meest geschikt voor de gegenereerde metadata van ISMetadata?
    a. Sluit dit aan bij de wensen van Infosupport?

Het hoofdstuk eindigt met de analyse van verschillende Java library's. Dit beantwoordt de vraag: Welke code library's van Java zijn van toepassing op het visualiseren van data?

In hoofdstuk 5 worden de deel- en hoofdvragen beantwoord. Hierbij wordt een advies gegeven omtrent het project van data flow visualisatie van Info Support. Het document sluit af met de gebruikte bronnen voor dit onderzoek.

2. Vooronderzoek

Een dataflow is een gegevensstroom. De gegevensstroom kan plaatsvinden tussen twee of meer systemen. Ook kan een gegevens stroom binnen het systeem gehouden worden. Denk hierbij aan de formules die gebruikt kunnen worden in excel om de waarde van een cel te laten afhangen van een of meerdere cellen. Bij een dataflow worden de waarden van een of meerdere velden (cellen in excel) afhankelijk gemaakt van andere velden of cellen.



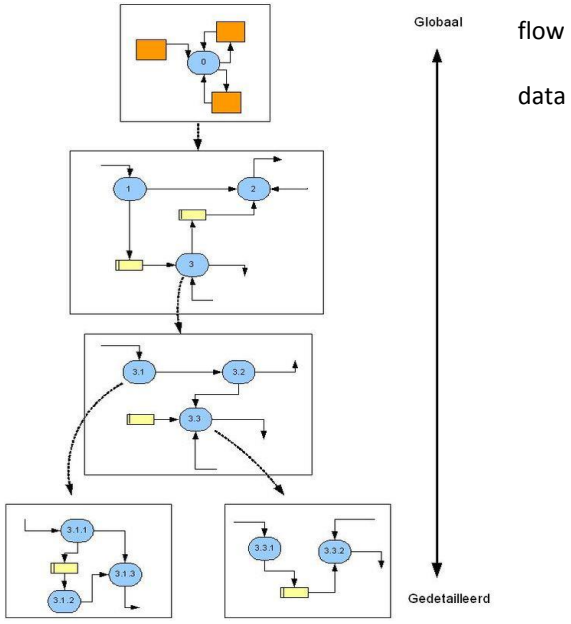
Figuur 1 *Voorbeeld van een bewerking in een dataflow*

Bovenstaand figuur illustreert het concept van een dataflow. Hierbij worden de waarden van veld A en veld B opgeteld. Als de waarde van veld A of B veranderd wordt de waarde van veld AB ook. Veld AB is dus afhankelijk gemaakt van zowel veld A als B. De gegevens van de velden A en B stromen geaggregeerd door naar veld AB.
Het geïllustreerde voorbeeld is een versimpeling van een dataflow. Om een grafische representatie te geven van de dataflow tussen de verschillende systemen is er een data flow diagram(DFD). Een data flow diagram wordt handmatig gemaakt. Dit wordt gedaan om een nieuw informatiesysteem of een bestaand informatiesysteem te beschrijven. Er zijn verschillende vormen voor een DFD. Het data flow diagram geeft een overzicht van de data flows en de tussenstations. In deze opdracht wordt dieper ingegaan op een wijze van data flow visualisatie en de bewerkingen op de data.
Om een beeld te krijgen van een bestaande wijze van visualisatie wordt er in dit hoofdstuk ingegaan op een data diagram. In een data flow diagram wordt onderscheid gemaakt tussen een aantal verschillende onderdelen. Een flow diagram bestaat uit de volgende onderdelen:

- Data store
- Terminator
- Data flow
- Process

De datastore is een opslagplaats van data. Hieronder worden allerlei vormen verstaan, zoals databases, excelsheet of boek. (in het figuur hiernaast gele rechthoeken)

Een terminator is een begin of een eindstation. Hierbij worden zowel de source systemen bedoelt als de eindbestemmingen (bijvoorbeeld een datawarehouse). De richting van de data flow geeft aan of het een begin of eindstation betreft.(in het figuur met oranje vierkanten)
Een process is een bewerking op de data. Hierbij worden



Figuur 2, *Voorbeeld van een dataflow*

de gegeven getransformeerd of verwerkt. Dit kunnen allerlei soorten bewerkingen zijn, zoals in het voorbeeld twee velden optellen of vermenigvuldigen. Ook het veranderen van datatype kan vallen onder een process. (in het bovenstaande figuur de blauwe ovalen)

Een data flow is een gegevensstroom. Dit kan gebeuren tussen data stores en terminators. Deze kan dan worden onderbroken door een of meerdere processen.(in het bovenstaande figuur weergegeven door pijlen)

Het data flow diagram laat gegevens stromen zien op architectuur niveau. Er worden vaak verschillende systemen gekoppeld en de stromen van gegevens tussen deze stromen visueel gemaakt. Echter kan er in een DFD ook op een systeem worden ingezoomd. Hierbij wordt de data tussen de verschillende elementen van een systeem getoond. In dit onderzoek wordt visualisatie op attribuutniveau van een database verwacht. Bestaande software heeft zich gericht op het maken van boven beschreven DFD. Dit is, als er is ingezoomd, op systeem niveau weergegeven. Daarom zal bestaande software geen mogelijkheid geven tot het automatisch genereren van een visualisatie op tabel niveau. In de resultaten zal dus worden beschreven welke visualisatie vormen toepasbaar zijn voor visualisatie op attribuutniveau.

Uit interviews met de heer van Bilsen is duidelijk geworden dat er een data flow visualisatie moet worden gemaakt op de metadata informatie van ISMetadata. Hierbij zijn er een aantal belangrijke eisen aan de visualisatie. In het hoofdstuk over ISMetadata zal worden ingegaan op de inhoud van de metadata van ISMetadata. Voor een gewenste visualisatie moeten de volgende onderdelen in ieder geval kunnen worden gevisualiseerd:

- Attribuutnaam in het bronsysteem
- Bewerkingen (Mappings) op de attributen tussen bronsystemen en datawarehouse
- Attribuutnaam in het datawarehouse

3. Onderzoeksontwerp

In dit hoofdstuk wordt het onderzoek beschreven. Uit dit hoofdstuk wordt duidelijk op welke wijze het Interview vorm heeft gekregen. Verder wordt de populatie en haar eigenschappen beschreven. Een korte beschrijving van de steekproef volgt hierop. Het hoofdstuk zal eindigen door aan te geven met welke onderzoeksmethode de deelvragen worden beantwoord.

3.1 Interview

Om duidelijkheid te scheppen over beschikbare Java code library's worden er interviews gehouden met een expert van Info Support. Hij kan uit ervaring en opgedane kennis informatie verschaffen over bestaande oplossingen voor data flow visualisatie. Met Orhan worden interviews gehouden totdat alle informatie die benodigd is om een eenduidig antwoord te geven op de deelvragen is bereikt. Bij dit interview zal de volgende topic lijst worden gebruikt:

- Dataflow en onderdelen daarvan
- Code library's van Java

Een andere expert, die tijdens dit onderzoek geraadpleegd zal worden, is Thomas van Bilsen. Hij is de opdrachtgever binnen Info Support. Verder werkt hij in de Data Solutions tak van Info Support. Hij zal de ontwikkelaars en ontwerpers van Data Solutions vertegenwoordigen. Dit omdat er binnen Info Support een centraal aanspreekpunt is aangewezen. Hierdoor kunnen er inhoudelijke vragen worden beantwoord over de wensen bij Info Support. Tijdens deze interviews worden de volgende topics besproken:

- De delen van de metadata die gevisualiseerd dienen te worden
- Dataflow en de verschillende onderdelen binnen een dataflow van ISMetadata

3.2 Populatie

De populatie is overeenstemmend met het voorgaande document, het plan van aanpak. Hierin wordt beschreven dat de ontwikkelaars en ontwerpers de data flow visualisatietool zullen gebruiken. Dit onderzoek legt de basis voor het ontwerp van de data flow visualisatietool. Waardoor het van belang is dat de meningen en kennis van de werking van ISMetadata worden verwerkt in dit onderzoek. Op deze manier kan er een passende visualisatie worden gecreëerd.

De populatie heeft dus de volgende kenmerken:
- Ontwikkelaar of ontwerper bij Info Support zijn
- Inhoudelijke technische kennis van ISMetadata hebben

3.3 Steekproef

Het onderzoek naar data flow visualisatie heeft twee delen. Het eerste deel is gericht op verschillende manieren om data flows te visualiseren. Het tweede deel richt zich op de technische kant van data flow visualisatie. (beschikbare code library's). De beschreven populatie in hoofdstuk 3.3 is een kleine groep. Uit de populatie zijn twee centrale aanspreekpunten aangewezen. Er zal daarom gebruik gemaakt worden van een selecte steekproef met twee experts.

De twee experts die geïnterviewd zullen worden, zijn Thomas van Bilsen en Orhan Uyan. Hierbij zal Thomas van Bilsen als expert dienen over de ISMetadata en de bijbehorende data flows. Orhan Uyan zal benaderd worden voor technisch inhoudelijke kennis over Java code library's.

3.4 Overzicht deel- en onderzoeksvragen

Deelvraag	Onderzoeksvraag	Kwalitatief/ kwantitief	Onderzoekssoort
Wat is een data flow?		Kwalitatief	Literatuuronderzoek
Waar bestaat een dataflow uit?		Kwalitatief	Literatuuronderzoek
Wat moet er van een dataflow worden gevisualiseerd?		Kwalitatief	Literatuuronderzoek/interview
	Welke eisen stelt dit aan de metadata?	Kwalitatief	Interview
Op welke manieren kan een dataflow worden gevisualiseerd?		Kwalitatief	Literatuuronderzoek/interview
	Wat zijn de voor- en nadelen van de verschillende manieren?	Kwalitatief	Literatuuronderzoek
Welke visualisatie is het meest geschikt voor de gegenereerde metadata van ISMetadata?		Kwalitatief	Literatuuronderzoek/interview
	Sluit dit aan bij de wensen van Infosupport?	Kwalitatief	Interview
Welke code library's van Java zijn van toepassing op het visualiseren van data?		Kwalitatief	Literatuuronderzoek/interview

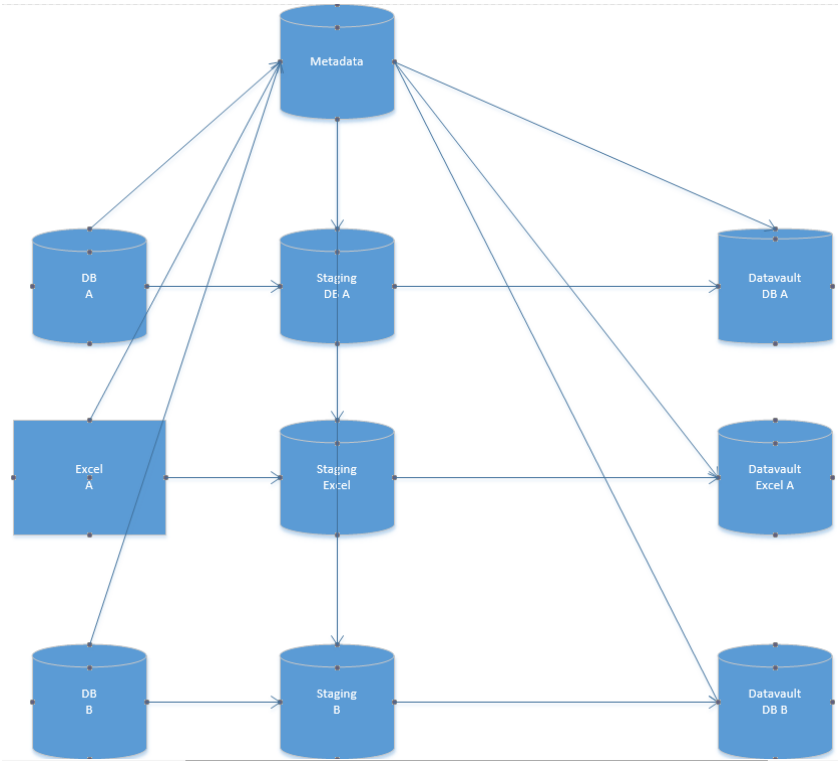
4. Resultaten

In de resultaten van dit onderzoek wordt begonnen met de werking van ISMetadata gevolgd door de verschillende vormen van data flow visualisatie. De resultaten worden afgesloten met een analyse van Java code library's, die gericht zijn op data flow visualisatie.

4.1 ISMetadata

Bij Info Support is er een geautomatiseerde manier ontwikkeld om van verschillende bron systemen datavaults te creëren. Door dit te doen kunnen Business Intelligence systemen gedeeltelijk geautomatiseerd worden opgezet. In dit hoofdstuk wordt de werkwijze van Info Support in het proces van het ontwikkelen van een Business Intelligence systemen toegelicht met betrekking tot het gebruik van ISMetadata.

Om te illustreren op welke wijze dit in zijn werk gaat is de onderstaande afbeelding opgesteld. De afbeelding is een schets van de werkelijkheid, dit betekent dat het aantal source systemen kan verschillen van de werkelijkheid. Het is mogelijk om verschillende soorten bronnen aan te koppelen. Zo kan er bijvoorbeeld een excel sheet of database worden gekoppeld. De onderstaande schets geeft een beeld van de essentiële functies van ISMetadata bij het creëren van een datavault.



Figuur 3, *Schets werking van de ISMetadata*

Bij het ontwerpen van een datavault worden er een aantal stappen ondernomen. De eerste stap is het identificeren van de bronsystemen. In bovenstaand figuur zijn dit “DB A”, “Excel A” en “DB B”. In werkelijkheid kunnen dit meer systemen zijn, ook kunnen er CSV files of andere vormen van dataopslag worden gebruikt als bronsysteem. Na de identificatie worden er staging area’s gemaakt voor de bronsystemen. Per bronsysteem wordt er door ISMetadata een staging database gemaakt.

De tweede stap is het kopiëren van de gehele data en structuur naar de staging area. Bij het kopiëren wordt de inhoud gekopieerd naar de bijbehorende staging database. Als het Business intelligence systeem draaiende is, wordt er periodiek(1x per dag) de data uit de bronnen aangevuld.

In de derde stap wordt door ISMetadata, in het figuur weergegeven door “Metadata”, een analyse gedaan. Deze analyse wordt gedaan op het bronsysteem. Na de analyse van de structuur van de bronsystemen wordt er door het toepassen van standaard regels datavaults gemodelleerd. Per bronsysteem ontstaat er een model voor de bijbehorende datavault. Deze modellen worden opgeslagen in ISMetadata.

De laatste stap is het genereren van de code en dus de datavaults. Dit wordt gedaan op basis van de modellen die zijn gegenereerd door ISMetadata. Hierna kan de data in de staging area’s worden bewerkt en worden verplaatst naar de bijbehorende datavault.

Na deze stap wordt er door ontwerpers en ontwikkelaars een datawarehouse gemodelleerd. Dit gebeurd dus niet door het gebruik van ISMetadata tool. Nadat de datawarehouse gemodelleerd is kan deze worden ingeladen in ISMetadata. Hierbij wordt de structuur van de datawarehouse in de ISMetadata database opgeslagen. Door middel van een module binnen ISMetadata kan er met de hand de mapping worden aangemaakt. Dit zodat de attributen uit de datavault kunnen worden gekoppeld aan de bijbehorende attributen in het datawarehouse. Deze mappings worden hierna opgeslagen in de ISMetadata database.

De metadata opgeslagen in de ISMetadata database. De verschillende stappen zorgen voor verschillende soorten data in de metadata In het kort bevat de metadata de volgende delen:

- Beschrijving van de structuur van bronsystemen en staging area’s (tabellen en kolommen)
- Model beschrijvingen van de datavaults (tabellen, kolommen en relaties)
- Bewerkingen (incl. Mappings) tussen de bronsystemen en de datavault (op attribuutniveau)
- Bewerkingen (incl. Mappings) tussen de datavaults en het datawarehouse (op attribuutniveau)

4.2 Vormen van data flow visualisatie

Het eerste model om data flows te visualiseren is besproken in de achtergrond informatie. Dit model focust zich op het in kaart brengen van data tijdens een business proces. In dit hoofdstuk worden nog twee andere modellen van data flow visualisatie in kaart gebracht. Allereest wordt aangegeven welke elementen er tijdens een data flow visualisatie naar voren moeten komen. Hierna zullen de twee data flow modellen worden behandelt en toegelicht met voorbeelden.

Uit de interviews met de opdrachtgever is naar voren gekomen dat bij een visualisatie van een data flow in ieder geval de volgende elementen moeten worden benaderd.

- Attribuutnaam in het bronsysteem
- Bewerkingen (Mappings) op de attributen tussen bronsystemen en datawarehouse

- Attribuutnaam in het datawarehouse

4.2.1 Bron, bewerking, bestemming

De eerste methode van data flow visualisatie is het bron-bewerking-bestemming model. In dit model wordt er gebruik gemaakt van drie fasen van data. De data bevind zich in de bron, bewerking of bestemmingsfase. Het model is veelgebruikt om data te tonen die deze fasen doormaakt.

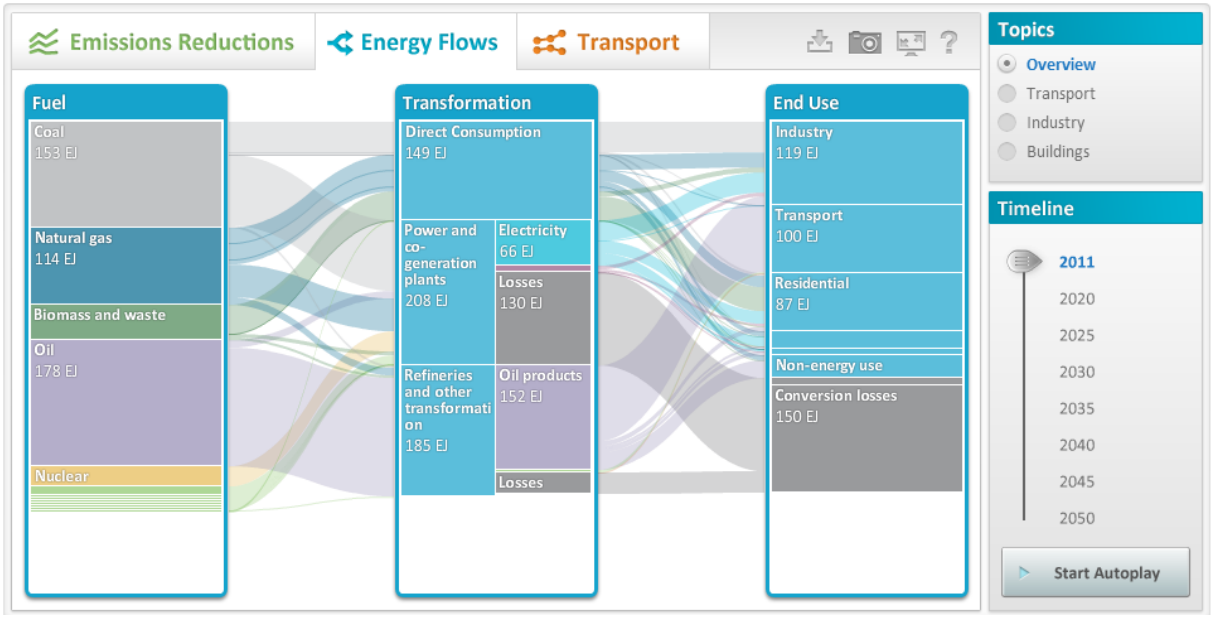
De data wordt vanuit de bron gehaald en naar de bewerking gebracht. In de bewerkingsfase worden er een of meerdere aanpassingen gedaan (zoals omzetten in een andere eenheid, optellen van meerdere dataregels). Nadat de bewerking voltooid is wordt de data naar de bestemmingsfase gebracht. De bestemmingsfase is de eindfase waarin de data verkeerd. Hier is de data naar wens aangepast en kan deze worden gebruikt voor het doeleinde van deze data. Dit model heeft de volgende voordelen:

- Overzichtelijk
- Duidelijke scheiding tussen de verschillende fasen
- Er is een begin en een eind aan de data flow

Verder zijn er een aantal nadelen te benoemen aan dit model:

- Het gebruik van meerdere bronnen, bewerkingen en bestemmingen kan voor een omvangrijke grafische weergave zorgen
- Onderscheid tussen verschillende bewerkingsfase moeten zelf visueel worden gemaakt (het verschil tussen een optelling of een eenheidsomzetting zichtbaar maken)
- Het model is gericht op visualiseren van data, die geen betrekking heeft tot database data flow (zie voorbeelden hieronder)

Onderstaand zijn twee voorbeelden van het bron-bewerking-bestemming model. De voorbeelden zullen worden uitgelegd aan de hand van een afbeelding. Hierbij wordt aangegeven op welke wijze het model is toegepast in het voorbeeld van visualisingdata.com.

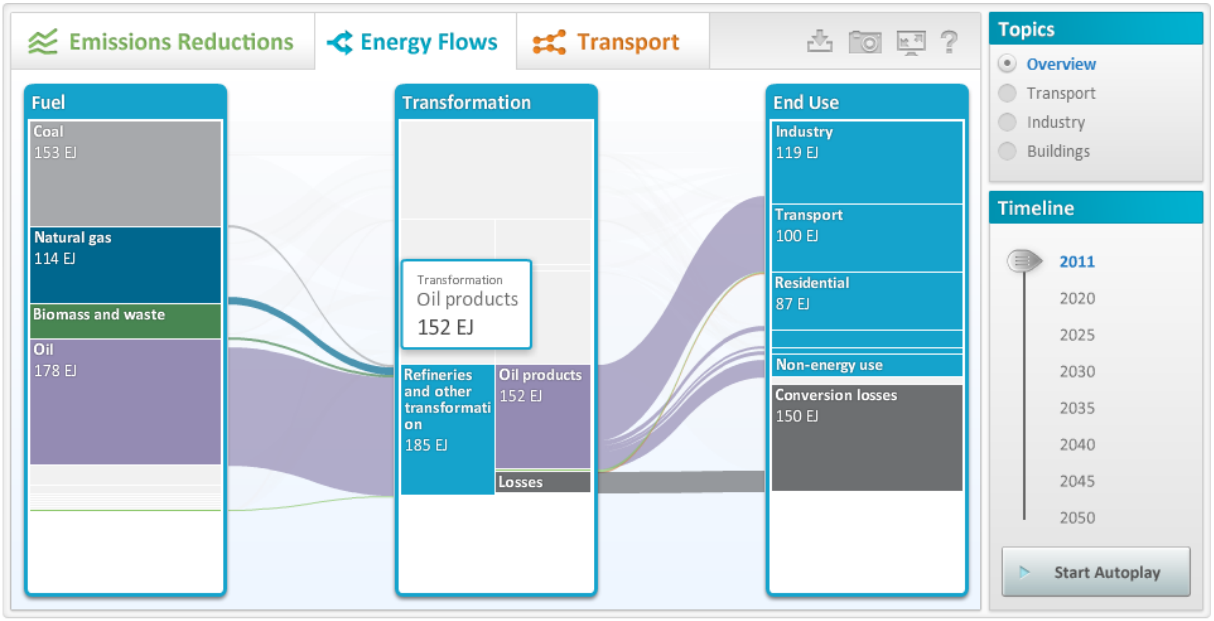


Figuur 4, *Flow chart van brandstof verbruik*

In bovenstaand figuur wordt het gebruik van brandstof in drie fasen laten zien. De eerste fase is de bron waar de brandstof uit wordt gewonnen. De grootte van het blok geeft de hoeveelheid schematisch weer. Hierna wordt de brandstof naar de tweede fase gestuurd. In het tweede blok wordt de brandstof getransformeerd, dit is de bewerkingsfase uit het model. Na de transformatie wordt de brandstof gebruikt in het eindproduct. Dit is de bestemmingsfase van het voorgaand beschreven model.

Tussen de blokken wordt de hoeveelheid verdeelt in verschillende stromen. De dikte van de stroom geeft de hoeveelheid van het totaal weer. In het transformatie blok wordt uit de verschillende bronnen een tussenproduct getransformeerd.

In het onderstaande figuur is te zien dat coal, natural gas, biomass and waste en oil worden in de transformatie (gedeeltelijk) omgezet naar olie producten en een deel verlies. Na de transformatie worden de tussenproducten gebruikt op verschillende plekken. In het figuur zijn dat industry, transport enzovoort. Ook deze zijn verbonden door een golf en de dikte van deze golf geeft de hoeveelheid schematisch weer.



Figuur 5, *Flow chart van brandstof verbruik met een selectie*

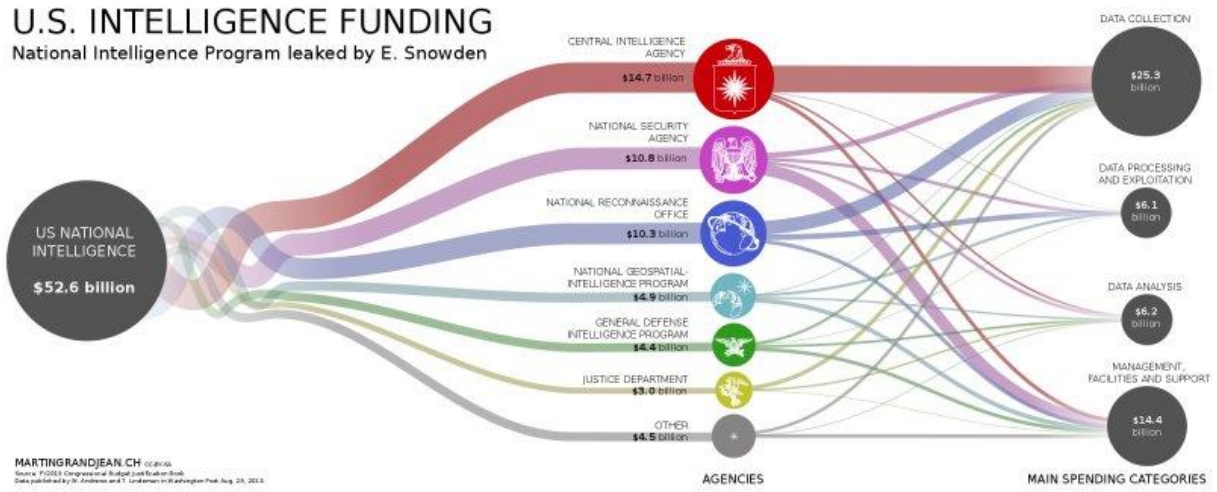
Door het gebruik van de bron-bewerking-bestemmingsmodel is er een data flow visualisatie gemaakt van de “data” brandstof. Het bovenstaande voorbeeld heeft een aantal nadelen opgeheven van het model. Het voorbeeld heeft namelijk:

- Onderscheidbare bewerkingsfase(n)
- Mate van samenhang tussen de fasen aangegeven door de dikte van de stromen en blokken

Het bovenstaande voorbeeld heeft de bewerking in de bewerkingsfase een naam gegeven. In het bovenstaande voorbeeld wordt de bewerking “refineries an other transformation” genoemd binnen de bewerkingsfase. Op deze manier kan er tussen de verschillende soorten bewerkingen onderscheidt worden gemaakt. Het nadeel van deze visualisatie is het overzicht. Er zijn een groot aantal stromen dat door elkaar loopt, hierdoor wordt het bemoeilijkt om in een oogopslag te zien waar de brandstof vandaan komt en uiteindelijk gebruikt wordt.

Een andere toepassing van dit model is het voorbeeld van businessinsider.com. Zij hebben de uitgave van de Verenigde Staten betreffende data verzamelen in beeld gebracht. Het onderstaande figuur is het resultaat van deze visualisatie.

Er wordt aan de linkerkant begonnen met de totale uitgave aan “intelligence funding”, in het model is dit de bron fase. De grootte van de cirkels en de dikte van iedere lijn geeft grafisch de hoeveelheid weer. De linker cirkel verdeelt het geld over de verschillende overheidsonderdelen. Deze overheidsonderdelen spenderen kunnen worden gezien als de bewerkingsfase. Er wordt namelijk gekozen waar het gekregen bedrag over wordt verdeelt per overheidsonderdeel. Het geld stroomt na de overheidsonderdelen naar de bestemmingsfase. Waarna het doel van de bestemming wordt behaalt door het vergaarde geld (data).



Figuur 6, *Flow chart van uitgaven per onderdeel overheid VS*

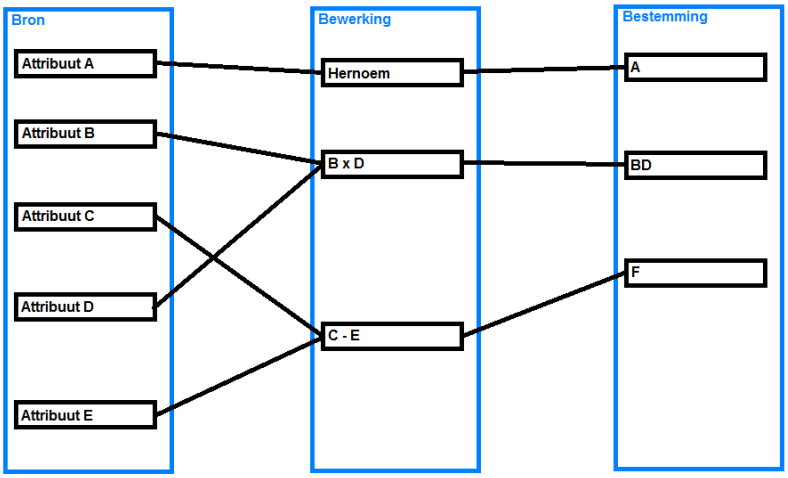
Een aantal voordelen zijn er te zien aan deze visualisatie vorm van de bron-bewerking-bestemming te zien.

- Er is een duidelijk start en eindpunt
- Gebruik van verschillende kleuren voor de stromen maakt de flow overzichtelijk
- Gebruik van meerdere bewerkingsfasen toegepast
- Meerdere bestemmingsfasen toegepast

Een aantal nadelen van deze toepassing zijn:

- Onduidelijkheid in de exacte bewerking bij de “agencies”. Het wordt uit het figuur niet duidelijk waarom uitgaveposten zijn gekozen
- Bij meerdere bronfasen of bewerkinsfasen wordt deze grafiek onoverzichtelijk door beperkt kleuren onderscheid (oranje lijkt immers op rood enzovoort)

Het bron-bewerking-bestemming model kan worden toegepast op het visualiseren van metadata. Hierbij worden de bron attributen geplaatst in de bron fase. Door middel van pijlen worden de bron attributen uit de bron fase gekoppeld aan de bewerkingen in de bewerkingsfase. In de bewerkingsfase staat de aanpassing op de data beschreven. Door middel van pijlen wordt de bewerking gekoppeld aan het bijbehorende attribuut in de bestemmingsfase. Zie het onderstaande voorbeeld van dit model met attributen.



Figuur 7, *Bron-bewerking-bestemming met attributen database*

4.2.2 Bron, bron/drop-off, bron/drop-off

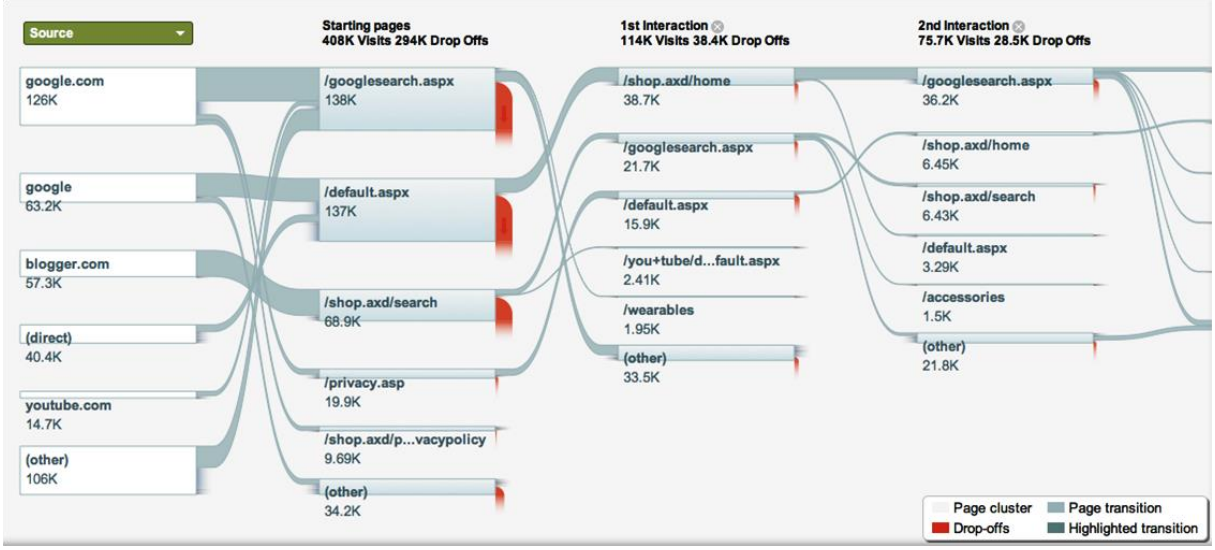
De tweede methode om data flow te visualiseren is het bron dorp-off model. In dit model wordt uitgegaan van een of meerdere bronnen. Vanuit die bronnen wordt de data doorgegeven aan de tweede bron. In de opvolgende bron wordt er gebruik gemaakt van de data. De data wordt na de tweede bron doorgegeven aan de volgende bron. Ook kan er een gedeelte van de data niet worden doorgestuurd naar de volgende bron, dit wordt de drop-off genoemd. Aan dit model zijn de volgende voordelen te zien:

- Mogelijkheid tot het tonen van data die niet verder wordt verplaatst
- Stroom van bron tot bron in een oogopslag te zien
- Complexe data stromen met verschillende bronnen zijn te visualiseren

Dit model heeft ook een aantal nadelen:

- Er is geen bestemming fase, het onderscheid tussen bron en datawarehouse kan hierdoor vervagen (behalve door naamgeving van de bronnen toe te passen kan dit worden verholpen
- Bewerkingen op data zijn niet meegenomen, tenzij deze worden aangegeven als bron

In Google Analytics wordt gebruik gemaakt van het bron-drop-off model. In Google Analytics kunnen de bezoekers en hun pad over website grafisch worden weergegeven. In het onderstaande figuur is een voorbeeld van een data flow gemaakt met Google Analytics. Deze zal worden besproken aan de hand van het bron-drop-off model.



Figuur 8, *Bron-drop-off model in Google Analytics*

Gebruikers komen via een bron op een website. Aan de linker zijde staan de mogelijke verschillende bronnen. De gebruikers stromen dan door naar de volgende bronpagina, dit is de tweede bron uit het model. Na de tweede bron, internetpagina van een website, wordt er doorgeklikt naar een volgende pagina. Deze gebruikers stromen door naar een derde bron. Het is echter ook mogelijk om niet door te gaan op de website. Dit wordt weergegeven in een drop-off (de rode stroom uit een bron). De gebruikers die door zijn gestuurd naar de derde bron kunnen daar weer door worden gestuurd naar de vierde bron of stoppen en in de drop-off terecht komen enzovoort.

Het bovenstaande figuur heeft gebruik gemaakt van het bron-drop-off model. Uit dit figuur komen een aantal voordelen naar voren:

- Grote van de blokken en stromen geven hoeveelheid weer
- Er kunnen een groot aantal bronnen worden weergegeven, hierdoor is een complexe datastroom overzichtelijk weergegeven
- Gebruikers(data), die niet door navigeren, vallen af
- Detail niveau dunt zich uiteindelijk uit tot minimaal 1 gebruiker

Echter blijkt uit het bovenstaande figuur ook een aantal nadelen naar boven te komen. Deze nadelen zijn:

- Geen eind duidelijk eindpunt(bestemming) van de data flow zichtbaar, de flow blijft doorgaan totdat gebruikers allemaal in de drop-off zijn gekomen. Hierdoor kan de relevantie van de data flow worden verminderd. De interesse in de flow van een enkele gebruiker kan buiten de scope vallen
- De redenen van een drop-off zijn onduidelijk uit het figuur
- Het figuur en model geven geen ruimte tot eventuele bewerkingen op de data

4.3 Java Code library

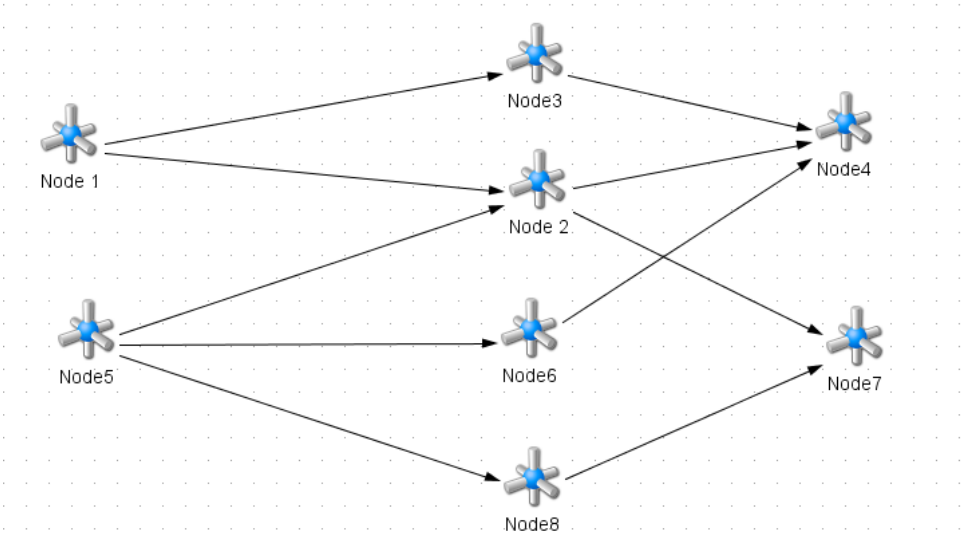
Er is een aantal Java library's die niet worden beschreven in dit hoofdstuk. De Java JGraphT library is geanalyseerd voor dit onderzoek. De reden dat deze library niet is meegenomen in de resultaten is het gebrek aan een stabiele release (alleen bèta versies beschikbaar). Verder is deze library voornamelijk gericht op het creeren van grafieken. Terwijl tijdens dit onderzoek wordt gefocussed op data flow visualisatie.

De Big Faceles Graph library is ook onder de loep genomen. De reden dat deze library niet is uitgewerkt is de 3D engine. Deze library is gericht op het maken van grote 3D grafieken. In dit onderzoek wordt gezocht naar een 2D data flow visualisatie. De meeste van de functionaliteiten van de BFG library zijn dan ook voor de 3D engine. Door de onwaarschijnlijkheid van het gebruik van de 3D engine is deze library niet in de resultaten te vinden.

De resultaten in dit hoofdstuk zijn library's die gericht zijn op het genereren van een visualisatie van de metadata. Hierbij worden de code libary's gekoppeld aan de eerder beschreven data flow visualisatie vormen.

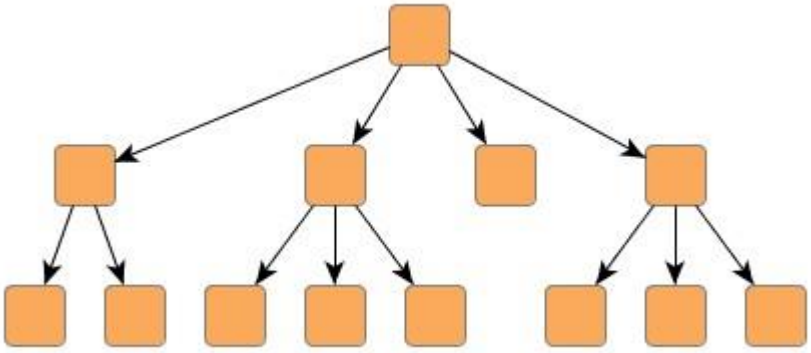
4.3.1 Netbeans Visual

De visualisatie van data flows kan, zoals blijkt uit vorige hoofdstukken, op verschillende manieren worden gedaan. Netbeans Visual is het mogelijk om bron-bewerking-bestemming te produceren. Deze library maakt gebruik van componenten van JavaFx. Door het gebruik van deze library wordt er een widget component toegevoegd. Dit element heeft de mogelijkheid om andere objecten te bevatten. Onderstaand figuur toont een voorbeeld van de mogelijkheden van Netbeans Visual.



Figuur 10, *Netbeans visual voorbeeld*

In het voorbeeld wordt gebruik gemaakt van de widgets uit de Netbeans Visual library. In dit voorbeeld zijn de Node de widgets. Met de hulp van eigenschappen van de widgets kunnen deze met pijlen worden verbonden. Tevens kan een widget informatie bevatten over de inhoud van de node. Hierin zou kunnen staan hoe een attribuut heet, uit welke bron deze komt en wat voor datatype het is. De nodes in het midden zouden bewerkingen kunnen zijn. De informatie die deze dan bezit is wat de bewerking inhoud en welke attributen er in gebruikt worden. De nodes aan de rechterkant kan dan de bestemming zijn. Hier kan de naam van de datavault in staan en de naam van het attribuut in de datavault. Op deze manier kan het bron-bewerking-bestemming model worden toegepast in Netbeans Visual.



Figuur 11, *Netbeans visual tree layout*

Netbeans Visual heeft de volgende voordelen:

- Gebruik maken van JavaFX
- Widget objecten met dynamische eigenschappen, zoals children objecten, listeners en pijlen
- Exporteer mogelijkheden naar Jpeg of PDF
- Animaties toevoegen aan diagrammen

Netbeans Visual heeft de volgende nadelen:

- Beperkt aantal layout mogelijkheden, grid en decision tree (layout is de wijze waarop de elementen worden gestructureerd bij de weergave)
- Geen directe database integratie

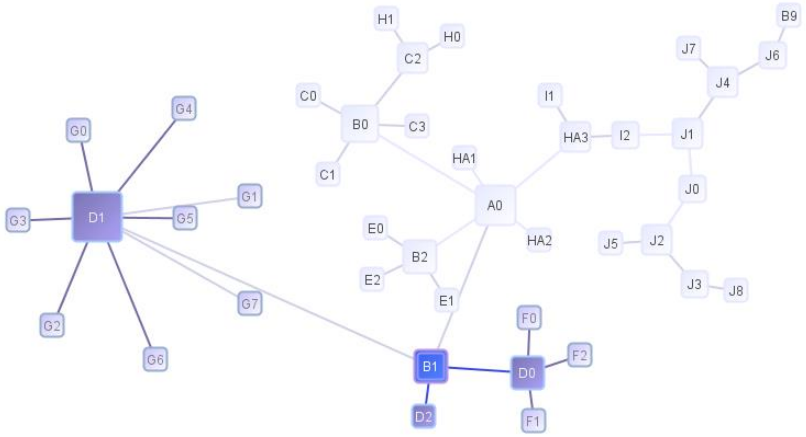
4.3.2 Jung

Jung is een Java library, die gericht is op het visualiseren van complexe datasets. In deze library worden handvaten geboden om met een dataset met verschillende soorten grafieken te benaderen. Met Jung kan op relationele data analyse worden uitgevoerd. Uit de documentatie van Jung blijkt dat deze library gebruikt kan worden om datamining uit te voeren op relationele data. De data kan worden geclusterd op basis van het bijgeleverde algoritmen. Hiermee kan het bron-bewerking-bestemming worden geproduceerd.

Er wordt gebruik gemaakt van de JavaFx componenten. Hierdoor worden de grafische componenten beschikbaar binnen de applicaties met Jung. Jung is gemaakt om statische grafieken te tonen. Er kan dynamisch een grafiek worden opgebouwd. Echter kan deze grafiek niet worden aangepast na het plotten.

Er zijn een groot aantal layout algoritmes toegevoegd aan de Jung Library. Hieronder een complete lijst van de beschikbare layouts binnen Jung:

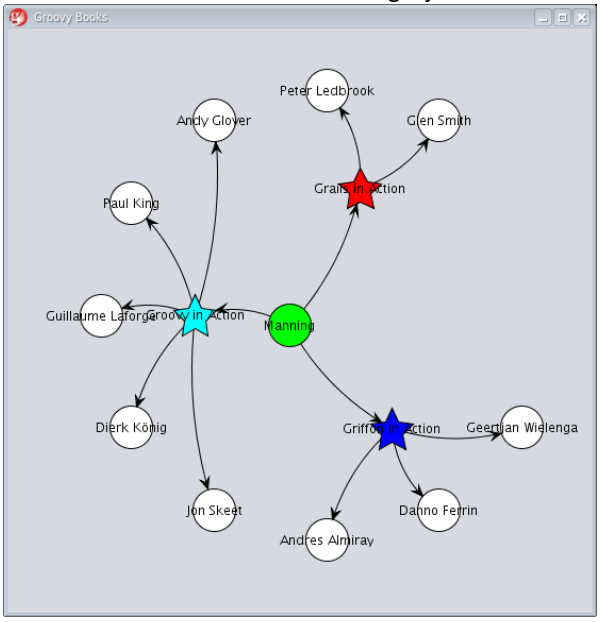
- KKlayout
- TreeLayout
- BalloonLayout
- RadialLayout
- CircleLayout
- FRLayout
- ISOMLayout
- SpringLayout
- DAGLayout
- XLayout



Figuur 12, *Voorbeelden van data flow met Jung en Netbeans Visual*

Bovenstaand figuur is een data flow visualisatie gemaakt met Jung en Netbeans visual. Hierbij zijn de widgets gebruikt en de layout mogelijkheden van Jung. Door te klikken op een widget kan worden gezien waar de data heen stroomt. Hierbij kan elke widget een bron, bewerking of bestemming zijn. Zo wordt hier gebruik gemaakt van het bron-bewerking-bestemming model op dezelfde wijze als bij vorige voorbeeld van Netbeans visual.

Onderstaand is een voorbeeld van een applicatie gemaakt met de hulp van Jung. De groene cirkel in het middel geeft de bron weer van de data. De gekleurde sterren kunnen een bewerking aangeven op de data. Waarbij de witte cirkels aan de rand de bestemming weergeeft. Op deze manier heeft deze applicatie een bron-bewerking-bestemmingsmodel ingevoegd. Het nadeel van enkel Jung gebruiken is het gebrek aan interactie. Het diagram is statisch. Dit betekent dat verschillende fasen (bron, bewerking, bestemming) geen extra informatie kan bevatten in de cirkels of sterren. Dit wat wel mogelijk was met Netbeans Visual.



Figuur 13, *Voorbeelden van data flow met Jung*

De voordelen van Jung zijn als volgt:

- Grote diversiteit aan layouts beschikbaar
- Ontworpen voor complexe relationele data

Het nadeel van Jung zijn als volgt:

- Beperkte grafische elementen (alleen basis vormen zoals cirkels, vierkanten sterren)

4.3.3 yWorks

yWorks is een betaalde library voor Java en Wpf, de licentie kosten zijn rond de €10.000 per jaar. De yFiles die bij dit pakket horen hebben de volgende functionaliteiten:

- Netwerk analyse en visualisatie
- Business proces modeling
- Data mining functions (analyse van log files)
- Database management en modellering
- Sociale netwerk plug-ins
- UML diagrammen genereren en aanpassen
- Flow chart generen en aanpassen
- 3D visualisaties
- Visueel programmeren

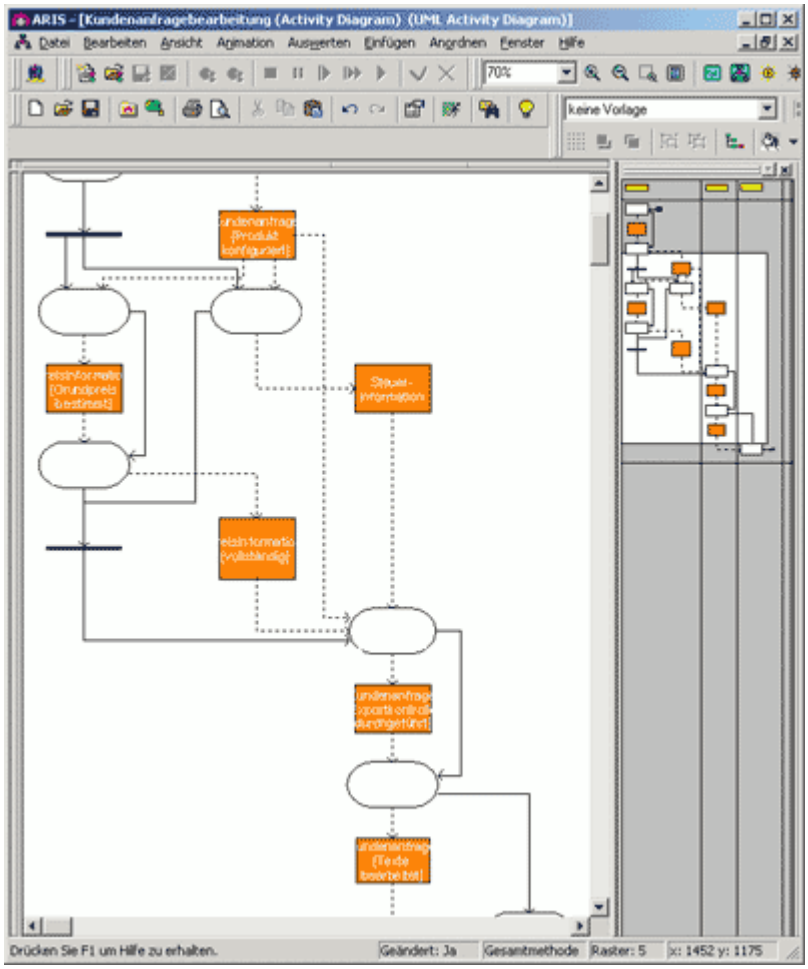
De eerder genoemde functionaliteiten worden niet uitgelegd. Dit omdat het gebruik van deze functionaliteiten betaald is. Wel zijn er op de website een aantal voorbeelden te vinden van mogelijke data flow visualisaties. Deze zijn gebaseerd op het data flow diagram model, gericht op processen. Ook lijkt het mogelijk om een bron-bewerking-bestemming model te creëren. In de onderstaande voorbeelden wordt dit uitgebeeld.

In het kort zijn de voordelen van yWorks:

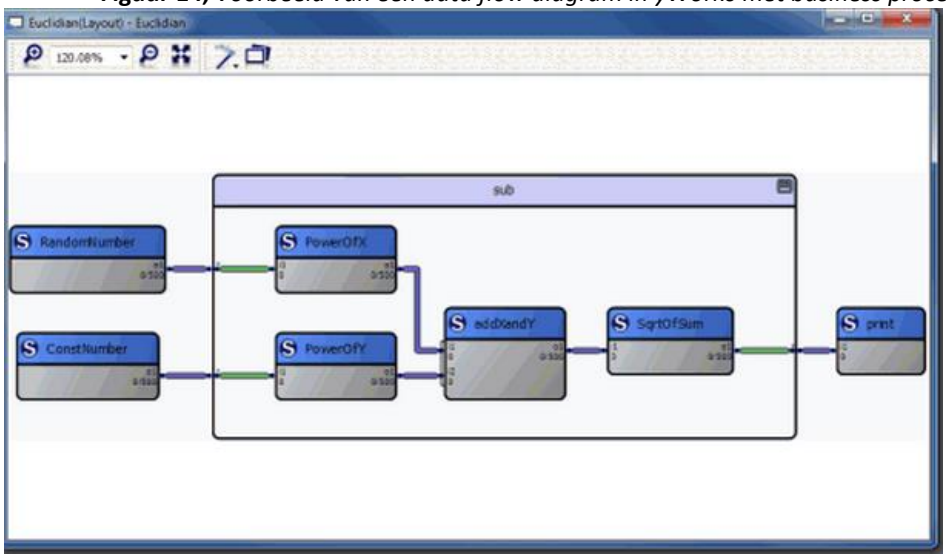
- Visueel programmeren
- Automatisch flow chart genereren module
- Database connectoren ingebouwd

In het kort zijn de nadelen als volgt:

- Jaarlijkse betaalde licentie
- Mogelijke layouts zijn onduidelijk



Figuur 14, *Voorbeeld van een data flow diagram in yWorks met business proces als kern*



Figuur 15, *Voorbeeld van een data flow diagram in yWork met bron-bewerking- bestemming model*

5. Conclusie

In dit document is onderzoek naar verschillende onderdelen van data flow visualisatie. Allereerst is er kort stilgestaan bij de ISMetadata software. Hierbij is geanalyseerd op welke metadata genereerd wordt en welke informatie deze metadata bevat (zie hoofdstuk 4.1). Als tweede is er onderzoek gedaan naar huidige vormen van data flow visualisatie. Hiermee wordt geanalyseerd op welke manieren data flows grafisch kunnen worden weergegeven. Het laatste deel van het onderzoek heeft aandacht besteed aan bestaande Java library's om data (flow) grafisch weer te geven.

In dit hoofdstuk zullen aan de hand van de resultaten en deelvragen conclusies worden getrokken. Met deze conclusies wordt de hoofdvraag van dit onderzoek beantwoord. Verder eindigt dit hoofdstuk met een advies betreffende het lopende project bij Info Support over data flow visualisatie.

5.1 Deelvragen

Om de deelvragen; wat moet er van een dataflow worden gevisualiseerd?, welke eisen stelt dit aan de metadata? Er is in het interview met de heer van Bilsen duidelijkheid gekomen over de eisen van de visualisatie. Hij geeft aan dat de volgende onderdelen van de metadata gevisualiseerd dienen te worden:

- bronattributen met naam, tabelnaam en databasenaam
- bestemmingsattributen met naam, tabelnaam en databasenaam
- bewerking die plaatsvindt op het bronattribuut alvorens deze het bestemmingsattribuut wordt

De metadata die wordt gegenereerd door ISMetadata bevat naar de volgende onderdelen:

- Beschrijving van de structuur van bronsystemen en staging area's (tabellen en kolommen)
- Model beschrijvingen van de datavaults (tabellen, kolommen en relaties)
- Bewerkingen (incl. Mappings) tussen de bronsystemen en de datavault (op attribuutniveau)
- Bewerkingen (incl. Mappings) tussen de datavaults en het datawarehouse (op attribuutniveau)

In de metadata gegenereerd door ISMetadata worden de bewerkingen tussen de verschillende systemen op attribuutniveau opgeslagen. Hierdoor is het mogelijk om de attribuutnaam van het bronsysteem te achterhalen. Verder worden de bewerkingen met een naam beschreven. Zodat het attribuut uit het bronsysteem aan de bewerkingsvorm kan worden gekoppeld. En deze op zijn beurt aan het attribuut in het datawarehouse. Hierdoor is het mogelijk om van de metadata gegenereerd door ISMetadata een visualisatie te creëren waarbij de attribuutnaam in het bronsysteem, bewerking met naam en attribuutnaam in het datawarehouse worden getoond. Het is dus mogelijk om met ISMetadata metadata de gewenste visualisatie te produceren.

Het onderzoek heeft zich hierna gericht op de volgende twee deelvragen:

- Op welke manieren kan een dataflow worden gevisualiseerd?
- Welke visualisatie is het meest geschikt voor de gegenereerde metadata van ISMetadata?

Zoals gebleken is uit de wensen van de heer Van Bilsen moeten er drie onderdelen duidelijk worden uit de visualisatie: het bron attribuut, de bewerking en het bestemming attribuut. De meest geschikte vorm voor deze eisen is het bron-bewerking-bestemming model. Hierbij zijn de drie onderdelen vertegenwoordigd in de visualisatie. In de metadata van ISMetadata zijn deze onderdelen terug te vinden (zie begin hoofdstuk). Hierdoor is het mogelijk om de gewenste visualisatie te plaatsen in het bron-bewerking-bestemming model.

De laatste deelvraag die in dit onderzoek is beantwoord is: welke code library's van Java zijn van toepassing op het visualiseren van data?
Hieronder in het kort de voor en nadelen van de geanalyseerde library's:

Library naam	Voordelen	Nadelen
Netbeans Visual	- Gebruik maken van JavaFX - Widget objecten met dynamische eigenschappen, zoals children objecten, listeners en pijlen - Exporteer mogelijkheden naar Jpeg of PDF - Animaties toevoegen aan diagrammen	- Beperkt aantal layout mogelijkheden, grid en decision tree (layout is de wijze waarop de elementen worden gestructureerd bij de weergave) - Geen directe database integratie
Jung	- Grote diversiteit aan layouts beschikbaar - Ontworpen voor complexe relationele data	- Beperkte grafische elementen (alleen basis vormen zoals cirkels, vierkanten sterren)
yWorks	- Visueel programmeren - Automatisch flow chart genereren module - Database connectoren ingebouwd	- Jaarlijkse betaalde licentie - Mogelijke layouts zijn onduidelijk

5.2 Hoofdvraag en advies

De hoofdvraag van dit onderzoek is:
Welke vorm van data flow visualisatie is het meest geschikt om de data flow informatie uit ISMetadata weer te geven en welke Java code library's kunnen dit ondersteunen?

Uit de analyse van data flow visualisatie blijkt dat de wensen van Info Support om de metadata van ISMetadata het beste kunnen worden gevisualiseerd met de hulp van het bron-bewerking-bestemming model. In dit model kan, naar wens van de opdrachtgever, op attribuut niveau de data flow worden weergegeven. Hierbij worden de bewerkingen en de uiteindelijke eindbestemming ook getoond. Dit geeft volgens de eisen van de opdrachtgever een compleet beeld van de data flow uit ISMetadata. Het gebruik van dit model wordt dan ook sterk aangeraden.

Uit het voor onderzoek blijkt dat het gebruik van bestaande software onmogelijk is. Deze software is gericht op het handmatig modelleren van proces data flows. Voor dit onderzoek is het visualiseren van metadata uit ISMetadata het centrale uitgangspunt. Deze metadata beschrijft relationele data die wordt verplaatst naar een andere database. Dit zou automatisch moeten worden gegenereerd. Hierdoor is het gebruik van bestaande software onmogelijk en zal er een maatwerk oplossing moeten worden gebouwd.

Het gebruik van de Java library's Netbeans Visual en Jung kan ondersteuning bieden tot het genereren van een data flow visualisatie. Jung kan ervoor zorgen dat het diagram het juiste data flow model aanhoudt. Verder kan het gebruik van Netbeans Visual ervoor zorgen dat elementen extra informatie kunnen bevatten. Dit door gebruik te maken van de widget objecten uit deze library. Het advies is dan ook om zowel Netbeans Visual als Jung te gebruiken. Zodat de voordelen van beide library's kunnen worden gebruikt. Op deze manier kan er een veelzijdige visualisatie met verschillende mogelijke layouts worden geproduceerd.

6. Bronnenlijst

Vooronderzoek:
http://nl.wikipedia.org/wiki/Data_flow_diagram
http://en.wikipedia.org/wiki/Dataflow
http://www.edrawsoft.com/Data-Flow-Diagrams.php

Vormen van data flow visualisatie:
http://www.visualisingdata.com/index.php/2012/06/energy-technologies-visualisation-for-the-iea/
http://www.businessinsider.com/data-visualization-of-the-black-budget-2013-9?IR=T
http://www.searchengineoptimizations.co/2011/12/01/new-google-analytics-features/

Bestaande software:
http://www.graphviz.org/
http://www.conceptdraw.com/How-To-Guide/data-flow-diagram-symbols
http://www.conceptdraw.com/examples/dfd-diagram
http://www.edrawsoft.com/Data-Flow-Model-Diagram.php

Java library's:
http://bits.netbeans.org/dev/javadoc/org-netbeans-api-visual/overview-summary.html
https://platform.netbeans.org/tutorials/nbm-quick-start-visual.html
http://jung.sourceforge.net/
http://stackoverflow.com/questions/6162618/java-graph-library-for-dynamic-visualisation
https://github.com/timboudreau/vl-jung
http://java.dzone.com/news/how-create-visual-applications
https://community.oracle.com/thread/1660768
http://www.manageability.org/blog/stuff/open-source-graph-network-visualization-in-java/view
http://www.yworks.com/en/products_yfiles_about.html
http://java.dzone.com/news/how-create-visual-applications

Requirements

Data flow visualisatie

Titel	Requirements
Project/Onderwerp	Data flow visualisatie
Versie	1.3
Status	Internal Draft
Datum	07-mei-2015
Bestand	Requirements discipline
Bedrijf	Info Support B.V.

Historie

Versie	Status	Datum	Auteur	Verandering
1.0	Concept	12-03-2015	Maarten Koene	Creatie
1.3	Definitief	7-5-2015	Maarten Koene	Aanvulling van gevonden requirements en priotering

Inhoudsopgave

1. Inleiding 4

2. System Scope 5

3. Requirements 6

3.1 Functionele requirements 6

3.2 Niet functionele requirements 6

4. Use case diagram 7

4.1 Use case diagram 7

4.2 Actor list 8

4.3 Use case lijst 8

5. Requirements prioriteit 9

5.1 Must Have 9

5.2 Should have 9

5.3 Could have 10

5.4 Would have 10

6. Technische constraints 11

7. Use case beschrijvingen 12

7.1 Use Case 1 12

7.2 Use Case 2 12

7.3 Use case 3 13

1. Inleiding

Het project omtrent het creëren van een data flow visualisatie tool wordt ondersteund door een reeks van documenten. De tool moet een grafische weergave genereren op de opgeslagen metadata in de ISMetadata database. Deze database kan handmatig en geautomatiseerd worden gevuld met metadata. De data beschrijft onder andere de data flow, de verplaatsing van data van bron naar bestemming.

Om een overzichtelijke situatie te creëren voor ontwerpers en ontwikkelaars wordt in dit project een visualisatie tool gemaakt. In dit document zullen de wensen en eisen van de stakeholders worden gepresenteerd. Deze wensen en eisen zijn geuit door Thomas van Bilsen.

In dit document worden de eisen geprioriteerd. Dit wordt in samenwerking met Thomas van Bilsen gedaan. De use cases met requirements, die hoogste prioriteit hebben, worden uitgewerkt. Deze zullen worden uitgewerkt in de documenten reeks van de visualisatie tool(het functioneel-, technisch ontwerp en testrapportage) en in de visualisatie tool.

De opbouw van dit document is als volgt. Er wordt begonnen met de scope van het project en de visualisatie tool vast te leggen. Daarna worden de functionele en niet functionele requirements getoond. In hoofdstuk 4 worden de use cases naar voren gebracht. Deze use cases zijn voortgekomen uit de opgestelde functionele requirements. Per use case wordt aangegeven welke requirements hierbij worden vervult. In hoofdstuk 5 worden de requirements per prioriteit aangegeven. Zoals eerder vermeld komt dit voor uit gesprekken met Thomas van Bilsen. In hoofdstuk 6 worden de technische beperkingen van het systeem naar voren gebracht. Deze kunnen voortkomen uit de bestaande systemen waarmee de visualisatie tool moet samenwerken. Hierna worden de use cases met de hoogst geprioriteerde uitgeschreven in een use case beschrijving. Het document sluit af met een schets van de schermverhoudingen. Deze zijn op basis van de uitgewerkte requirements en kan in een latere iteratie worden uitgebreid.

2. System Scope

Het proces van het ontwikkelen van Business Intelligence systemen wordt ondersteund door het gebruik van de ISMetadata tool. Deze tool heeft een database, deze heet ISMetadata. In deze database wordt metadata van staging area's, datavaults, datawarehouses en datamarts opgeslagen. De staging area's en datavaults worden gemodelleerd door de ISMetadata tool. Dit wordt gedaan door de module datavault Model Generator in ISMetadata. Verder kunnen de datawarehouses en datamarts worden ingeladen in ISMetadata. Door dit te doen kunnen de mappings tussen de datavaults en datawarehouse of datamart worden toegevoegd.

Op basis van de gegenereerde staging area's en datavaults kunnen er ETL schema's worden gegenereerd. Hiermee kan de data uit de bron naar de staging area worden verplaatst. Ook kunnen de ETL's worden gegenereerd om de data vanuit de staging naar de datavault te verplaatsen. De gegenereerde ETL bestaat uit stored procedures en SSIS packages. Dit proces wordt gedaan door de module datavault ETL generator.

Het beschreven proces wordt doorlopen door ontwikkelaars en ontwerpers. Om overzicht te houden in dit proces is er vraag naar een visualisatie. De stakeholders, ontwikkelaars en ontwerpers die ISMetadata gebruiken, willen door middel van een visualisatie van de data flow fouten voorkomen en efficiënter werken. De metadata kan in de verschillende stappen worden aangepast. Als dit niet naar wens is worden deze bewerkingen aangepast. Een visuele representatie van de data in de verschillende stappen kan ondersteunen om het maken van fouten te voorkomen.

Het doel van de reeks documenten, zoals beschreven in het plan van aanpak, is om een systeem te ontwerpen en (gedeeltelijk) ontwikkelen om een visuele representatie te geven van de data flow. Dit om het proces van het ontwerpen en ontwikkelen van Business Intelligence systemen te ondersteunen. De visualisatie van de data flow wordt gedaan op de geproduceerde metadata van de ISMetadata tool. Hierbij kan de data flow worden aangepast in de visualisatie tool.

De gestelde wensen en eisen voor het systeem worden beschreven in dit document. Hierbij zal er door middel van iteraties requirements aan worden toegevoegd. Verder wordt er in overleg met de opdrachtgever een prioriteit gesteld aan requirements. De requirements met de hoogste prioriteit zullen als eerste worden uitgewerkt in het ontwerp en tool.

3. Requirements

3.1 Functionele requirements

Requirement ID	Requirement beschrijving
UR 1	Er kan een systeem worden gekozen uit de ISMetadata database om te visualiseren
UR 2	De data flow moet op attribuut niveau worden gevisualiseerd
UR 3	De brontabellen en databasenaam moeten worden kunnen herleid uit de visualisatie
UR 4	De bestemmingstabellen en databasenaam moeten worden kunnen herleid uit de visualisatie
UR 5	Attributen worden weergegeven in de data flow met een rechthoek
UR 6	De flow van een attribuut moet kunnen worden aangepast
UR 7	De data flow van een attribuut moet kunnen worden verwijderd
UR 8	De aggregatie bewerkingen kunnen worden getoond
UR 9	De aggregatie bewerkingen kunnen worden aanpast
UR 10	De aggregatie bewerkingen kunnen worden verwijderd uit de visualisatie
UR 11	De aggregatie bewerkingen kunnen worden toegevoegd aan de data flow
UR 12	De scalar bewerkingen kunnen worden getoond
UR 13	De scalar bewerkingen kunnen worden aangepast
UR 14	De scalar bewerkingen kunnen worden verwijderd uit de visualisatie
UR 15	De scalar bewerkingen kunnen worden toegevoegd aan de data flow
UR 16	De attributen in de visualisatie kunnen worden gefilterd op schemanaam en tabelnaam
UR 17	De data flow visualisatie wordt in een layout geplaatst. De attributen en transformaties worden op een nader te specificeren wijze op het scherm geplaatst
UR 18	De door de gebruiker gemaakte layout kan worden opgeslagen
UR 19	Per tabelnaam moet het attribuut een unieke kleur krijgen
UR 20	Van het attribuut moet de databasenaam naar voren komen in een wolkje als er met de muis over het attribuut wordt gegaan

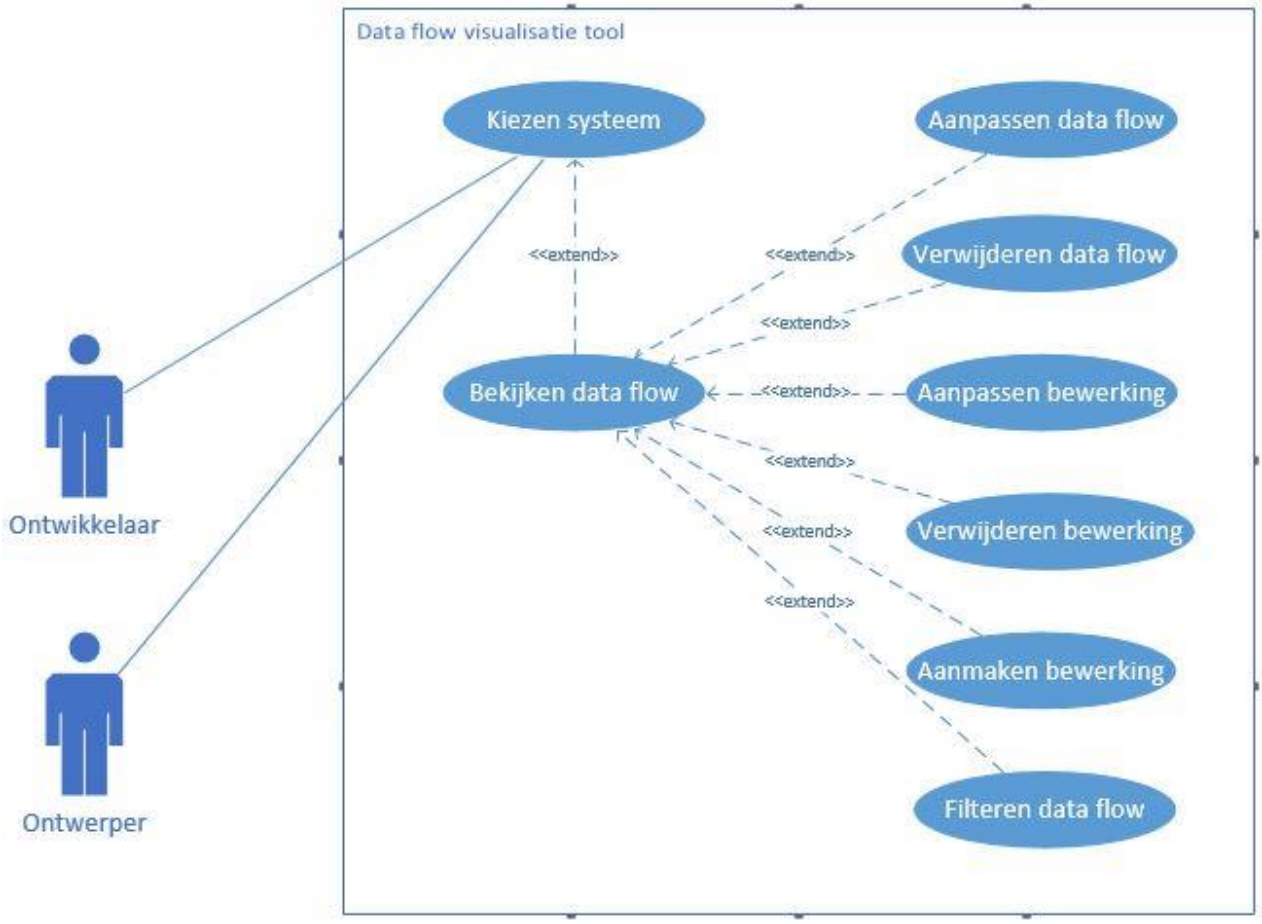
3.2 Niet functionele requirements

Requirement ID	Requirement beschrijving
NFR 1	De data flow visualisatie tool wordt geschreven in het Java platform
NFR 2	De data flow visualisatie tool moet gebruik kunnen maken van Microsoft SQL server
NFR 3	De data flow visualisatie tool moet metadata interpreteren van de ISMetadata database
NFR 4	De data flow visualisatie tool moet binnen 2 seconden een data flow tonen
NFR 5	Bij code libraries moeten gebruik maken van een stabiele release
NFR 6	Bij code libraries moet er een actieve ontwikkelaar zijn voor de library

4. Use case diagram

Op basis van de vergaarde requirements kan er een use case diagram worden gemaakt. Dit is een grafische weergave van de mogelijke gebruikerssituaties. In dit hoofdstuk worden verder ook de actoren naar voren worden gebracht. Er wordt afgesloten met een lijst van de use cases met een korte beschrijving en de bijbehorende requirements.

4.1 Use case diagram



4.2 Actor list

Actor	Role	Beschrijving
Ontwikkelaar/Ontwerper	User	De actor kan de visualisatie en bewerkingen, die beschikbaar zijn uit ISMetadata database, opvragen. Dan kan de actor de flow van attributen bekijken en aanpassen. Verder kan de ontwikkelaar bewerkingen aanmaken, aanpassen.

4.3 Use case lijst

Use Case ID	Use case naam	Beschrijving	Requirement
UC 1	Kiezen systeem	Het kiezen van het gewenste systeem om te visualiseren uit een lijst die wordt gegenereerd uit ISMetadata database	UR 1
UC 2	Bekijken data flow	Het bekijken van de volledige data flow visualisatie	UR 2, UR 3, UR 4, UR 5, UR 8, UR 12, UR 17, UR 19, UR 20
UC 3	Aanpassen data flow	Het aanpassen van de flow van een attribuut in de visualisatie	UR 6
UC 4	Verwijderen data flow	Het verwijderen van de flow van een attribuut in de visualisatie	UR 7
UC 5	Aanpassen bewerking	Het veranderen van een bestaande bewerking	UR 9, UR 13
UC 6	Verwijderen bewerking	Het verwijderen van een bewerking op attribuut/attributen	UR 10, UR 14
UC 7	Aanmaken bewerking	Toevoegen van een nieuwe bewerking op een attribuut of attributen	UR 11, UR 15
UC 8	Filteren data flow	De attributen van een data flow filteren op schemanaam of tabelnaam	UR 16

5. Requirements prioriteit

De hiervoor beschreven requirements zullen in dit hoofdstuk worden geprioriteerd. Dit zal gebeuren met de MoSCoW methode. Deze methode verdeeld de requirements in 4 niveau's via prioriteit, namelijk:

- M - must haves: deze requirements moeten in het eindresultaat terugkomen, zonder deze requirements is het product niet bruikbaar;
- S - should haves: deze requirements zijn zeer gewenst, maar zonder is het product wel bruikbaar;
- C - could haves: deze requirements zullen alleen aan bod komen als er tijd genoeg is;
- W - would haves: deze requirements zullen in dit project waarschijnlijk niet aan bod komen maar kunnen in de toekomst, bij een vervolgproject, interessant zijn.

5.1 Must Have

Requirement ID	Requirement beschrijving
UR 1	Er kan een systeem worden gekozen uit de ISMetadata database om te visualiseren
UR 2	De data flow moet op attribuut niveau worden gevisualiseerd
UR 3	De brontabellen en databasenaam moeten worden kunnen herleid uit de visualisatie
UR 4	De bestemmingstabellen en databasenaam moeten worden kunnen herleid uit de visualisatie
UR 5	Attributen worden weergegeven in de data flow met een rechthoek
UR 8	De aggregatie bewerkingen kunnen worden getoond
UR 12	De scalar bewerkingen kunnen worden getoond

5.2 Should have

Requirement ID	Requirement beschrijving
UR 6	De flow van een attribuut moet kunnen worden aangepast
UR 19	Per tabelnaam moet het attribuut een unieke kleur krijgen
UR 20	Van het attribuut moet de databasenaam naar voren komen in een wolkje als er met de muis over het attribuut wordt gegaan

5.3 Could have

Requirement ID	Requirement beschrijving
UR 7	De data flow van een attribuut moet kunnen worden verwijderd
UR 9	De aggregatie bewerkingen kunnen worden aanpast
UR 10	De aggregatie bewerkingen kunnen worden verwijderd uit de visualisatie
UR 11	De aggregatie bewerkingen kunnen worden toegevoegd aan de data flow
UR 13	De scalar bewerkingen kunnen worden aangepast
UR 14	De scalar bewerkingen kunnen worden verwijderd uit de visualisatie
UR 15	De scalar bewerkingen kunnen worden toegevoegd aan de data flow
UR 16	De attributen in de visualisatie kunnen worden gefilterd op schemanaam en tabelnaam
UR 18	De door de gebruiker gemaakte layout kan worden opgeslagen

5.4 Would have

Requirement ID	Requirement beschrijving
UR 17	De data flow visualisatie wordt in een layout geplaatst

6. Technische constraints

In dit hoofdstuk worden de technische beperkingen naar voren gebracht. Deze kunnen voortkomen uit gestelde eisen of bestaande systemen waarmee moet worden samengewerkt.

- De visualisatie tool moet gebruik maken van de ISMetadata database
- De visualisatie tool moet een connectie kunnen leggen met MSSQL database management systeem
- De visualisatie tool moet geschreven worden in Java
- De visualisatie tool moet bewerkingen gebruiken conform de eisen van ISMetadata

7. Use case beschrijvingen

In dit hoofdstuk worden de use cases uitgewerkt. Dit zijn de use cases waarvan de requirements een must-have prioriteit hebben gekregen. Deze beschrijvingen geven een beeld van de interactie tussen systeem en actor.

7.1 Use Case 1

Naam	Kiezen systeem
ID	UC 1
Beschrijving	Het kiezen van het gewenste systeem om te visualiseren uit een lijst die wordt gegenereerd uit ISMetadata database
Primaire actor(s)	Ontwikkelaar of ontwerper (Actor)
Secondaire actor(s)	-
Pre-conditie(s)	-
Scenario beschrijving	1. Actor start de visualisatie tool 2. Systeem haalt de lijst met beschikbare systemen op[1] 3. Systeem toont de lijst met beschikbare systemen 4. Actor selecteert het gewenste systeem 5. Systeem haalt de attributen met bijbehorende bewerkingen en bestemmingsattributen van het geselecteerde systeem op
Postconditie(s)	De actor heeft het gewenste systeem geselecteerd ter visualisatie
Alternatieve flow	[1] Geen beschikbare systemen 3.Systeem toont een lege lijst van systemen 4.Actor sluit de visualisatie tool af

7.2 Use Case 2

Naam	Bekijken data flow
ID	UC 2
Beschrijving	Het bekijken van de volledige data flow visualisatie
Primaire actor(s)	Ontwikkelaar of ontwerper
Secondaire actor(s)	-
Pre-conditie(s)	Er is een beschikbaar systeem geselecteerd
Scenario beschrijving	1. Systeem haalt de gegevens van het geselecteerde systeem op 2. Systeem toont de data flow van het systeem van de geselecteerde systeem. Waarbij de attributen(met naam, tabelnaam en databasenaam) uit de brontabellen, de attributen uit het bestemmingsmodel(met naam, tabelnaam en databasenaam), aggregatie beweringen en de flow van de data tussen bron en bestemming
Postconditie(s)	De actor kan de gewenste data flow visualisatie bekijken

Alternatieve flow	-

7.3 Use case 3

Naam	**Aanpassen data flow**
ID	UC 3
Beschrijving	Het aanpassen van de flow van een attribuut in de visualisatie
Primaire actor(s)	Ontwikkelaar of ontwerper
Secondaire actor(s)	-
Pre-conditie(s)	Er is een systeem met een data flow geselecteerd
Scenario beschrijving	1. Actor selecteert de data flow pijl van het attribuut 2. Systeem geeft de versleeppunten van de pijl weer 3. Actor selecteert het pijluiteinde dat versleept dient te worden en versleept deze naar de nieuwe transformatie 4. Actor zet de pijl op de nieuwe transformatie[1][2] 5. Actor klikt op opslaan 6. Het systeem slaat de wijziging op
Postconditie(s)	De actor heeft de data flow van een attribuut naar wens gewijzigd
Alternatieve flow	[1] geen transformatie geselecteerd 5. Systeem slaat wijziging niet op [2] Actor selecteert een ander bronattribuut of bestemmingsattribuut 5. Systeem zet de pijl terug op de transformatie

Functioneel ontwerp

Data flow visualisatie tool voor ISMetadata

Titel	Functioneel ontwerp
Project/Onderwerp	Data flow visualisatie tool voor ISMetadata
Versie	1.2
Status	Internal Draft
Datum	20-mei-2015
Bedrijf	Info Support B.V.

Historie

Versie	Status	Datum	Auteur	Verandering
1.0	Concept			Creatie
1.1	Elaboration 2	9-4-2015	Maarten Koene	Eerste volledige versie voor elaboration 2. Eerste twee use cases uitgewerkt.
1.2	Definitief		Maarten Koene	Domein model toegevoegd. Schermontwerp aangepast

Inhoudsopgave

1. Inleiding **4**

1.1 Doel document 4

1.2 Doelgroep en opbouw 4

1.3 Project scope 4

2. Algemene beschrijving **5**

2.1 Use cases 5

2.2 Actor 6

3. Beschrijving van functionaliteiten **7**

3.1 Use case beschrijving 7

3.2 Package diagram 8

4. Schermontwerp **11**

1. Inleiding

1.1 Doel document

Het functioneel ontwerp beschrijft de functies van de te ontwikkelen visualisatie tool. De functionaliteiten zijn opgesteld aan de hand van de requirements. Deze requirements zijn verkregen tijdens de gesprekken met Thomas Bilsen. Deze requirements zijn terug te vinden in het requirements rapport van de visualisatie tool. Het doel van dit document is het beschrijven van de gewenste functionaliteiten van de ISMetadata visualisatie tool.

Aan de hand van het functioneel- en technisch ontwerp zal de ISMetadata visualisatie tool worden ontwikkeld. Het functioneel ontwerp is onderdeel van een reeks documenten. Deze documenten vormen de basis van de ISMetadata visualisatie tool. De documenten in de ISMetadata visualisatie tool reeks zijn: requirements rapport, functioneel ontwerp, technisch ontwerp en testrapportage.

1.2 Doelgroep en opbouw

De ontwerpen van de ISMetadata visualisatie tool zijn gericht op de ontwikkelaar. Deze ontwikkelaar, Maarten Koene, kan op basis van het functioneel en technisch ontwerp de visualisatie tool ontwikkelen. Ook dienen de ontwerpen als basis voor het testrapport. Hierbij kan de tester de ontwerpen gebruiken om de functionaliteiten te testen.

Het functioneel ontwerp is als volgt opgebouwd. Er wordt begonnen met alle opgestelde use cases. Deze zijn voortgekomen uit de opgestelde requirements. Hierna worden de betrokken actoren naar voren gebracht. In het volgende hoofdstuk worden de use case scenario's uitgewerkt. De uitgewerkte use cases zijn voortgekomen uit de geprioriteerde requirements. De hoogst geprioriteerde requirements hebben twee use cases naar voren gebracht. Na het uitwerken van de use cases wordt de structuur van de visualisatie tool geschetst aan de hand van een UML package diagram. Het functioneel ontwerp sluit af met de schermenontwerpen te tonen. Hierbij wordt de globale opzet van het scherm van ISMetadata visualisatie tool besproken.

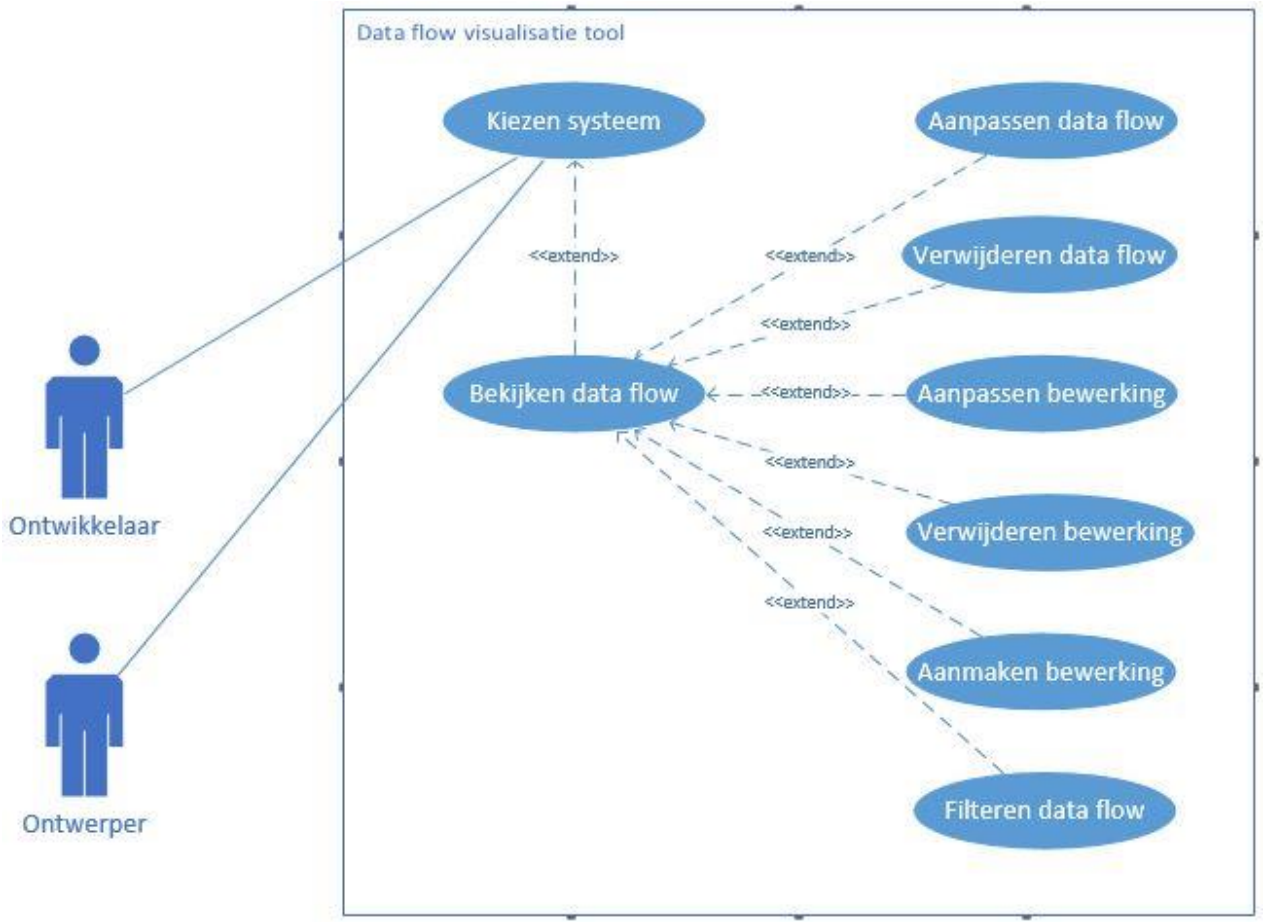
1.3 Project scope

De scope van het functioneel ontwerp is de use cases met de hoogste prioriteit. Hierbij worden de use cases van de ISMetadata visualisatie tool met de, door de opdrachtgever aangegeven, hoogste prioriteit uitgewerkt in het ontwerp. Het ontwerp zal zich dus enkel richten op het beschrijven van de benodigde functionaliteiten van deze use cases voor de ISMetadata visualisatie tool.

2. Algemene beschrijving

In dit hoofdstuk wordt een totaal overzicht geschetst van de functionaliteiten van ISMetadata visualisatie tool. Dit wordt gedaan door alle use cases met afhankelijkheden te tonen. Dit use case diagram is overeenstemmend met het use case diagram in het requirements rapport. Hierna worden de bijbehorende actoren met hun rechten binnen het systeem naar voren gebracht.

2.1 Use cases



2.2 Actor

Actor	Role	Beschrijving
Ontwikkelaar/Ontwerper	User	De actor kan de visualisatie en bewerkingen, die beschikbaar zijn uit ISMetadata database, opvragen. Dan kan de actor de flow van attributen bekijken en aanpassen. Verder kan de ontwikkelaar bewerkingen aanmaken, aanpassen.

3. Beschrijving van functionaliteiten

In dit hoofdstuk zijn de use case beschrijvingen(scenario's) opgenomen. Enkel de beschrijvingen van de use cases die worden uitgewerkt zijn hier te vinden. Deze zijn bepaalt door de prioriteiten die de opdrachtgever heeft gesteld aan de requirements. Voor deze prioritering en compleet overzicht van de use cases zie het Requirements Discipline document van de data flow visualisatie tool.

Verder wordt er in dit hoofdstuk de structuur van de ISMetadata visualisatie tool naar voren gebracht aan de hand van een UML klassendiagram.

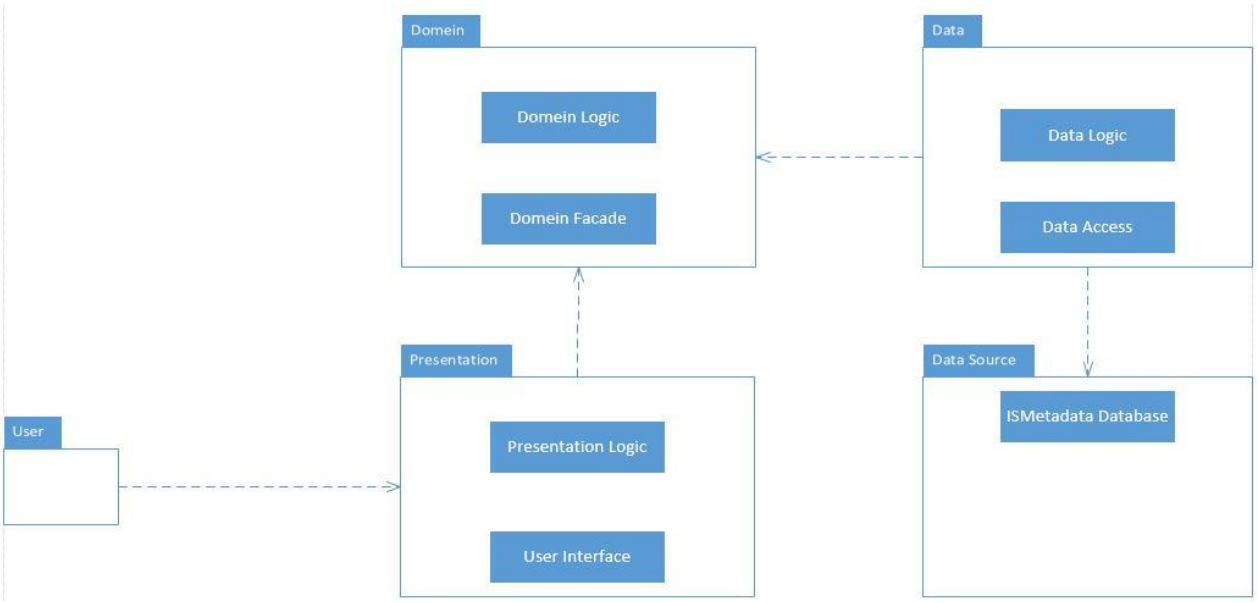
3.1 Use case beschrijving

Naam	Kiezen systeem
ID	UC 1
Beschrijving	Het kiezen van het gewenste systeem om te visualiseren uit een lijst die wordt gegenereerd uit ISMetadata database
Primaire actor(s)	Ontwikkelaar of ontwerper (Actor)
Secondaire actor(s)	-
Pre-conditie(s)	-
Scenario beschrijving	1. Actor start de visualisatie tool 2. Systeem haalt de lijst met beschikbare systemen op[1] 3. Systeem toont de lijst met beschikbare systemen 4. Actor selecteert het gewenste systeem 5. Systeem haalt de attributen met bijbehorende bewerkingen en bestemmingsattributen van het geselecteerde systeem op
Postconditie(s)	De actor heeft het gewenste systeem geselecteerd ter visualisatie
Alternatieve flow	[1] Geen beschikbare systemen 3.Systeem toont een lege lijst van systemen 4.Actor sluit de visualisatie tool af

Naam	Bekijken data flow
ID	UC 2
Beschrijving	Het bekijken van de volledige data flow visualisatie
Primaire actor(s)	Ontwikkelaar of ontwerper
Secondaire actor(s)	-
Pre-conditie(s)	Er is een beschikbaar systeem geselecteerd
Scenario beschrijving	1. Systeem haalt de gegevens van het geselecteerde systeem op 2. Systeem toont de data flow van het systeem van de geselecteerde systeem. Waarbij de attributen uit de brontabellen, de attributen uit het bestemmingsmodel, aggregatie bewerkingen en de flow van de data tussen bron en bestemming[1]
Postconditie(s)	De actor kan de gewenste data flow visualisatie bekijken
Alternatieve flow	[1] Geselecteerde systeem heeft nog geen data flow 2. Er wordt een lege data flow weergegeven

Naam	Aanpassen data flow
ID	UC 3
Beschrijving	Het aanpassen van de flow van een attribuut in de visualisatie
Primaire actor(s)	Ontwikkelaar of ontwerper
Secondaire actor(s)	-
Pre-conditie(s)	Er is een systeem met een data flow geselecteerd
Scenario beschrijving	1. Actor selecteert de data flow pijl van het attribuut 2. Systeem geeft de versleeppunten van de pijl weer 3. Actor selecteert het pijluiteinde dat versleept dient te worden en versleept deze naar de nieuwe transformatie 4. Actor zet de pijl op de nieuwe transformatie[1][2] 5. Actor klikt op opslaan 6. Het systeem slaat de wijziging op
Postconditie(s)	De actor heeft de data flow van een attribuut naar wens gewijzigd
Alternatieve flow	[1] geen transformatie geselecteerd 5. Systeem slaat wijziging niet op [2] Actor selecteert een ander bronattribuut of bestemmingsattribuut 5. Systeem zet de pijl terug op de transformatie

3.2 Package diagram



De Data Flow Visualisatie Tool moet gebruik maken van een database. Deze database wordt gemaakt door ISMetadata. In de database wordt de data flow van de systemen op attribuut niveau beschreven. Om de

mogelijkheid te houden dat de database wordt aangepast, is ervoor gekozen de logica van de database en de applicatie gescheiden te houden. In het package diagram is daarom te zien dat de database in een aparte data source package is geplaatst.

De data uit de database wordt gebruikt om een data flow te kunnen visualiseren. Het data package is binnen de applicatie verantwoordelijk voor het maken van een connectie met de database, het ophalen van de data en het opslaan van aanpassingen. Dit wordt binnen de package gedaan door de Data Access klassen. In de Data Logic van dit package staat alle logica omtrent het ophalen en aanpassen van de data. Hierin worden dus de benodigde query's beheerd om data aan te passen en op te halen. Deze worden dan door de database connectie uitgevoerd op de database. Het resultaat wordt teruggegeven aan de Data Logic. Bij het veranderen van de database dient deze package te woorden aangepast. De connectie zal moeten worden aangepast als deze gebruik gaat maken van een andere DBMS.

De Data package is verbonden aan het domein. Binnen het domein wordt de opgehaalde data uit de Data package geïnterpreteerd. Dit betekent dat de relationele data die wordt gegeven vanuit de Data Logic wordt omgezet naar objecten met behulp van de business logica. Verder worden de aanpassingen in de data flow geregistreerd. Hierbij worden de veranderingen voorbereid om met de data package te worden opgeslagen.

Om de business logica te begrijpen achter een data flow is er gesproken met een expert, in dit geval de opdrachtgever. Er moet worden vastgesteld op welke manier een data flow dient te worden gevisualiseerd. Er is eerder vastgesteld dat er drie verschillende onderdelen zijn in een data flow, namelijk: bronattributen, bewerkingen en bestemmingsattributen. De wijze waarop deze drie onderdelen zich met elkaar verhouden is nog niet vastgesteld. Dit is gedaan door een interview te houden met de opdrachtgever. Deze is opgesteld op dezelfde wijze als bij de requirements analyse. Er zijn hierbij een aantal regels naar voren gekomen. Een data flow voldoet altijd aan deze regels verzekerde de opdrachtgever. Hierbij zijn de volgende regels naar voren gekomen:

Regel Nr	Regel
1	Een bestemmingsattribuut heeft altijd een bewerking, ook al is deze bewerking leeg
2	Een bestemmingsattribuut kan ontstaan uit meerdere bronattributen. Voor een bronattribuut moet de naam, tabel en database en bijbehorende bewerking overeenkomen om dit vast te stellen
3	Een bestemmingsattribuut kan ontstaan zonder bronattributen te hebben
4	Een bewerking kan meerdere bronattributen bewerken
5	Een bronattribuut kan meerdere bestemmingen hebben, deze worden als een nieuwe data flow gezien
6	Een bronattribuut kan geen data flow hebben naar een ander bronattribuut
7	Bij het aanpassen (van een data flow) kan een bronattribuut alleen worden gekoppeld aan een bewerking
8	Bij het aanpassen (van een data flow) kan een bewerking alleen worden gekoppeld aan een bestemmingsattribuut
9	Als een aanpassing ongedaan wordt gemaakt moet er niets worden opgeslagen

De presentation package is verantwoordelijk voor het maken en tonen van de data flow visualisatie. Uit het domein wordt een lijst met objecten en tussenliggende relaties gegeven. De Presentation Logic gaat van de lijst toonbare objecten maken. Deze objecten worden dan door de Presentation Logic op een bepaalde wijze, zie regels hierboven, op het scherm geplaatst. De gebruiker(ontwikkelaar of ontwerper) maakt gebruik van de User Interface. Deze wordt getoond bij het starten van de applicaties. In deze interface kan de gebruiker het systeem selecteren,

waarvan de data flow moet worden getoond. Verder geeft de tekent de Presentation Logic op de User Interface de, uit het domein opgehaalde data flow.

In de User Interface kan de gebruiker de data flow aanpassen. De aanpassingen kunnen alleen worden gedaan binnen de regels van een data flow. Deze aanpassingen worden door de Presentation logic teruggegeven aan het domein. De gebruiker geeft bij de User Interface aan de aanpassingen te willen opslaan. Daarna wordt in het domein package de ongedane aanpassingen verwijderd. De aanpassingen, die niet ongedaan zijn, worden voorbereid voor de data package. In de data package worden de aanpassingen geheel opgeslagen (of geheel ongedaan gemaakt bij een fout).

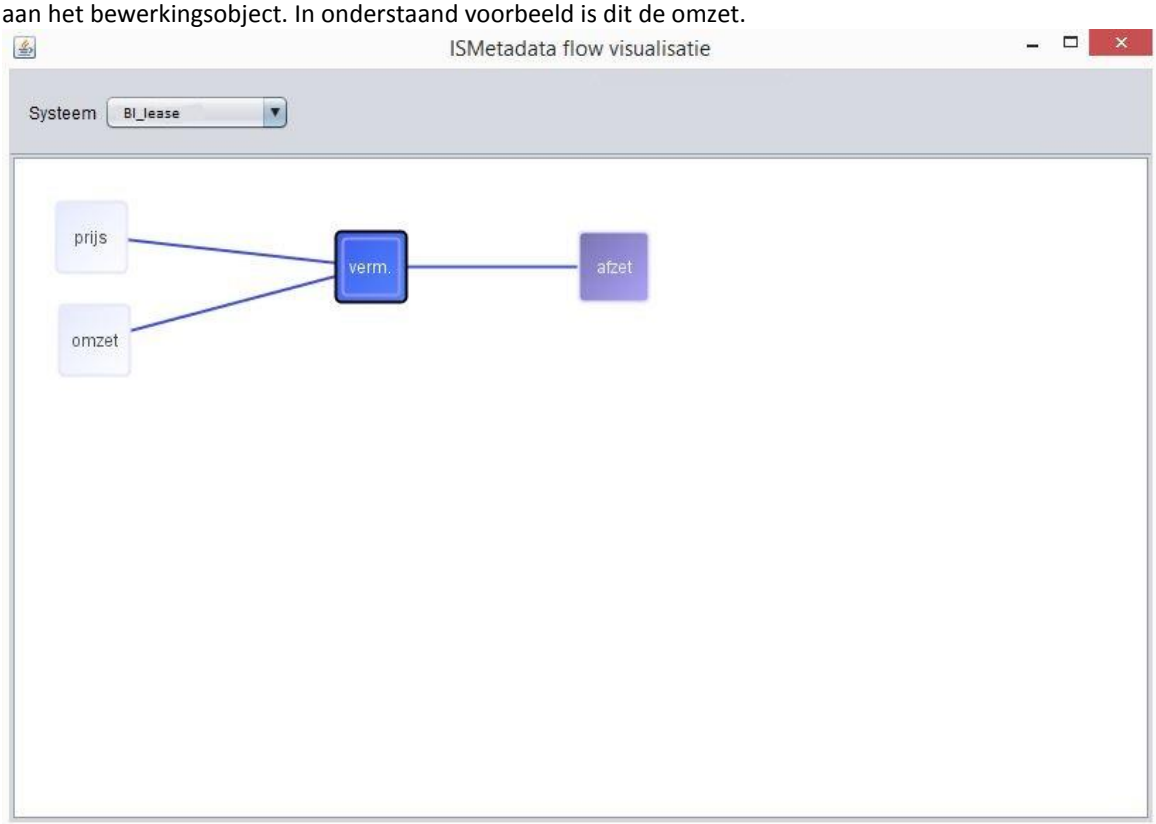
4. Schermontwerp

In dit hoofdstuk wordt het schermontwerp besproken. Het schermontwerp geeft een schets van de User Interface weer. Hierbij worden de elementen in het scherm kort toegelicht.

Het onderstaande figuur is het hoofdscherm. De gebruiker kan in dit scherm verschillende elementen onderscheiden. In het dropdown-menu kan worden gekozen tussen systemen. Een systeem is in dit geval een database. Van deze database worden de attributen beschreven en eventueel naar een andere database verplaatst. Door het kiezen van een database worden alle bijbehorende attributen en hun mapping opgehaald. Hiermee kan dus van het systeem attribuut worden getoond welke bewerking en bestemming deze heeft. Op deze manier wordt er van een systeem de data flow worden getoond van de attributen.

Het tweede element dat te onderscheiden is het tekenvlak. Hierop wordt, na de selectie van een systeem de visualisatie getekend. De grootte van dit element kan worden aangepast door de gebruiker door de randen van de applicatie te verslepen.

Het tekenvlak wordt gevuld met visualisatie objecten deze kunnen van drie verschillende vormen zijn. De bron attributen worden weergegeven met een rechthoek met attribuutnaam erin. Deze zijn via een lijn verbonden aan het bewerkingsobject. Deze is in het voorbeeld een blauw gekleurde rechthoek met de naam van de bewerking erin. Het derde soort attribuut is de bestemming attribuut, deze is paars in het voorbeeld. Deze is met een lijn verbonden aan het bewerkingsobject. In onderstaand voorbeeld is dit de omzet.



Technisch ontwerp

Data flow visualisatie tool voor ISMetadata

Titel	Technisch ontwerp
Project/Onderwerp	Data flow visualisatie tool voor ISMetadata
Versie	1.3
Status	Internal Draft
Datum	14-apr-2015
Bestand	Technisch ontwerp
Bedrijf	Info Support B.V.

Historie

Versie	Status	Datum	Auteur	Verandering
1.0	Concept			Creatie
1.1	definitief		Maarten Koene	Invoegen sequentie diagram en het invoeren van het view-domein model.
1.2	Definitief		Maarten Koene	Aanvullen met de technische aspecten van de use case aanpassen data flow
1.3	Definitief		Maarten Koene	Up to date brengen van diagrammen

Inhoudsopgave

1. Inleiding 4

1.1 Doel van het document 4

1.2 Doelgroep en opbouw 4

1.3 Project scope 4

2. Techniek 5

3. Data flow visualisatie tool 6

3.1 Presentation klassendiagram 6

3.2 Domein klassendiagram 7

3.3 Data klassendiagram 9

3.4 ISMetadata database 9

3.4.1 Klassendiagram 9

3.4.2 Query's 10

3.5 Sequence diagram 12

3.5.1 getSystems() 12

3.5.2 getDataflow() 12

3.5.3 saveChanges() 15

1. Inleiding

1.1 Doel van het document

Het technisch ontwerp beschrijft de technische beslissingen van de te ontwikkelen visualisatie tool. Dit document beschrijft op technisch niveau de applicatie. Het doel van dit document is om een technische basis te leggen van de visualisatie tool. Het technisch ontwerp dient als basis tot het ontwikkelen van de ISMetadata visualisatie tool.

Het technisch ontwerp is een onderdeel van een reeks documenten. Deze documenten reeks vormt de basis tot het ontwikkelen van de ISMetadata visualisatie tool. Het requirements document v1.3 en het functioneel ontwerp v1.2 zijn de voorgaande documenten uit deze reeks.

1.2 Doelgroep en opbouw

De ontwerpen van de ISMetadata visualisatie tool zijn gericht op de ontwikkelaar. Het functioneel en technisch ontwerp zijn de basis documenten waarop de ontwikkeling van de tool zal plaatsvinden. Verder zullen de ontwerpen als basis dienen voor het testproces en rapport.

Het technisch ontwerp is als volgt opgebouwd. Er wordt begonnen met het naar voren brengen van de code library's. Deze zijn voort gekomen uit het onderzoeksrapport van dit project.

1.3 Project scope

De scope van het technisch ontwerp is het waarmaken van de functionaliteiten uit het functioneel ontwerp. Hierbij worden de belangrijkste requirements bewerkstelligd. Verder geeft het technisch ontwerp geen verdere beschrijving van ISMetadata. Omdat de werking van deze tool geen invloed heeft op de visualisatie tool. Beide tools zullen echter gebruik maken van dezelfde database. Daarom is deze voor de volledigheid meegenomen in dit rapport.

2. Techniek

In dit hoofdstuk worden de te gebruiken technieken behandelt. Hierbij worden code library's en het te gebruiken ontwerp patroon naar voren gebracht.

Uit het onderzoek blijkt dat er een aantal code library's zijn van Java die gebruikt kunnen worden om een data flow te visualiseren. Hierbij is JUNG een library die de layout van de visualisatie kan automatiseren. Deze library zorgt ervoor dat de objecten die getekend dienen te worden niet over elkaar heen worden geplaatst. Daarentegen worden layout vormen gebruikt als hiërarchieën of boomstructuren. Er zijn een aantal standaard klassen in deze library die ervoor kunnen zorgen dat de bron-, bewerking-, bestemmingsobjecten niet over elkaar heen worden getekend. In een latere iteratie wordt verder ingegaan op het toepassen van een gewenste layout op de visualisatie. In dit ontwerp wordt enkel uitgegaan van het voorkomen dat objecten over elkaar heen getekend worden.

De tweede te gebruiken library is de Netbeans Visual library. Deze library is erop gericht om door middel van een aangepast Scene object te tekenen. Hierbij worden Widget objecten aangemaakt. Deze objecten hebben hun eigen vorm en eigenschappen. Zo kunnen ze worden geclicked, versleept, connecties met andere objecten maken en nog veel meer. Met het voordeel dat ze overerven van een Scene object. Hierdoor kunnen deze snel en efficiënt worden getekend in het tekenvlak. Op deze manier kan er op een eenduidige wijze worden getekend waarbij de object ieder zelf hun eigenschappen kunnen aanpassen.

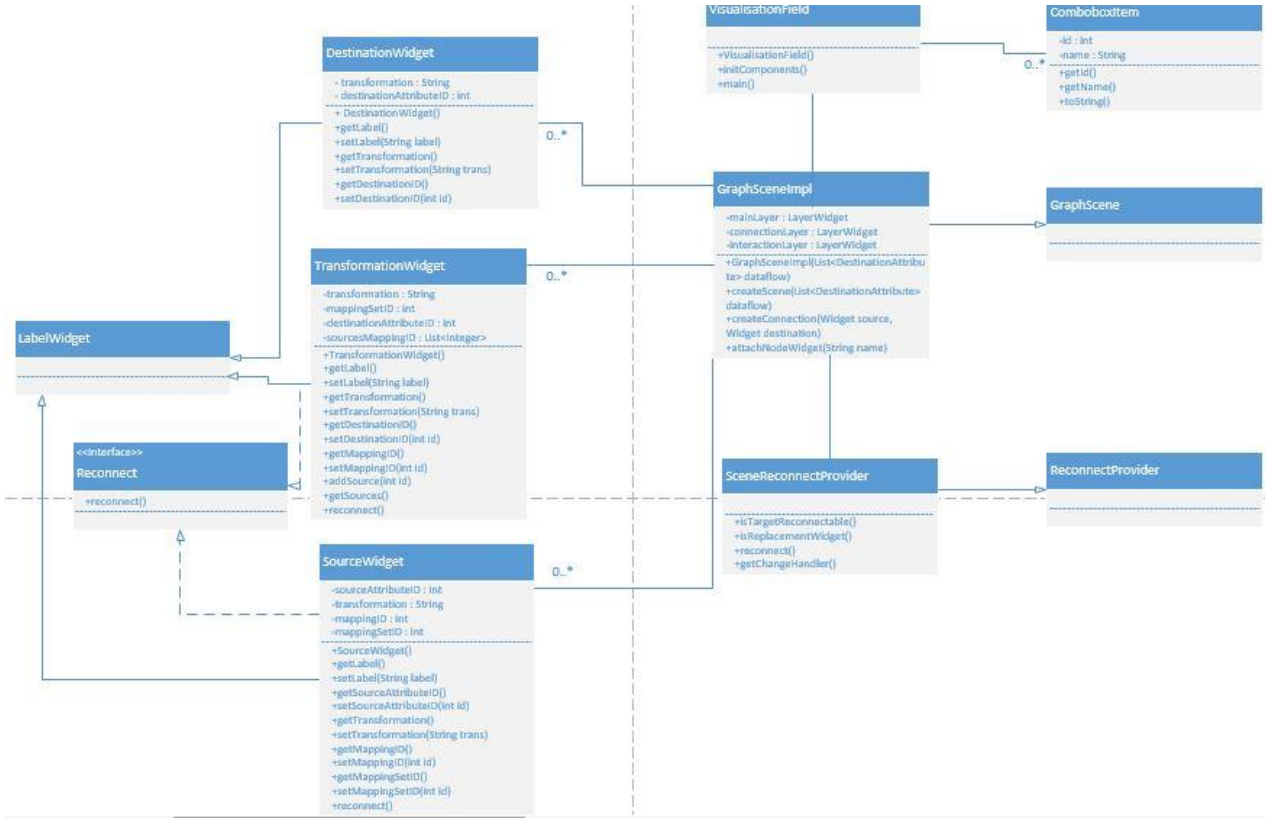
Verder dient er een connectie te worden gemaakt met een MSSQL database. Deze database wordt gemaakt door de ISMetadata tool. De structuur hiervan is te vinden in hoofdstuk 3.2. De database bevat bronattributen, bewerking en bestemmingsattributen. Deze drie verschillende objecten dienen met elkaar te worden verbonden zodat er een dataflow kan worden weergegeven. Om de connectie met de database te realiseren kan gebruik worden gemaakt van JDBC library van Java.

De applicatie zal in drie onderdelen worden verdeeld. De visualisatie wordt gedaan door een view. Hierin wordt enkel de data flow opgehaald en getekend. Het tweede deel is het domein. In het domein wordt het geraamte van de data flow opgezet. Dit wordt gedaan door de data in objecten te plaatsen en de relaties tussen attributen met bewerkingen vast te leggen. Dit gebeurd met de data uit het derde deel, het database connectie deel. Dit deel is verantwoordelijk voor de communicatie met de database. Het verbinden en ophalen van de data wordt in deze module gedaan.

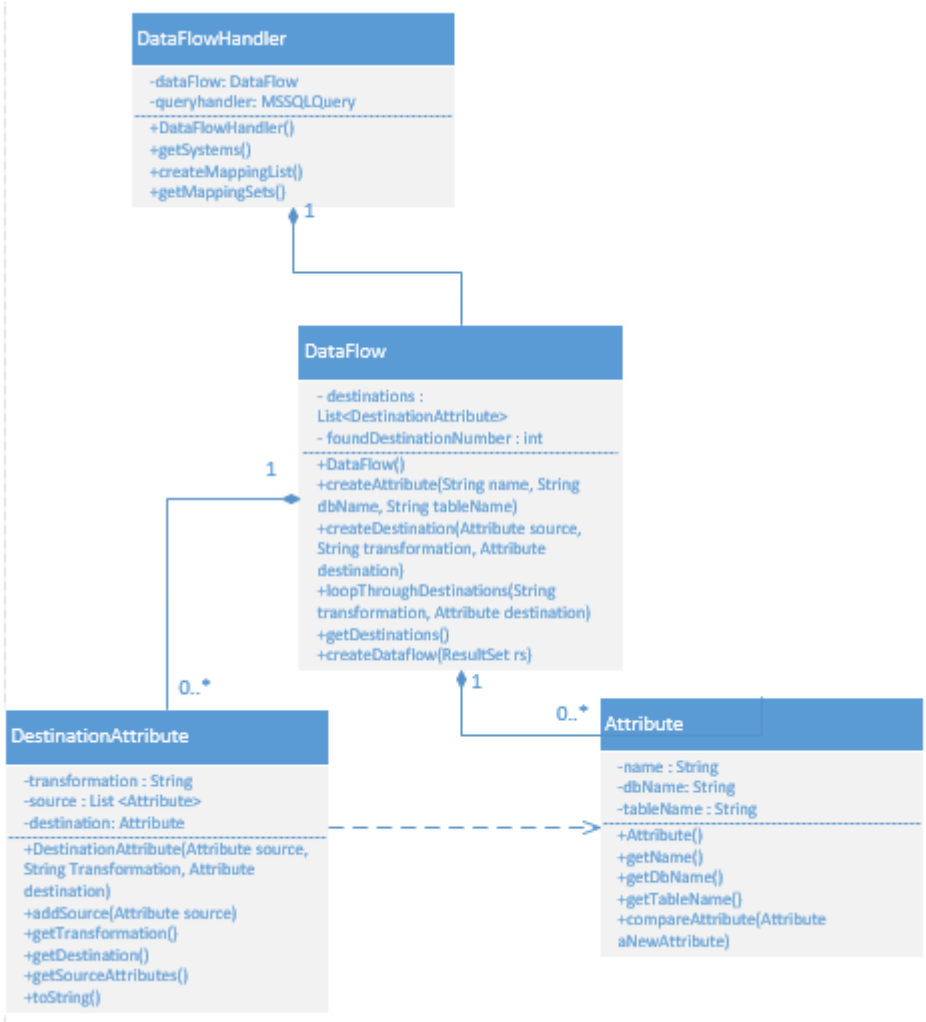
3. Data flow visualisatie tool

In dit hoofdstuk worden de delen van het package diagram van het functioneel ontwerp besproken. In het functioneel ontwerp is aangegeven dat de structuur van de applicatie uit meerdere delen bestaat. In dit hoofdstuk worden de verschillende delen naar voren gebracht. Hierbij wordt het klassendiagram van ieder module besproken. Ook wordt de wijze waarop de verbindingen met de andere modulen wordt gemaakt naar voren gebracht. Het hoofdstuk eindigt met de sequentie diagrammen van het ophalen van de beschikbare systemen en het ophalen van de bijbehorende data flow.

3.1 Presentation klassendiagram



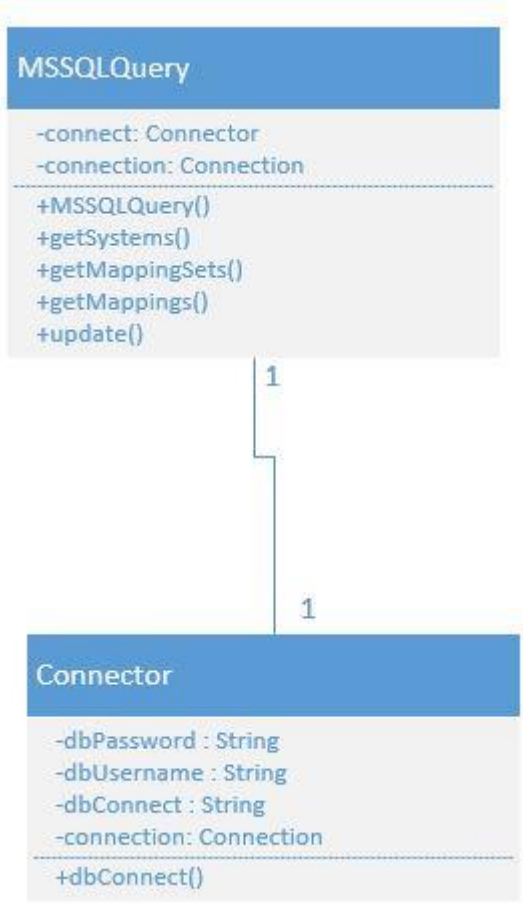
3.2 Domein klassendiagram



Data flow ophalen en toonbaar maken.

Beheren van aanpassingen. Weghalen van ongedaan gemaakte acties.

3.3 Data klassendiagram



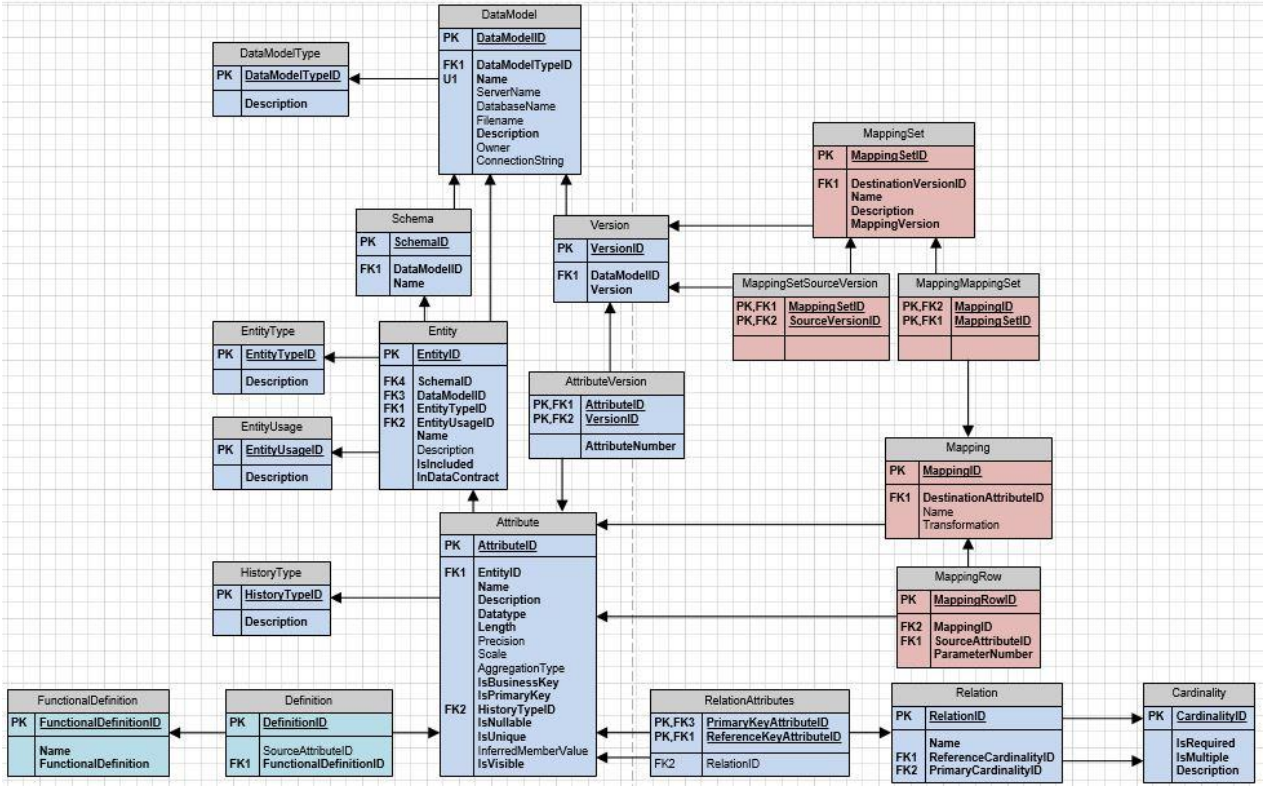
3.4 ISMetadata database

Voor het visualiseren van de data flow wordt gebruik gemaakt van een database. Deze database wordt door de ISMetadata tool gebruikt. Hierin worden de structuren van systemen ingeladen. Hierbij kunnen er datavaults, staging area's en datawarehouse worden ingeladen. Ook kunnen er verschillende soorten structuren worden gegenereerd. Tussen deze systemen kan een data flow worden aangemaakt. Deze worden ook opgeslagen in de ISMetadata database. Daarom is in dit hoofdstuk het klassendiagram meegenomen van de database.

3.4.1 Klassendiagram

Het klassendiagram is afkomstig uit de documentatie van ISMetadata en dus een exacte kopie. Voor het visualiseren van de data flow zijn er een aantal tabellen van belang. In het onderstaande figuur beschrijven de rode tabellen de mappings. Dit is de data die nodig is om een data flow te visualiseren. Echter zal er ook informatie over de

attributen en systemen worden gehaald uit de datamodel tabel en de attribute tabel. Dit zodat er een leesbare visualisatie kan ontstaan. De keuzen hiervoor komen voort uit de gestelde eisen van de opdrachtgever, zoals deze staan beschreven in het requirements document.



Voor een gedetailleerde beschrijving van de attributen kan worden gekeken in de datadictionary van de ISMetadata database.

3.4.2 Query's

Om een data flow te kunnen visualiseren moet er data worden opgehaald uit de database. In dit hoofdstuk worden de query's getoond die nodig zijn om de benodigde data op te halen.

Uit de wensen van de opdrachtgever blijkt dat er een systeem moet kunnen worden gekozen waarvan de data flow moet worden gevisualiseerd. Daarom moet er eerst een lijst worden getoond die alle beschikbare systemen ophaalt. De onderstaande query haalt alle ingeladen systemen:

```
SELECT [DataModelID]
      ,[DataModelTypeID]
      ,[Name]
      ,[DatabaseName]
      ,[Description]
  FROM [ISMetadata].[ismd].[DataModel]
```

Nadat er een systeem is gekozen moet de data flow worden opgehaald. Hierbij zijn de bronattributen met bijbehorende bewerkingen en de bestemmingsattributen nodig om op te halen. Dit wordt in een aantal stappen gedaan.

Als eerste moeten de MappingSet(s) van het gekozen systeem worden opgehaald. Het DataModelID wordt bepaalt door de keuze van systeem door de gebruiker. Deze query haalt alle MappingSet(s) op van de datamodel:

```
SELECT [MappingSetID]
      ,[DestinationVersionID]
      ,[Name]
      ,[Description]
      ,[MappingVersion]
  FROM [ISMetadata].[ismd].[MappingSet]
  WHERE [DestinationVersionID] = GekozenDataModelID;
```

Hierna worden van iedere MappingSet de bestemmingsattributen, transformatie en bronattribuut opgehaald. Dit gebeurd voor elke MappingSet die bij het gekozen systeem hoort. Het MappingSetID wordt dus bepaalt door eerdere keuzes. De volgende query haalt dus de informatie over de bestemmingsattributen op:

```
SELECT Dest.MappingID, Dest.DestinationAttributeID, DestAttr.Name, DTabel.Name,
DDatabase.Name, Dest.Transformation, Src.MappingID, Src.SourceAttributeID, SrcAttr.Name,
STabel.Name,SDatabase.Name
FROM ISMetadata.ismd.MappingMappingSet MMS
LEFT JOIN ISMetadata.ismd.Mapping Dest
ON MMS.MappingID = Dest.MappingID
LEFT JOIN ISMetadata.ismd.MappingRow Src
ON Src.MappingID = MMS.MappingID
LEFT JOIN ISMetadata.ismd.Attribute DestAttr
ON Dest.DestinationAttributeID = DestAttr.AttributeID
LEFT JOIN ISMetadata.ismd.Entity DTabel
ON DestAttr.EntityID = DTabel.EntityID
LEFT JOIN ISMetadata.ismd.DataModel DDatabase
ON DTabel.DataModelID = DDatabase.DataModelID
LEFT JOIN ISMetadata.ismd.Attribute SrcAttr
ON Src.SourceAttributeID = SrcAttr.AttributeID
LEFT JOIN ISMetadata.ismd.Entity STabel
ON SrcAttr.EntityID = STabel.EntityID
LEFT JOIN ISMetadata.ismd.DataModel SDatabase
ON STabel.DataModelID = SDatabase.DataModelID
WHERE MappingSetID = GekozenMappingSetID;
```

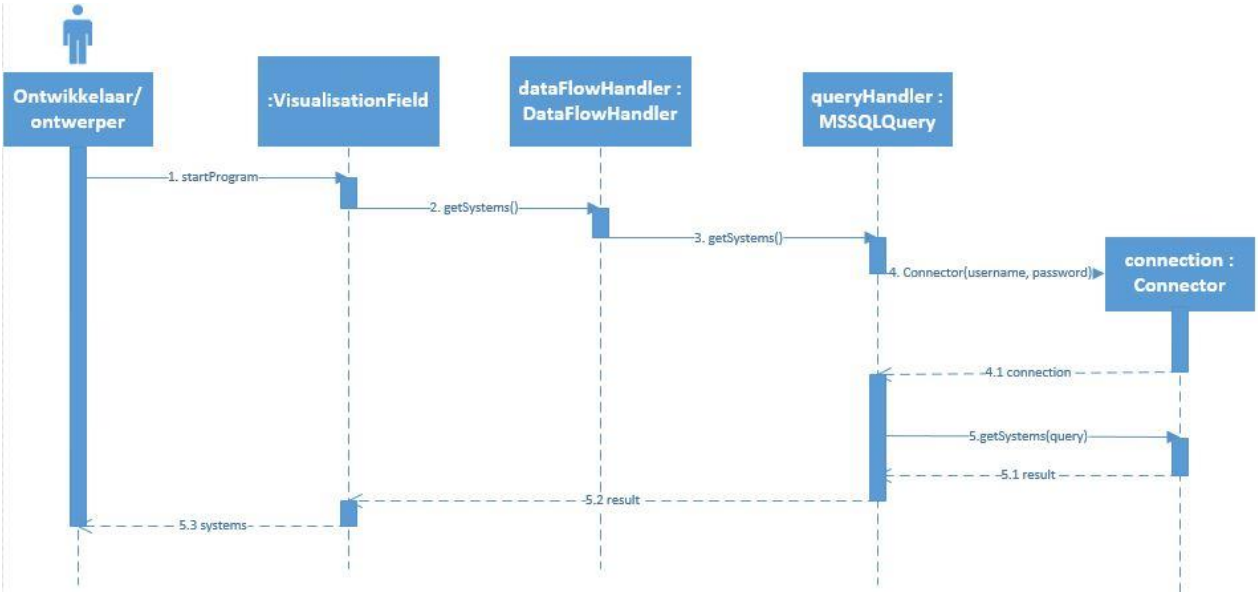
Met de onderstaande query kan de data flow van een bestaande mapping worden aangepast. Hierbij wordt data doorgegeven vanuit het domein. Hierdoor kan voor elke soort bewerking de mapping met dezelfde query worden aangepast. Van belang is het gebruik van een transactie, zodat bij een foute update er teruggedraaid wordt naar de vorige stabiele status.

```
UPDATE ISMetadata.ismd.Mapping
SET [DestinationAttributeID] = NieuweDestinationID,
       [Transformation] = NieuweTransformatie
        WHERE [MappingID] = OudeMappingID;
```

3.5 Sequence diagram

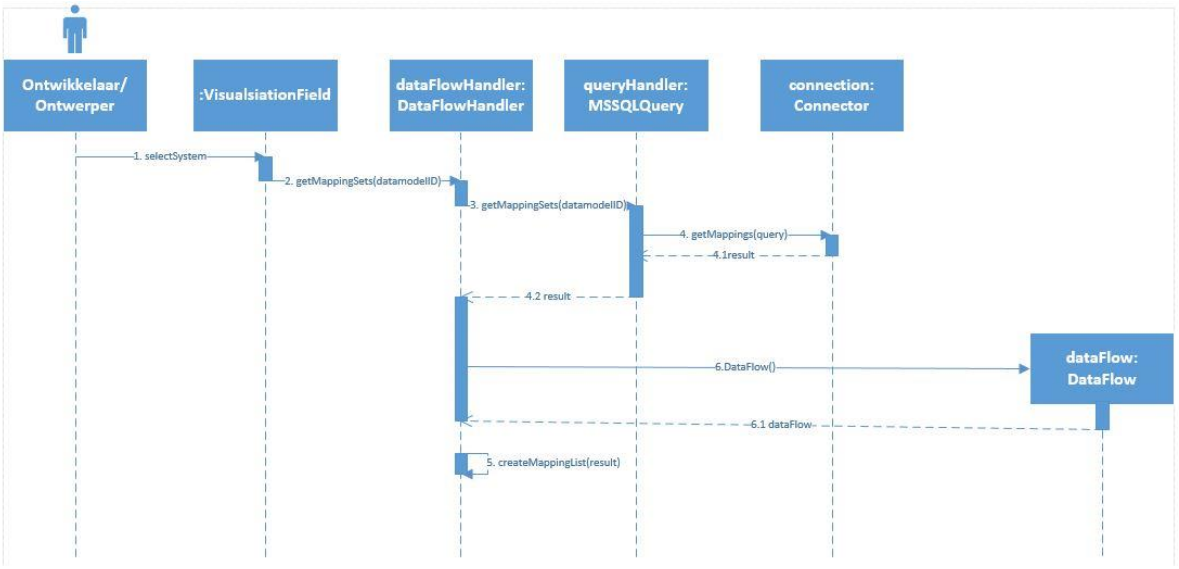
In dit hoofdstuk worden sequentie diagrammen getoond. Deze beschrijven de sequentie die wordt doorlopen om de systemen uit de database naar de frontend te krijgen. Verder wordt het ophalen en maken van de dataflow getoond. Dit sequentie diagram is opgesplitst in 4 delen. Ieder deel is het vervolg op het vorige deel. De stappen zijn genummerd en worden op die wijze doorlopen.

3.5.1 getSystems()

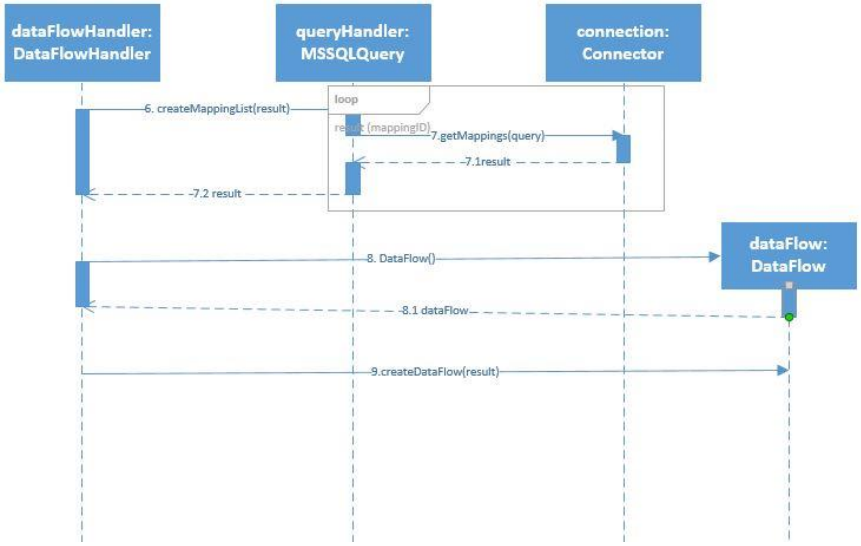


3.5.2 getDataflow()

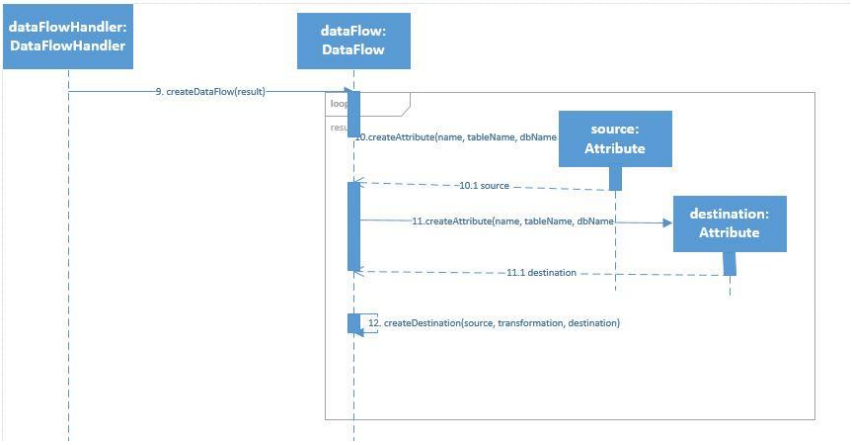
De gebruiker selecteert volgens de use case “kiezen systeem” een systeem uit de getoonde lijst (voor het sequentie diagram van deze actie, zie het technisch ontwerp). Hierna wordt via de dataFlowHandler een aanvraag gestuurd naar de data package. De data package haalt uit de database met de bovenstaande query, om mappingSet(s) op te halen, de bijbehorende mappingSet(s) op. Als er mappingSets bestaan wordt er door de dataFlowHandler een dataFlow object aangemaakt. Echter als er geen mappingSets worden opgehaald wordt er een melding getoond aan de gebruiker dat er geen data flow beschikbaar is voor het systeem.



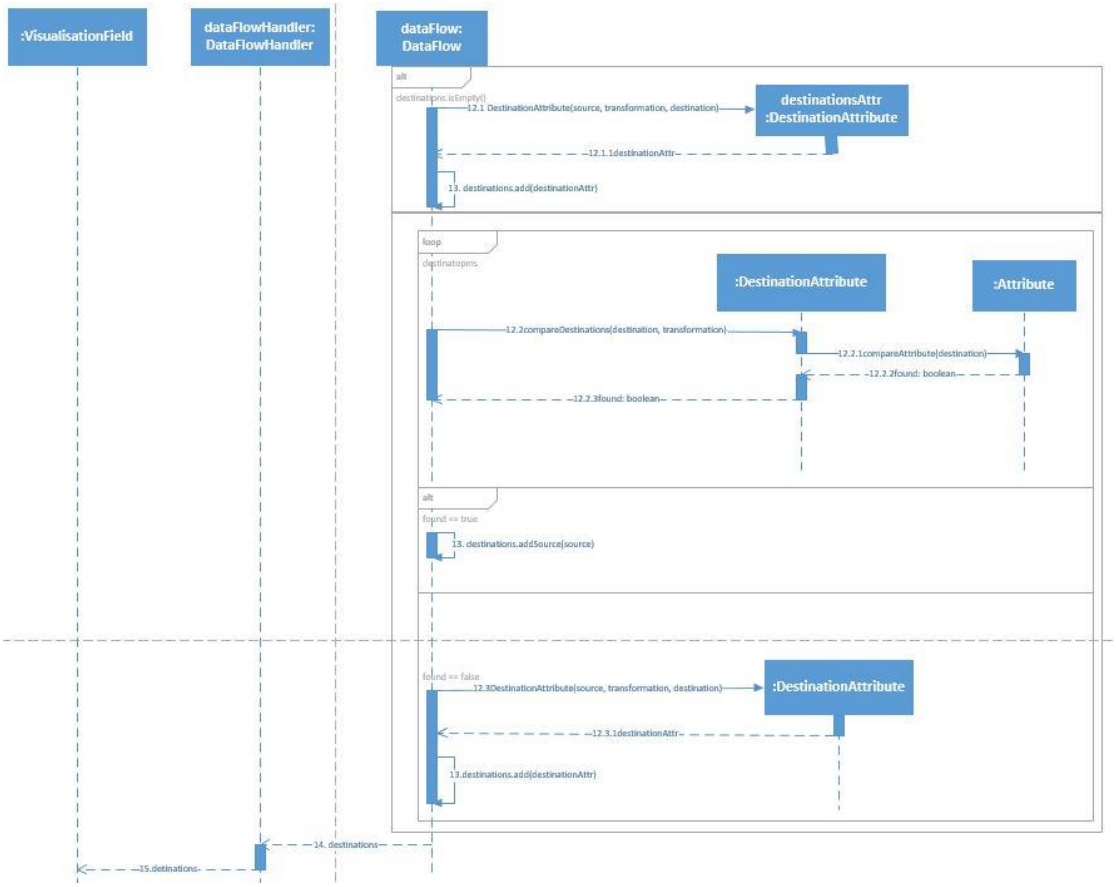
Als er mappingSets zijn opgehaald, dan wordt er voor iedere mappingSet de bijbehorende mappings opgehaald. Dit wordt gedaan door voor iedere mappingSet de derde query uit vorig hoofdstuk af te vuren. Het resultaat wordt voor iedere mappingset teruggegeven. De reeds aangemaakte data flow maakt uit ieder gegeven resultaat een objecten die de data flow representeren.



De opgehaalde data bestaat uit drie verschillende onderdelen. Zo wordt van ieder bronattribuut de bijbehorende bewerking en bestemming opgehaald. Aan de aangemaakte data flow wordt het resultaat van de opgehaalde mappings van een mappingSet gegeven. Het data flow object maakt van iedere regel in het resultaat een aantal objecten aan. Allereerst wordt van de data het bronattribuut aangemaakt. Hierna wordt een bestemmingsattribuut gemaakt op dezelfde manier. Nadat dit gebeurd is wordt de data gekoppeld in de methode createDestination. Hierbij worden wordt uit de data ook de bewerking opgehaald en in combinatie met de gemaakte bron- en bestemmingsattribuut meegegeven.



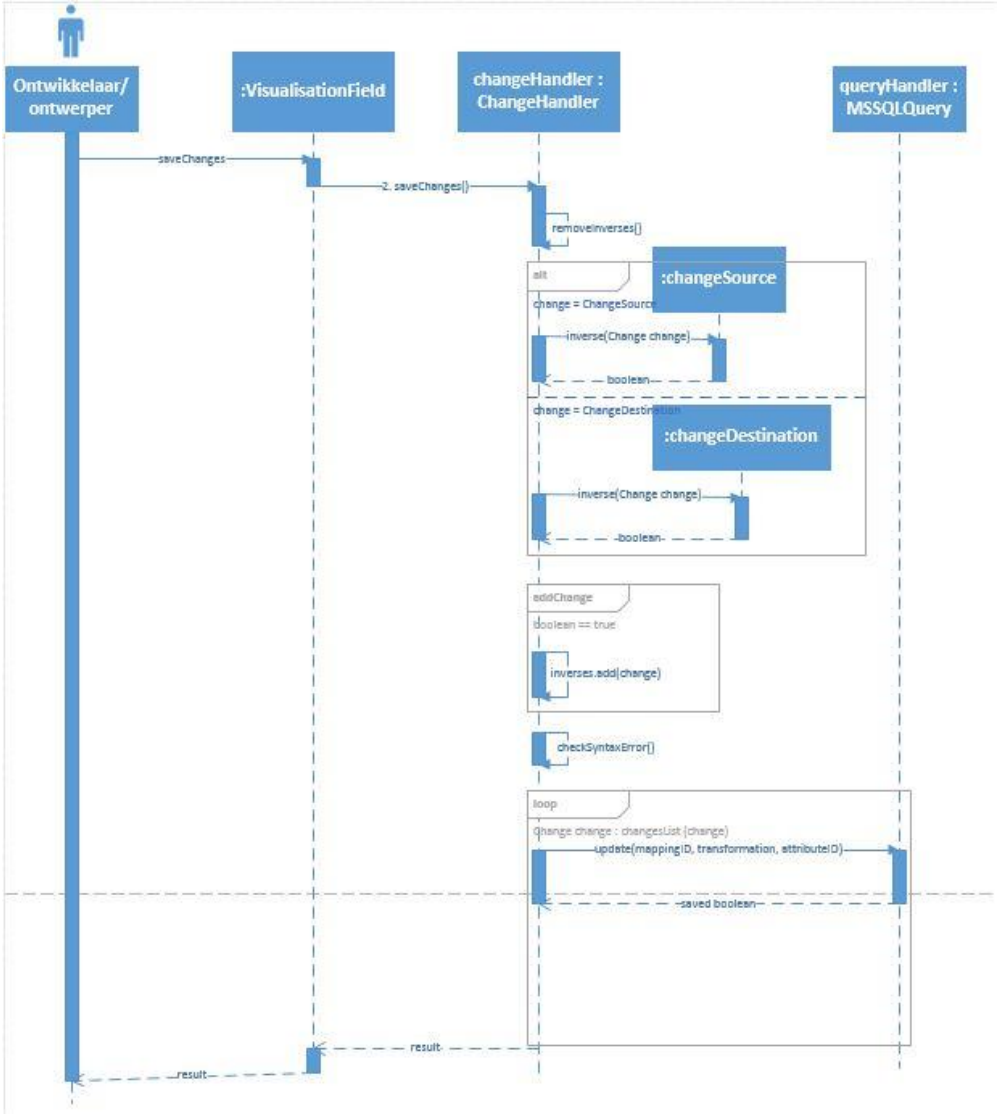
In de createDestination methode wordt een controle gedaan. Als de lijst met destinationAttributes leeg is wordt van de meegegeven bron, bewerking en bestemming een DestinationAttribute gemaakt. Deze wordt toegevoegd direct toegevoegd aan de destinationAttribute lijst van de data flow. Als de lijst niet leeg is wordt gecontroleerd of de meegegeven bewerking en bestemmingsattribuut overeenkomen met een bestaand destinationAttribute bewerking en bestemmingsattribuut. Als de bewerking en bestemmingsattribuut beiden overeenkomen is de bronattribuut onderdeel van een data flow naar een bekend bestemmingsattribuut. De controle wijst uit of regel 2 en 4 van een data flow aan de orde zijn.



Als er een destinationAttribute wordt gevonden in de lijst van de data flow die overeenkomt met de bewerking en bestemming, dan wordt het bronattribuut toegevoegd aan de bronattributenlijst van het destinationAttribute. Als er geen overeenkomstig destinationAttribute wordt gevonden dan wordt van het bronattribuut, de bewerking en bijbehorende bestemmingsattribuut een nieuw destinationAttribute gemaakt. Deze wordt daarna direct toegevoegd aan de data flow lijst.

Nadat alle mappings van de opgehaalde mappingSet(s) zijn toegevoegd wordt de lijst met destinationAttributes naar de presentation package gestuurd. Deze maakt op basis van de methoden uit de Netbeans Visual library widget objecten die kunnen worden getekend. Omdat de data flow een lijst terug geeft met destinationAttribute objecten kan er van ieder bestemmingsattribuut een data flow worden getekend. Deze objecten bevatten namelijk een bestemmingsattribuut met de bewerking en de lijst van bronattributen waaruit de bestemming wordt gevuld.

3.5.3 saveChanges()



Testplan

Data flow visualisatie tool

Titel	Testplan
Project/Onderwerp	Data flow visualisatie tool
Versie	1.1
Status	Internal Draft
Datum	20-mei-2015
Bedrijf	Info Support B.V.

Historie

Versie	Status	Datum	Auteur	Verandering
1.0	Eerste definitieve versie	27-04-2015	Maarten Koene	Eerste versie
1.1	Tweede versie applicatie	20-05-2015	Maarten Koene	Nieuwe use case en testcases toegevoegd uit de tweede elaboration fase 2

Inhoudsopgave

1. Opdrachtformulering 4

1.1 Trajectnaam 4

1.2 Beoogd resultaat 4

1.3 Opdrachtgever 4

1.4 Opdrachtnemer 4

1.5 Beschouwingsgebied 4

1.6 Testsoort 4

2. Testbasis 5

2.1 Inleiding 5

2.2 Documenten 5

2.3 Applicatie 5

2.4 Testware 5

3. Teststrategie 6

3.1 Omschrijving testaanpak 6

3.2 Risicoanalyse 6

3.3 Kwaliteitsattributen 6

3.4 Gebruikte testtechnieken en benodigdheden 8

3.4.1 Afvinklijst en checklist 8

3.4.2 Use Case test 8

3.4.3 Unit tests 8

4. Testrapport 9

4.1 Afvinklijst 9

4.2 Vragenlijst 9

4.3 Use Case test 11

4.3.1 Logische testcases 11

4.3.2 Fysieke testcases 15

5. Gecontroleerde in en uitvoer metadata 22

6. Vrijgave advies 24

7. Bronnen 25

1. Opdrachtformulering

1.1 Trajectnaam

De data flow visualisatie tool voor de metadata uit ISMetadata.

1.2 Beoogd resultaat

De metadata uit de ISMetadata database moet visueel kunnen worden benaderd. Deze metadata beschrijft de data flow van attritbuten tussen databases. Ontwikkelaars en ontwerpers willen een overzicht hebben van deze data flow en deze data kunnen aanpassen. Hiervoor wordt de data flow visualisatie tool ontwikkeld.

1.3 Opdrachtgever

Thomas van Bilsen

1.4 Opdrachtnemer

Maarten Koene

1.5 Beschouwingsgebied

Het testtraject wordt uitgevoerd op de data flow visualisatie tool. Deze applicatie zal worden getest aan de hand van een usability test. Hierbij wordt vastgesteld of de applicatie de juiste functionaliteiten bevat aan de hand van use cases, of de visualisatie tool naar wens werkt en werkt de applicatie prettig. Op deze wijze kan de gebruiksvriendelijkheid worden getest. Ook wordt de werking van de use cases bevestigd door middel van testcases. Van de logische testcases worden fysieke testcases gemaakt. Deze worden uitgevoerd in de usability tests. Verder zorgen unit tests in de applicatie voor het vaststellen van de interne werking. Ook wordt de visualisatie met verwachte uitvoer getest.

1.6 Testsoort

De gekozen testsoort is: usabilty test inclusief use case tests en unit tests

2. Testbasis

2.1 Inleiding

In dit hoofdstuk worden de documenten en software waarop de tests worden uitgevoerd naar voren gebracht. Dit wordt gedaan door eerst de documenten die als basis dienen naar voren te brengen. Deze documenten beschrijven de applicatie op functioneel en technisch niveau. Verder wordt (als dit mogelijk is) eerdere testresultaten en testvormen naar voren gebracht.

2.2 Documenten

Documentnaam	Versie	Status	Opmerking
Requirements rapport	1.3	Klaar	-
Functioneel ontwerp	1.2	Klaar	-
Technisch ontwerp	1.2	Klaar	-

2.3 Applicatie

Applicatienaam	Versie	Build	Opmerkingen
IS DataFlow visualisation tool	1.1	19-05-2015	Deze bevat interne Junit tests

2.4 Testware

Testware	Versie	Status	Opmerkingen
JUnit tests	1.0	klaar	Geïntrigeerd in de data flow visualisatie applicatie

3. Teststrategie

3.1 Omschrijving testaanpak

De IS DataFlow visualisation tool is een nieuw stuk software. De stakeholder heeft van te voren zijn eisen en wensen geuit. Hierna zijn de requirements op prioriteit gezet. De requirements met de hoogste prioriteit zijn uitgewerkt in de IS DataFlow visualisation tool. Voor de gebruikers is het van belang dat de use cases, met requirements in de hoogste prioriteit, naar wens werken. Hiervoor moeten de use cases worden gewaarborgd. Verder moet de gebruiker tevreden zijn met de werking en het uiterlijk van de applicatie.

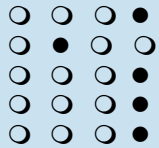
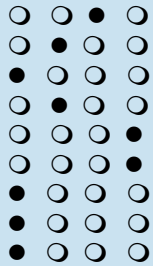
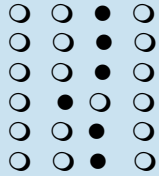
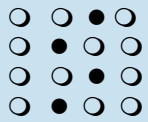
3.2 Risicoanalyse

Om de belangrijkste use cases te bepalen is er een risicoanalyse uitgevoerd. Deze is gedaan met de opdrachtgever, deze is tevens de belangrijkste stakeholder. Hieruit is het volgende naar voren gekomen:

Risicocategorie	Risicogebied	Relatief belang
Kritisch	Kiezen systeem, bekijken data flow,	10
Hoog	Aanpassen data flow	8
Middel		6
Laag		4

3.3 Kwaliteitsattributen

Kwaliteitsattributen	Omschrijving	Relatief Belang H M L G
Functionaliteit • Suitability (Geschiktheid) • Accuracy (Accuratesse) • Interoperability (Interoperabiliteit) • Compliance (Conformiteit) • Security (Beveiliging) • Traceability (Traceerbaarheid)	Mate waarin het systeem functioneel correct gedrag vertoont. Het gaat om de aanwezigheid van functies en hun gespecificeerde eigenschappen.	● ❍ ❍ ❍ ● ❍ ❍ ❍ ❍ ❍ ❍ ● ❍ ❍ ❍ ● ❍ ❍ ● ❍ ❍ ❍ ● ❍

Betrouwbaarheid • Maturity (Volwassenheid) • Fault tolerance (Fouttolerantie) • Recoverability (Herstelbaarheid) • Availability (Beschikbaarheid) • Degradability (Degradeerbaarheid)	De mate waarin het systeem blijft functioneren onder vastgestelde voorwaarden tijdens een vastgestelde periode	
Bruikbaarheid • Understandability (Begrijpelijkheid) • Learnability (Leerbaarheid) • Operability (Gebruiksgemak) • Explicitness (Explicietheid) • Customisability (Aanpasbaarheid) • Attractivity (Aantrekkelijkheid) • Clarity (Duidelijkheid) • Helpfulness (Behulpzaamheid) • User-friendliness (Gebruiksvriendelijkheid)	Mate waarin het systeem geschikt is voor gebruik.	
Efficiëntie • Time behaviour (Tijdsgedrag) • Resource behaviour (Resourcegedrag)	Mate waarin het systeem presteert. Uitgedrukt in de transactiesnelheid en de gebruikte capaciteit bij een vastgestelde belasting.	
Onderhoudbaarheid • Analysability (Analyseerbaarheid) • Changeability (Veranderbaarheid) • Stability (Stabiliteit) • Testability (Testbaarheid) • Manageability (Beheersbaarheid) • Reusability (Herbruikbaarheid)	Het gemak waarmee het systeem kan worden gewijzigd, of de inspanning die nodig is voor het maken van bepaalde wijzigingen.	
Overdraagbaarheid • Adaptability (Aanpasbaarheid) • Installability (Installeerbaarheid) • Conformance (Conformance) • Replaceability (Vervangbaarheid)	De gemak waarmee de software van de ene omgeving naar een andere omgeving over te dragen is.	

3.4 Gebruikte testtechnieken en benodigdheden

De focus van de tests moeten volgens de kwaliteitsattributen voornamelijk liggen op bruikbaarheid en functionaliteit. Om deze kwaliteitsattributen te kunnen testen worden de afvinklijst, een checklist en een use case test uitgevoerd. Hiermee worden de kwaliteitsattributen: bruikbaarheid en functionaliteit afgedekt

3.4.1 Afvinklijst en checklist

Voor het vaststellen van de bruikbaarheid zal er gebruik worden gemaakt van een afvinklijst. Deze afvinklijst wordt uitgebreid met een aantal vragen. Deze vragen moeten vaststellen of de functionaliteiten, de weergave in de interface, leesbaarheid van de visualisatie en duidelijkheid van de visualisatie correct zijn. Deze vragen worden gesteld aan de opdrachtgever, de stakeholder, van het project. Hij geeft een representatieve mening over de bruikbaarheid van de IS Data Flow visalisation tool. De vragen zijn opgesteld zodat de requirements aan bod worden gebracht in combinatie met de bruikbaarheid van de applicatie.

Ook wordt een lijst met requirements afgevinkt. Hierbij wordt per requirement gekeken of deze is meegenomen in het programma. Hierbij wordt dus vastgesteld of de requirement duidelijk terug is te vinden in de applicatie. Verder kan er per requirements opmerkingen worden geplaatst. Deze opmerkingen kunnen als basis dienen voor het requirements rapport van elaboration 2.

3.4.2 Use Case test

Verder blijkt uit de kwaliteitsattributen analyse dat functionaliteit een belangrijk attribuut is. Ook is gebleken uit de risico analyse dat de use cases van kritisch belang zijn. Het is daarom belangrijk om de functionaliteiten te testen. Om de functionaliteit vast te stellen wordt er gebruik gemaakt van een Use Case test. Met deze use case test worden de belangrijke use cases van de applicatie logisch en fysiek uitgewerkt. Hierna wordt door een stakeholder gevraagd om de use case te doorlopen. Op deze wijze kan ook de bruikbaarheid verder worden vastgesteld. Verder worden de use cases doorlopen door de tester, Maarten Koene. Hierbij wordt van iedere use case stap bekeken of het verwachte resultaat naar voren komt.

3.4.3 Unit tests

In de data flow visualisatie zijn unit tests aanwezig. Deze unit tests worden uitgevoerd en moeten voor de volle 100% slagen. Ook unit tests uit voorgaande releases moeten worden uitgevoerd als onderdeel van de regressietest. Hiermee wordt de interne werking van de data flow visualisatie tool gewaarborgd.

4. Testrapport

4.1 Afvinklijst

Requirement ID	Requirement beschrijving	Mogelijk/Gedaan	Opmerkingen
UR 1	Er kan een systeem worden gekozen uit de ISMetadata database om te visualiseren	✓	
UR 2	De data flow moet op attribuut niveau worden gevisualiseerd	✓	
UR 3	De brontabellen moeten worden kunnen herleid uit de visualisatie	✓	
UR 4	De bestemmingstabellen moeten worden kunnen herleid uit de visualisatie	✓	
UR 5	Attributen worden weergegeven in de data flow met een rechthoek	✓	
UR 6	De flow van een attribuut moet kunnen worden aangepast	✓ X	Wijzigingen mogen niet automatisch teruggedraaid worden De gemaakte wijzigingen conflict foutmelding afhandelen Bestemmingen zonder bron laten zweven
UR 8	De aggregatie bewerkingen kunnen worden getoond	✓	
UR 19	Per tabelnaam moet het attribuut een unieke kleur krijgen	X	Geen kleuren gegeven aan de attributen
UR 20	Van het attribuut moet de databasenaam naar voren komen in een wolkje als er met de muis over het attribuut wordt gegaan	✓	

4.2 Vragenlijst

Is het duidelijk hoe het systeem werkt?

- Verslepen alleen via vierkantje bij pijl is niet zo duidelijk
- Vervelend dat wijzigingen bij error ongedaan gemaakt worden

Is het duidelijk op welke wijze een database systeem geselecteerd kan worden?
Ja

Is het duidelijk uit welke tabel het bronattribuut komt?
Ja

Is het duidelijk uit welke tabel het bestemmingsattribuut komt?
Ja

Is het duidelijk uit welke database het bronattribuut komt?
Ja

Is het duidelijk uit welke database het bestemmingsattribuut komt?
Ja

Is het wolkje bij muishover duidelijk te vinden en leesbaar?
Ja, graag aangeven dat het over Systeem gaat.

Is het duidelijk wat de bewerking inhoudt?
Ja, maar later graag via icoontje.

Is het duidelijk op welke wijze een aanpassing kan worden gedaan?
Ja, maar zie boven.

Is het duidelijk op welke manier de aanpassingen kunnen worden opgeslagen?
Met de save knop!
Maar liever zonder save knop.

Zijn de afgesproken regels van een data flow terug te vinden in de applicatie?

a. Een bestemmingsattribuut heeft altijd een bewerking, ook al is deze bewerking leeg
b. Een bestemmingsattribuut kan ontstaan uit meerdere bronattributen. Voor een bronattribuut moet de naam, tabel en database en bijbehorende bewerking overeenkomen om dit vast te stellen
c. Een bestemmingsattribuut kan ontstaan zonder bronattributen te hebben
d. Een bewerking kan meerdere bronattributen bewerken
e. Een bronattribuut kan meerdere bestemmingen hebben, deze worden als een nieuwe data flow gezien
f. Een bronattribuut kan geen data flow hebben naar een ander bronattribuut
g. Bij het aanpassen kan een bronattribuut alleen worden gekoppeld aan een bewerking
h. Bij het aanpassen kan een bewerking alleen worden gekoppeld aan een bestemmingsattribuut
i. Als een aanpassing ongedaan wordt gemaakt moet er niets worden opgeslagen

Allemaal Ja
Hoe prettig werkt de applicatie op een schaal van 1 tot 10? (10 is hierbij het prettigst)
7

4.3 Use Case test

4.3.1 Logische testcases

Logische testcase 1 1: Kiezen systeem Main flow	Uitzondering	Gewenst resultaat	Werkelijk resultaat	Opmerkingen
1. Actor start de visualisatie tool		Systeem haalt de lijst met beschikbare systemen op.	Systeem haalt de lijst met beschikbare systemen op.	
		Systeem toont de lijst met beschikbare systemen.	Systeem toont de lijst met beschikbare systemen.	
4.Actor selecteert het gewenste systeem		Systeem haalt de attributen met bijbehorende bewerkingen en bestemmingsattributen van het geselecteerde systeem op	Systeem haalt de attributen met bijbehorende bewerkingen en bestemmingsattributen van het geselecteerde systeem op	

Logische testcase 2 1: Kiezen systeem Main flow	Uitzondering	Gewenst resultaat	Werkelijk resultaat	Opmerkingen
1. Actor start de visualisatie tool		Systeem haalt de lijst met beschikbare systemen op.	Systeem haalt de lijst met beschikbare systemen op.	
	[1] Geen beschikbare systemen	Systeem toont een lege systemen lijst	Systeem toont een lege systemen lijst	
4.Actor kan geen systeem selecteren		Actor ziet een lege lijst en kan geen systeem selecteren	Actor ziet een lege lijst en kan geen systeem selecteren	

Logische testcase 3: Bekijken data flow				
Main flow	Uitzondering	Gewenst resultaat	Werkelijk resultaat	Opmerkingen
1.actor doorloopt use case kiezen systeem succesvol.		Systeem haalt de gegevens van het geselecteerde systeem op	Systeem haalt de gegevens van het geselecteerde systeem op	
		Systeem toont de data flow van het systeem van de geselecteerde systeem. Waarbij de attributen uit de brontabellen, de attributen uit het bestemmingsmodel, aggregatie bewerkingen en de flow van de data tussen bron en bestemming. Ook hebben de attributen een kleur gegeven naar tabelnaam	Systeem toont de data flow van het systeem van de geselecteerde systeem. Waarbij de attributen uit de brontabellen, de attributen uit het bestemmingsmodel, aggregatie bewerkingen en de flow van de data tussen bron en bestemming. Attributen hebben geen kleur gekregen	

Logische testcase 4: Bekijken data flow Main flow	Uitzondering	Gewenst resultaat	Werkelijk resultaat	Opmerkingen
1.actor doorloopt use case kiezen systeem succesvol.		Systeem haalt de gegevens van het geselecteerde systeem op	Systeem haalt de gegevens van het geselecteerde systeem op	
	Gekozen systeem heeft geen data flow	Systeem toont een lege data flow	Systeem toont een lege data flow	

Logische testcase 5: Aanpassen data flow Main flow	Uitzondering	Gewenst resultaat	Werkelijk resultaat	Opmerkingen
1.Actor selecteert de data flow pijl van het attribuut		Systeem geeft de versleeppunten weer	Systeem geeft de versleeppunten weer	
3. Actor selecteert het pijluiteinde dat versleept dient te worden en versleept deze naar een nieuwe transformatie		Pijl verschuift mee met de muis	Pijl verschuift mee met de muis	
4.Actor zet de pijl op de nieuwe transformatie		Pijl zit vast aan de nieuwe transformatie	Pijl zit vast aan de nieuwe transformatie	
5.Actor klikt op opslaan		Het systeem slaat de wijziging op	Het systeem slaat de wijziging op	De data flow refreshed na opslaan dus de data flow verandert een van het origineel

Logische testcase 6: Aanpassen data flow Main flow	Uitzondering	Gewenst resultaat	Werkelijk resultaat	Opmerkingen
1.Actor selecteert de data flow pijl van het attribuut		Systeem geeft de versleeppunten weer	Systeem geeft de versleeppunten weer	
3. Actor selecteert het pijluiteinde dat versleept dient te worden en versleept deze naar een nieuwe transformatie		Pijl verschuift mee met de muis	Pijl verschuift mee met de muis	
4.Actor zet de pijl op de nieuwe transformatie	[1] geen transformatie geselecteerd	Systeem slaat wijziging niet op	Systeem slaat wijziging niet op	De pijl zit vast aan het oorspronkelijke transformatie
5. Actor klikt op opslaan	Het systeem geeft de melding dat er geen aanpassingen zijn om op te slaan	Het systeem geeft de melding dat er geen aanpassingen zijn om op te slaan	Het systeem geeft de melding dat er geen aanpassingen zijn om op te slaan	

Logische testcase 7: Aanpassen data flow Main flow	Uitzondering	Gewenst resultaat	Werkelijk resultaat	Opmerkingen
1.Actor selecteert de data flow pijl van het attribuut		Systeem geeft de versleeppunten weer	Systeem geeft de versleeppunten weer	
3. Actor selecteert het pijluiteinde dat versleept dient te worden en versleept deze naar een nieuwe transformatie		Pijl verschuift mee met de muis	Pijl verschuift mee met de muis	
4.Actor zet de pijl op de nieuwe transformatie	[2] bronattribuut of bestemmingsattribuut geselecteert	Systeem slaat wijziging niet op	Systeem slaat wijziging niet op	De pijl zit vast aan het oorspronkelijke transformatie
5. Actor klikt op opslaan	Het systeem geeft de melding dat er geen aanpassingen zijn om op te slaan	Het systeem geeft de melding dat er geen aanpassingen zijn om op te slaan	Het systeem geeft de melding dat er geen aanpassingen zijn om op te slaan	

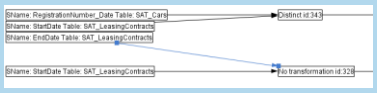
4.3.2 Fysieke testcases

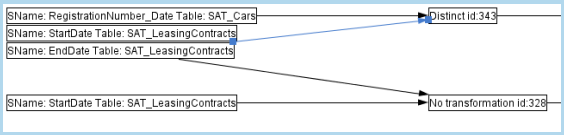
Fysieke testcase 1:	**Kiezen systeem**		
Naam tester:	**Maarten Koene**		
Pre-conditie:	-		
Requirements:	UR1		
Main Flow	Verwacht resultaat	Werkelijk resultaat	Opmerkingen
1. Actor start de visualisatie tool door dubbel te klikken op het icoontje	Het systeem toont een lijst met de volgende systemen: LeaseCarFleet_NL LeaseRelation_NL SA_LeaseCarFleet_NL SV_LeaseCarFleet_NL SA_LeaseRelation_NL BILEASE_DWH	Het systeem toont een lijst met de volgende systemen: LeaseCarFleet_NL LeaseRelation_NL SA_LeaseCarFleet_NL SV_LeaseCarFleet_NL SA_LeaseRelation_NL BILEASE_DWH	
4.Actor selecteert LeaseCarFleet_NL	Het systeem haalt de attributen, bewerkingen en bestemmingsattributen op uit de database	Het systeem haalt de attributen, bewerkingen en bestemmingsattributen op uit de database	

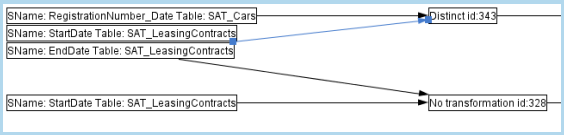
Fysieke testcase 2:	**Kiezen systeem**		
Naam tester:	**Maarten Koene**		
Pre-conditie:	-		
Requirements:	UR1		
Main Flow	Verwacht resultaat	Werkelijk resultaat	Opmerkingen
[1] Actor start de visualisatie tool door dubbel te klikken op het icoontje	Systeem toont een lege systemen lijst	Systeem toont een lege systemen lijst	

Fysieke testcase 3:	Bekijken data flow		
Naam tester:	**Maarten Koene**		
Pre-conditie:	-		
Requirements:	UR2, UR3, UR4, UR5, UR8, UR19, UR20		
Main Flow	Verwacht resultaat	Werkelijk resultaat	Opmerkingen
1.actor doorloopt use case kiezen systeem	Het systeem haalt het gekozen systemen op	✓	
4.Actor selecteert BILease_DWH	Het systeem haalt de onderstaande lijst van attributen met hun bewerkingen en bestemmingen met voor iedere tabel een unieke kleur op: 	✓	Geen kleur. Zie screenshot 1 onderaan deze pagina

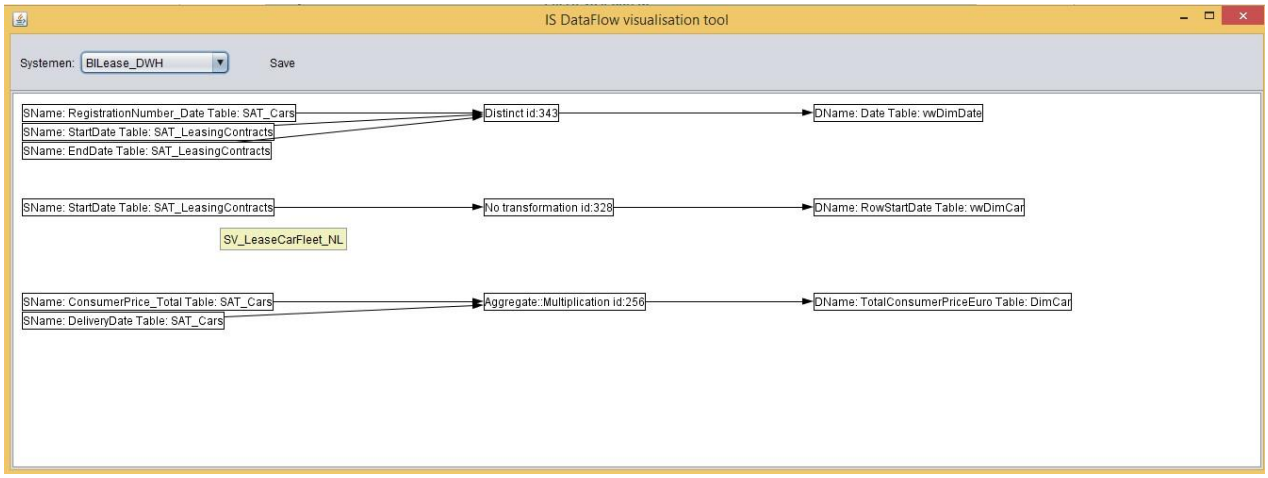
Fysieke testcase 4:	Bekijken data flow		
Naam tester:	**Maarten Koene**		
Pre-conditie:	-		
Requirements:	UR2, UR3, UR4, UR5, UR8, UR19, UR20		
Main Flow	Verwacht resultaat	Werkelijk resultaat	Opmerkingen
1.actor doorloopt use case kiezen systeem	Het systeem haalt het gekozen systemen op	✓	
4.Actor selecteert LeaseRelation_NL	Er wordt een lege data flow getoond.	✓	Er wordt een lege data flow getoond Zie screenshot 2

Fysieke testcase 5:	Aanpassen data flow		
Naam tester:	**Maarten Koene**		
Pre-conditie:	Er is een systeem met een data flow gekozen (BILease_DWH)		
Requirements:	UR6		
Main Flow	Verwacht resultaat	Werkelijk resultaat	Opmerkingen
1.Actor selecteert de data flow pijl van EndDate		✓	
3. Actor selecteert het pijluiteinde van de transformatie "Distinct"		✓	
4.Actor zet de pijl op "No transformation"		✓	
5.Actor klikt op opslaan	Het systeem toont de melding: De "wijzigingen zijn opgeslagen"	✓	De data flow refreshed na opslaan dus de data flow verandert een van het origineel

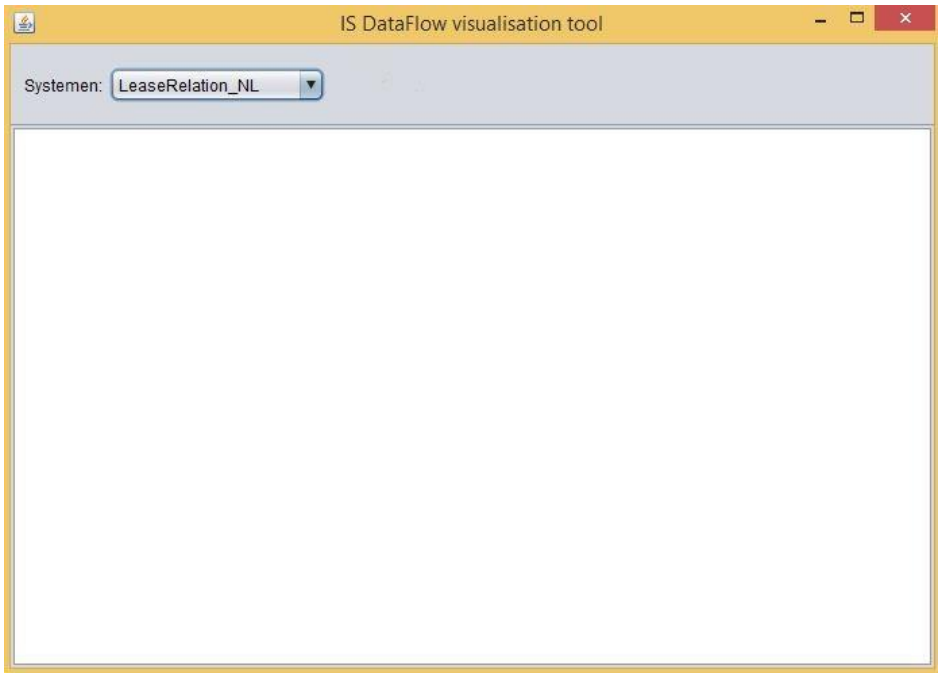
Fysieke testcase 6:	Aanpassen data flow		
Naam tester:	**Maarten Koene**		
Pre-conditie:	Er is een systeem met een data flow gekozen (BILease_DWH)		
Requirements:	UR6		
Main Flow	Verwacht resultaat	Werkelijk resultaat	Opmerkingen
1.Actor selecteert de data flow pijl van StartDate		✓	
3. Actor selecteert het pijluiteinde van de transformatie "Distinct"		✓	
4.Actor zet de pijl op een wit stuk achtergrond		✓	De pijl zit vast aan het oorspronkelijke transformatie
5. Actor klikt op opslaan	Het systeem geeft de melding dat er geen aanpassingen zijn om op te slaan	✓	

Fysieke testcase 7:	Aanpassen data flow		
Naam tester:	**Maarten Koene**		
Pre-conditie:	Er is een systeem met een data flow gekozen (BILease_DWH)		
Requirements:	UR6		
Main Flow	Verwacht resultaat	Werkelijk resultaat	Opmerkingen
1.Actor selecteert de data flow pijl van StartDate		✓	
3. Actor selecteert het pijluiteinde van de transformatie "Distinct"		✓	
4.Actor zet de pijl op de bestemmingsattribuut "Date"		✓	De pijl zit vast aan het oorspronkelijke transformatie
5. Actor klikt op opslaan	Het systeem geeft de melding dat er geen aanpassingen zijn om op te slaan	✓	

Screenshot 1:

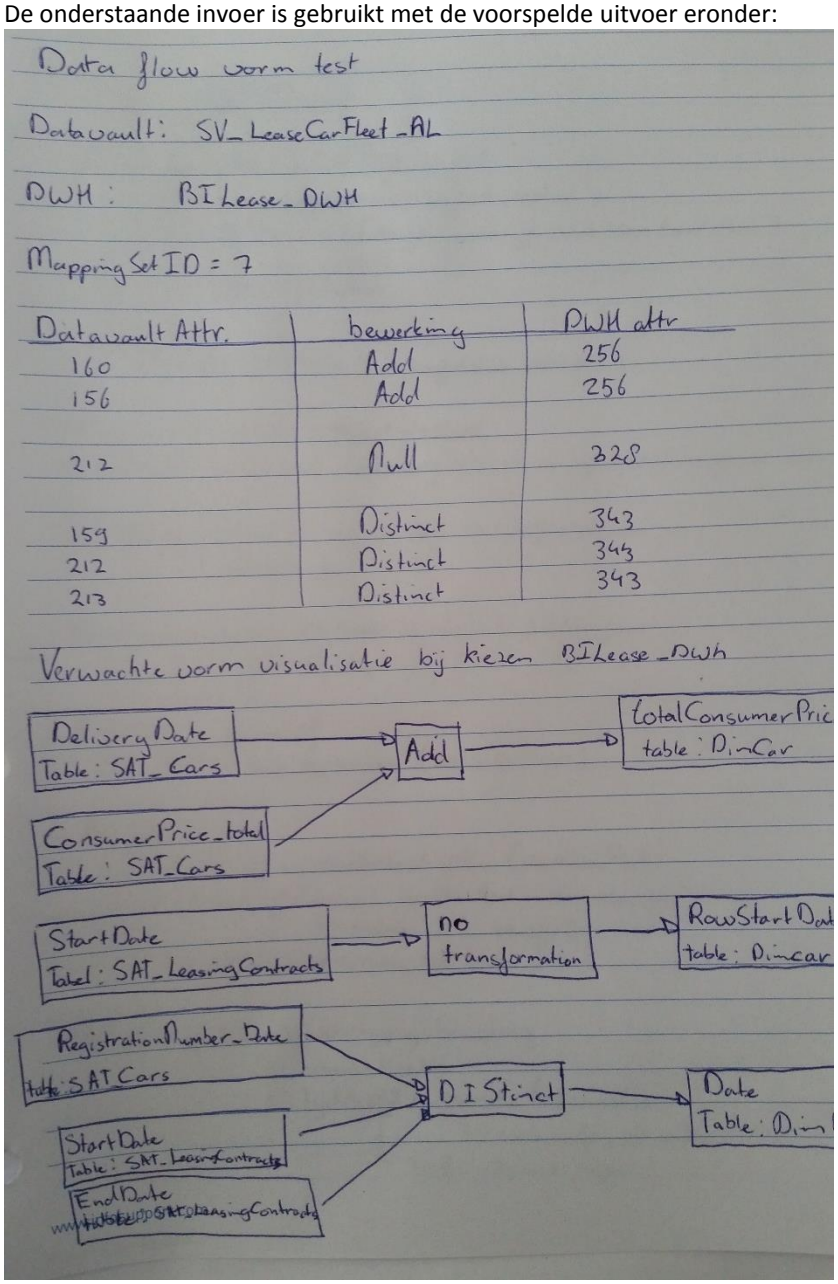


Screenshot 2:

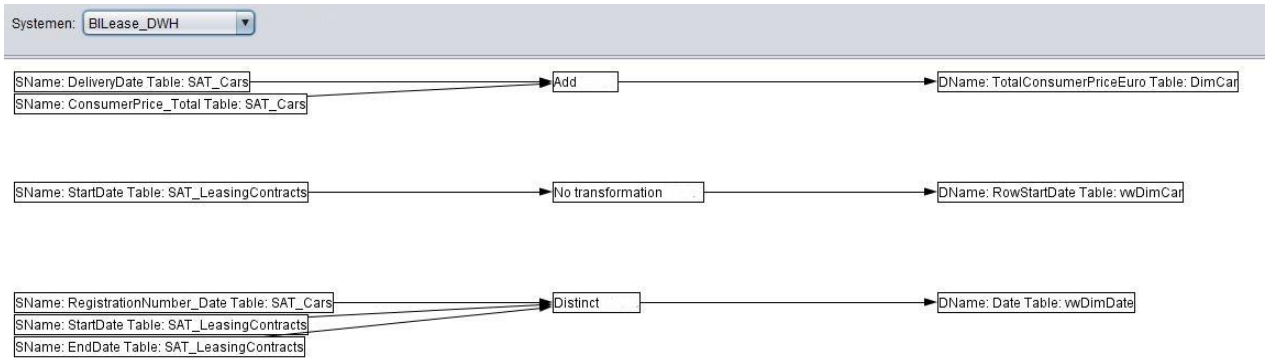


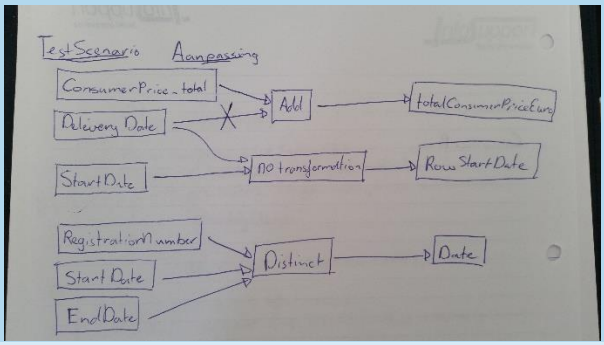
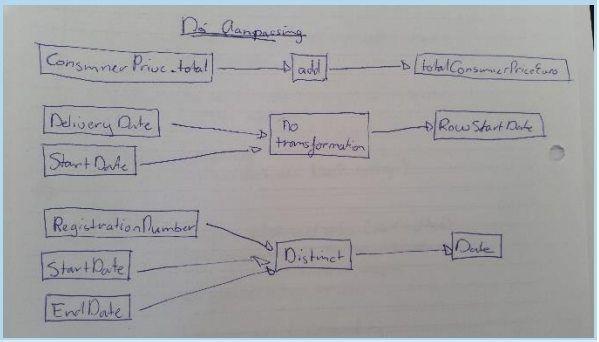
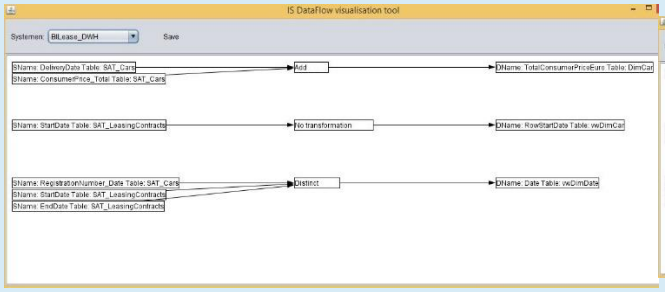
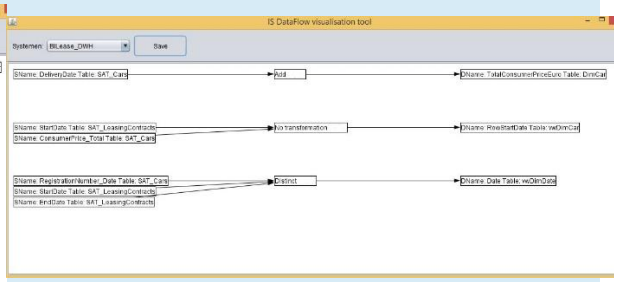
5. Gecontroleerde in en uitvoer metadata

De onderstaande invoer is gebruikt met de voorspelde uitvoer eronder:



De werkelijke uitvoer is dan ook als volgt geproduceerd door de data flow visualisatie tool en is dus geslaagd voor deze test.



Visualisatie voor aanpassing	Situatie na aanpassing
	
	

6. Vrijgave advies

Advies: positief

In de tweede versie van de data flow visualisatie tool kan de data flow worden aangepast. De voorgaande twee kritische use cases is de werking van behouden gebleven en succesvol door de regressie test heen gekomen. Het is nu ook mogelijk om de data flow van een attribuut te veranderen en op te slaan.

Echter zijn er uit de gebruikers acceptatie test een aantal belangrijke opmerkingen gemaakt. Door deze opmerkingen uit te werken ontstaat er een gebruiksvriendelijkere situatie. Zo moeten de volgende eisen verwerkt zijn in de volgende release van de data flow visualisatie tool:

- Bij een syntax fout mogen de gemaakte wijzigingen niet automatisch ongedaan gemaakt worden
- Bij het aanpassen data flow waarbij de laatste bronattribuut wordt losgekoppeld van de bewerking moet de bewerking en bestemmingsattribuut in de visualisatie blijven staan en dus worden getoond zonder bronattributen
- Er moeten duidelijke kleuren worden toegekend aan de bron-en bestemmingsattributen. De kleur is uniek voor iedere tabelnaam.

Er is echter wel een werkbare situatie ontstaan. Hierdoor is het vrijgave advies positief. Wel moeten de bovenstaande problemen worden verholpen zodat de eerste release in de productieomgeving deze functionaliteiten wel bevat.

7. Bronnen

http://www.softwaretesten.net/testmethodieken/tmap-next-te/samenvatting-tmap-next-hoofdstuk-14/
http://nl.wikipedia.org/wiki/Acceptatietest
http://www.examenvragen.info/tmap-next/tmap-samenvatting/tmap-kwaliteitsattributen-testvormen.html