AFSTUDEERVERSLAG

STANDAARDISEREN VAN DE UITWISSELING TUSSEN PLURIFORM GOEDE DOELEN EN EEN KLANT WEBSITE DOOR MIDDEL VAN EEN GENERIEK SET XML BERICHTEN

AUTEUR: DAVID ROOS

STUDENTNUMMER: 07074786

OPLEIDING: INFORMATICA

ONDERWIJSINSTELLING: DE HAAGSE HOGESCHOOL

EXAMINATOREN: H.G.J BECHET-TJOONK EN W. VAN VLIET

AFSTUDEERPERIODE: 7 FEBRUARI 2011 – 23 SEPTEMBER 2011

BEDRIJF: MATTHAT SOFTWARE B.V.

ADRES: STATIONSPLEIN 9-G, GOUDA

BEGELEIDER: L.M. ROELEVELD

DATUM: 23 SEPTEMBER 2011

DOCUMENTVERSIE: 1.0

REFERAAT

David Roos, Afstudeerverslag, Standaardiseren van de uitwisseling tussen Pluriform Goede Doelen en een klantwebsite door middel van een generiek set XML berichten, Matthat Software B.V., Gouda, 23 september 2011.

Dit verslag is geschreven in het kader van het afstuderen voor de opleiding Informatica aan de Haagse Hogeschool. Het verslag beschrijft het proces dat de student heeft doorlopen tijdens het uitvoeren van de afstudeeropdracht bij Matthat Software B.V. en heeft als doel aantonen dat de student de inhouden en competenties van de opleiding Informatica beheerst.
Het afstudeertraject is gestart op 7 februari 2011 en geëindigd op 23 september 2011 en heeft een doorlooptijd van 22 werkweken.

XML
SOAP
RPC
Goede doelen
Uitwisseling
Kwalitatief onderzoek
Standaardisatie
Afstudeerverslag
Unified Process
Unified Modeling Language
David Roos
Matthat Software B.V.
De Haagse hogeschool

VOORWOORD

Voor u ligt het afstudeerverslag van David Roos. Dit document heb ik opgesteld tijdens mijn afstudeerproject bij Matthat Software B.V. te Gouda. Ondanks de vele aspecten waar ik rekening mee moest houden tijdens het schrijven van dit document kan ik terugkijken op een prettige afstudeerperiode waarin ik erg goed geholpen ben en veel geleerd heb. Door het afstudeertraject te doorlopen heb ik erg veel zin gekregen om het bedrijfsleven binnen te stappen.

Allereerst hartelijk dank aan Nico Sijrier, Leen Roeleveld en Martijn van Toor, interne begeleiders bij Matthat Software B.V. die me alle vrijheid gaven om mijn opdracht uit te voeren en altijd open stonden voor vragen en feedback. Hiernaast wil ik alle andere collega's bij Matthat Software B.V. bedanken voor hun ondersteuning tijdens mijn afstuderen en de gezellige sfeer op kantoor. Ook veel dank aan mevr. H.G.J. Bechet-Tjoonk en dhr. W. van Vliet, begeleiders vanuit de Haagse Hogeschool, die mij voorzagen van waardevolle feedback.

David Roos, vrijdag 23 september 2011

INHOUDSOPGAVE

INLEIDING

Matthat Software B.V. is softwareleverancier van een informatiesysteem voor goede doelen genaamd Pluriform Goede Doelen. Dit systeem helpt goede doelen met het aanpakken van branche-specifieke uitdagingen zoals fondswerving en verantwoording, leden- en activiteitenadministratie.

In de huidige situatie maakt Pluriform Goede Doelen interactie met een website mogelijk door gegevens uit Pluriform te verpakken in XML berichten, te versturen naar de website en daar te tonen. Andersom kunnen gegevens die ingevuld worden op de website verpakt worden in een XML bericht en verstuurd naar Pluriform. Deze manier van uitwisseling is de meest toegepaste uitwisselingsmethodiek door Matthat Software B.V. bij haar klanten.

De XML berichten die aan de klant aangeboden worden zijn echter voor iedere klant op maat gedefinieerd, terwijl er een overlap is tussen de uitwisselingswensen van klanten. Vanwege dit probleem is dit project gestart. Het stelt zich als doel de uitwisselingswensen te achterhalen door middel van een onderzoek en de samenvallende wensen te gebruiken om een (generiek) set van XML berichten te ontwerpen, om zo een gestandaardiseerde communicatie interface aan de klanten van Matthat Software B.V. te kunnen aanbieden.

Het verslag is opgedeeld in tien hoofdstukken. De eerste drie hoofdstukken beschrijven het afstudeerbedrijf en zijn systeem(1), de afstudeeropdracht (2) en de aanpak van het project (3). De volgende hoofdstukken beschrijven de eerste fase van Unified Process (4), requirements (5) en het opstellen van een architectuur (6). De daaropvolgende drie hoofdstukken zijn beschreven volgens drie iteraties genaamd ontwikkelen van een Proof Of Concept (7) en ontwikkelen van een wijzigingsinterface (8) en het ontwikkelen van service operations voor XML uitwisseling (9) en het laatste hoofdstuk evalueren (10). Het laatste hoofdstuk beschrijft de product en proces evaluatie en het aantonen van het behalen van de beroepstaken.

1 BEDRIJFSORGANISATIE

In dit hoofdstuk wordt achtergrond informatie gegeven om een beeld te kunnen vormen van de werkomgeving van de student en het bedrijf van de opdrachtgever waar de student zijn opdracht doet. Om een kader te stellen voor de opdracht wordt in paragraaf 1.1 en paragraaf 1.2 de specifieke goede doelen branche en het Pluriform informatiesysteem beschreven. In paragraaf 1.3 wordt de positie van Pluriform Goede doelen beschreven en de paragrafen 1.4 en 1.5 beschrijven Pluriform Studio en de filosofie achter Pluriform.

1.1 MATTHAT SOFTWARE B.V.

Matthat Software B.V. is gevestigd in Gouda. Het bedrijf richt zich op het ontwikkelen en implementeren van een geïntegreerde oplossing voor Goede Doelen. Deze doelgroep omvat fondswervende organisaties, platformorganisaties, stichtingen en verenigingen, NGO's etc. Het product heet Pluriform Goede Doelen en omvat functionaliteit voor relatiebeheer (CRM), projecten, financiën, subsidies, leden, marketing & fondsenwerving, activiteiten en evenementen, abonnementen, inkoop, verkoop en voorraadbeheer, personeel etc. Matthat Software B.V. voert de volgende activiteiten uit:

- Acquisitie en sales van nieuwe klanten
- Implementatie van Pluriform Goede Doelen
- Ontwikkeling van nieuwe functionaliteit op basis van wensen van een klant
- Support voor Pluriform Goede Doelen
- Trainingen rond de functionaliteit in Pluriform Goede Doelen
- Hosting van Pluriform
- Applicatiebeheer van Pluriform

Matthat Software heeft 10 medewerkers in dienst en werkt structureel samen met 3 freelancers. Opgericht in 2003 zijn ze sindsdien gegroeid naar ca. 35 klanten in 2011, waaronder War Child, Liliane Fonds, Leger des Heils, Diabetes Fonds, etc.
De afstudeerder is werkzaam op de afdeling Research en Development. Op deze afdeling wordt nagedacht over innovatieve ontwikkelingen binnen Pluriform Goede Doelen en hierbij wordt er vooral software en toepassingen ontwikkeld voor klanten van Matthat Software B.V. Zie ook het organogram van Matthat Software B.V. in bijlage E.

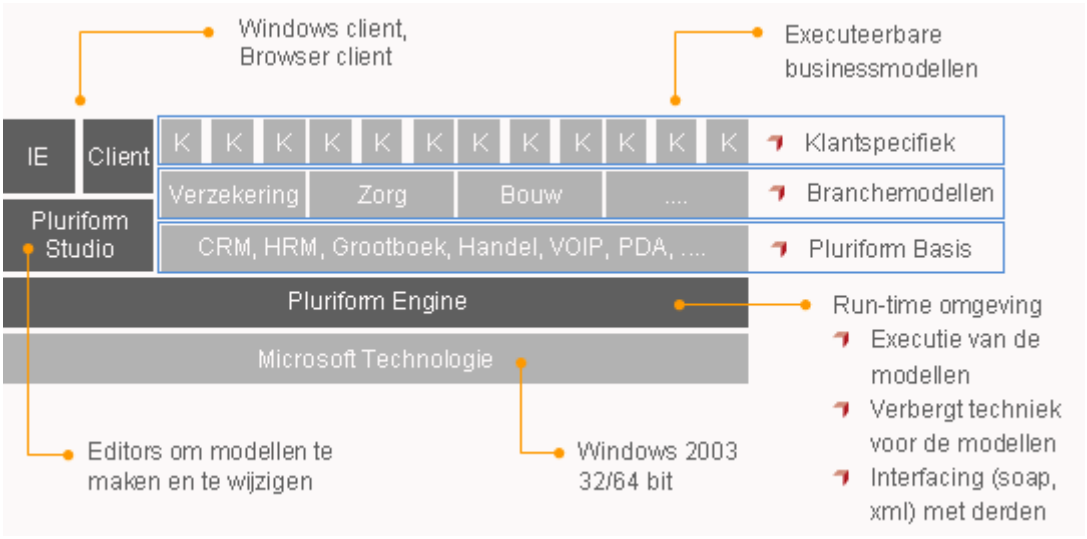
1.2 DE GOEDE DOELEN BRANCHE

Matthat Software B.V. richt zich op verschillende branches binnen de Goede Doelen sector. Hierdoor wordt Pluriform Goede Doelen ingezet bij organisaties in de volgende branches:

- Fondswervende organisaties
- Vermogensfondsen
- Jongerenorganisaties
- Inzamelings organisaties
- Gezondheidsfondsen
- Kerkelijke administraties
- Ledenorganisaties
- Ontwikkelingssamenwerkingsorganisaties

De achterban/doelgroep van deze organisaties omvat onder andere donateurs, donoren, leden, vrijwilligers en deelnemers. De organisaties in de Goede Doelen sector werken vaak samen met partners en NGO's en zijn vaak verbonden aan leveranciers. Veel van de organisaties in de Goede Doelen sector organiseren projecten voor bijvoorbeeld ontwikkelingswerk in zowel binnen- als buitenland. Om al deze partijen met elkaar samen te laten werken heeft Matthat Software B.V. voor deze specifieke sector een oplossing, hierna genoemd branchemodel, ontwikkeld (zie paragraaf 1.3).

1.3 DE POSITIE VAN PLURIFORM GOEDE DOELEN



Figuur 1 Positie van de drie lagen

Het bedrijf Pluriform Software ontwikkelt en distribueert Pluriform Basis voor en naar haar partners als AdapCare, BouwInfosys, ESI IT Solutions en Matthat Software B.V. Elk van deze partners krijgt de mogelijkheid om boven op de laag Pluriform Basis zijn eigen laag branchemodellen te ontwikkelen: dat wil zeggen een nieuwe verzameling Pluriform modellen die samen de processen in de betreffende branche ondersteunen. Matthat Software B.V. heeft daardoor voor de Goede Doelen sector Pluriform Goede Doelen ontwikkelt, wat een ERP-oplossing en een typische backoffice applicatie is voor de ondersteuning van de administratieve processen van Goede Doelen. Hiertoe hoort bijvoorbeeld de giftverwerking, subsidieaanvragen en het beheren van donateurs. Bijlage F toont een volledig overzicht van ondersteunde processen door het Pluriform Goede Doelen branchemodel. Figuur 1 toont de positie van de drie lagen (de blauw gearceerde kolommen), te weten Pluriform Basis, branchemodellen en klantspecifieke modellen, binnen de Pluriform architectuur. Om beter te begrijpen wat deze lagen inhouden worden de lagen afzonderlijk beschreven.

1.3.1 HET PLURIFORM DRIE LAGEN MODEL

Pluriform Basis
Pluriform Basis bevat de functionaliteit die ieder Pluriform informatiesysteem tot zijn beschikking heeft. Dit is algemene functionaliteit die in verschillende branches nodig is. Denk bijvoorbeeld aan relatiebeheer, financiële administratie, verkoop- en inkoopadministratie, projectadministratie en personeelsadministratie. Om snel te kunnen anticiperen op veranderende omstandigheden of inzichten, wordt dagelijks een nieuwe versie van Pluriform Basis ter beschikking gesteld aan Matthat Software B.V.

Branchemodel
Het branchemodel wordt ontwikkeld door Matthat Software B.V. Dit model omvat functionaliteit die de processen in de Goede Doelen branche ondersteunen (zie bijlage F). Zowel de functionaliteit van het branchemodel als van Pluriform Basis is beschikbaar voor de klant. Voorwaarde is dat deze functionaliteit wordt afgenomen en in de licentie staat opgenomen. Enerzijds is Matthat Software B.V. afhankelijk van wat er voor functionaliteit wordt aangeleverd en ondersteund vanuit Pluriform Basis om hierop verder te ontwikkelen, anderzijds is het mogelijk om gewenste functionaliteit zelf te ontwikkelen. Voor dit laatste is er Pluriform Studio, een geïntegreerd modelleerplatform, wat later in dit hoofdstuk aan de orde komt.

Klantmodel
De aanpak van Matthat Software B.V. is vraag gestuurd. Vaak is een vraag van een klant generiek voor de branche of zelfs generiek voor meerdere branches en wordt dan respectievelijk in een branchemodel of in de Pluriform Basis opgepakt. Als de vraag van een klant uniek is kan deze specifiek voor die klant worden

gerealiseerd. Een klantmodel wordt altijd gebruikt in combinatie met het branchemodel en Pluriform Basis. Doordat klantmodellen worden beheerd in dezelfde database als het branchemodel is Matthat Software B.V. in staat om klantspecifieke functionaliteit later indien nodig te 'verhuizen' naar het branchemodel.

1.4 PLURIFORM STUDIO

Vanuit de opleiding Informatica heb ik gewerkt met programmeerplatformen (NetBeans en Visual Studio) waarin onder andere een source code editor, een compiler, een debugger en ontwerptools voor een GUI aanwezig zijn. Pluriform Studio is geen programmeerplatform, maar een modelleerplatform. Waarom Pluriform Studio een modelleerplatform wordt genoemd en waarom een stuk software ontwikkelen binnen Pluriform Studio mij in de eerste weken wat moeite heeft gekost, zal ik uitleggen door een aantal verschillen met programmeerplatformen te noemen.

De scripttaal
Pluriform gebruikt een scripttaal waarmee bedrijfsprocessen zo gemodelleerd worden dat deze modellen, zowel leesbaar zijn voor modelleurs en kenniswerkers, als uitvoerbaar zijn door de Pluriform Engine (zie Figuur 1). Dit script is niet te vergelijken met een programmeertaal die ik ken. Het script bestaat uit elementen zoals bijvoorbeeld een try-catch en for each. Door een hulpscherm op te roepen wordt verwezen naar verschillende klassen zoals methoden, operations en generics. Als ontwikkelaar kan ik op een snelle manier gebruik maken van al bestaande methoden en dergelijke. Pluriform bevat een script element general call waarin het mogelijk door alle bestaande methoden en generics heen te lopen. Je kunt aangeven in welke context een bepaalde methode of generic moet zitten, bijvoorbeeld de klasse Persoon. Pluriform toont dan vervolgens methoden en generics die in de context klasse Persoon vallen. Daarnaast kan ik als ontwikkelaar filteren op resultaattype van een methode of generic, denk hierbij bijvoorbeeld aan void, date of amount. Door deze elementen te verbinden vormt zich een uitvoerbaar script. De manier van uitvoeren is te vergelijken met javascript, Pluriform bevat zijn eigen interpreter in de Pluriform Engine om het script uit te voeren.

Hergebruik
Pluriform heeft een bibliotheek met standaard objecten waaruit snel een nieuw model gebouwd kan worden. Modellen kunnen ook aangepast worden om ze passend te maken voor een bepaalde toepassing. Als onderdelen verbeterd zijn worden ze toegevoegd aan de bibliotheek zodat andere modellen ze kunnen hergebruiken.

Integratie
Het modelleerplatform is niet alleen een ontwikkelomgeving, het is ook de applicatie zelf. Het ontwikkelproces is volledig geïntegreerd met alle functionaliteit die Pluriform biedt. Het gehele systeem gebruikt dan ook eenzelfde GUI en kan volledig worden bediend met het toetsenbord.

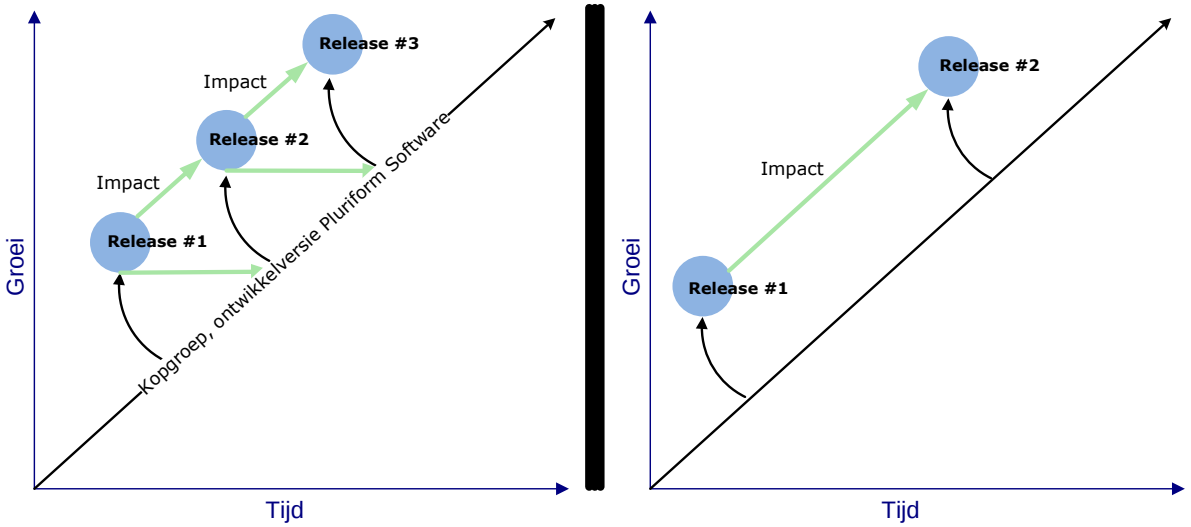
1.5 DE FILOSOFIE ACHTER PLURIFORM

Pluriform Software is het bedrijf achter Pluriform, wat veelvormig betekent. Het bedrijf zag twee kernproblemen bij softwareontwikkeling, namelijk: Het beheersen van de varianten/versies en het beheersen van veranderingen. Hierdoor is men gaan redeneren over het concept van het nieuwe platform dat men wilde ontwikkelen. Eén van de kernpunten is dat het platform een modeleeromgeving bevat waarin de gebruiker zelf nieuwe klassen moet kunnen definiëren, die vervolgens geïnstantieerd kunnen worden. De modeleeromgeving moet de klant in staat stellen zijn eigen applicatie te modelleren. De taal om dit te doen bestaat uit de kernbegrippen klasse en object. Dit concept vond zijn uitwerking in een platform wat ondersteuning van het proces van verandering faciliteert.

De uitwerking van het concept is met name terug te zien in een klasse die 'klasse' heet. Het is een object van de klasse 'klasse'. Hieruit valt af te leiden dat het model in zichzelf is uitgedrukt (de bootstrap). Het platform, geschreven in C, is de omgeving en is door de tijd heen een rijke bibliotheek geworden waaruit je de standaard bouwstenen kunt gebruiken voor het modelleren van een oneindig aantal variaties. In deze omgeving vind je de klasse 'klasse' terug die in zichzelf is uitgedrukt.

Het beheersen van veranderingen wordt mogelijk gemaakt door de groei en snoei bewegingen. Er wordt gekozen om een bestaande klasse aan te passen of een nieuwe toe te voegen. In het laatste geval van groeien geldt dat er geen discussie is of een klasse goed of fout is, want uiteindelijk zullen de beste overleven en zelfs standaard aangeboden kunnen worden. Op het moment dat meerdere klassen met dezelfde attributen bestaan en hetzelfde bedoelen worden deze samengenomen tot een nieuwe klasse met dezelfde attributen plus een context, ook wel het snoeien genoemd.

Een schaduwzijde van softwareontwikkeling is het proces van releases en afsplitsingen. Hoe blijven varianten/versies beheersbaar? Het antwoord is eenvoudig. Het softwareontwikkelproces zal gebeuren in korte iteraties. Het voordeel van korte iteraties (updates) tegenover grote iteraties zoals bijvoorbeeld een MS Windows platform is dat de mogelijke gevolgen van een iteratie een minder grote impact hebben (zie Figuur 2). In principe beschikt elke klant over een productie- en testomgeving. Na iedere korte iteratie is het mogelijk om het systeem als update in te lezen op de testomgeving van de klant die vervolgens de ontwikkelde functionaliteit zelf kan testen. Uiteindelijk wordt er een update ingelezen van het systeem op de productieomgeving.



Figuur 2 het proces van releases en afsplitsen van Pluriform tegenover MS Windows

Zowel Pluriform Software als Matthat Software B.V. splitst twee keer per jaar een stabiele versie/release af en ontwikkelt verder in de ontwikkelversie. Hierdoor hebben klanten twee mogelijkheden (zie Figuur 2):

- Van release naar release. Het voordeel hiervan is een stabiele versie, een nadeel is dat je niet beschikt over de meest recente versie.
- Meegaan in de zogeheten kopgroep/innovatierelease. Het voordeel hiervan is dat je beschikt over de meest recente versie, een nadeel kan zijn dat onderdelen in ontwikkeling niet altijd stabiel zijn. Als een klant in de kopgroep zit, kan het zijn dat er zelfs dagelijks een update van het systeem wordt ingelezen. Daarnaast is vastgesteld dat er minimaal 1 keer in de 2 weken een update wordt ingelezen bij de klant.

Naast korte iteraties hanteert zowel Pluriform als Matthat Software B.V. ondersteuning van softwarereleases tot vier releases terug (minder dan 2 jaar oud). Hierdoor blijven varianten/versies beheersbaar

2 DE AFSTUDEEROPDRACHT

Dit hoofdstuk geeft een beschrijving van de aanleiding (2.1), de uitgangssituatie (2.2), de probleemstelling (2.3) en de doelstelling (2.4) van de afstudeeropdracht zoals deze is opgegeven door Matthat Software B.V. Hiernaast wordt er een opsomming gedaan over de op te leveren producten (2.5) tijdens het afstudeerproject en mijn rol en positie (2.6).

2.1 DE AANLEIDING

Pluriform Goede Doelen maakt het mogelijk om een koppeling te maken met een website van een klant. Er zijn namelijk een aantal factoren in de Goede Doelen sector die uitwisseling van gegevens met de website meer en meer noodzakelijk maken:

1. *Informatie.* Het wordt steeds belangrijker voor een Goed Doel om de achterban/giftgevers gerichter te informeren. Donateurs geven giften maar hebben steeds meer de vraag hoe het geld wat gegeven is, wordt besteed en welke resultaten daarmee worden behaald.
2. *Efficiency.* Om een hogere efficiency te bereiken wordt de achterban/giftgever steeds vaker ingeschakeld om gegevens zelf in te voeren of te muteren. Hierbij kan gedacht worden aan het inschrijven voor evenementen, acties en vakanties, maar ook aan het doorgeven van een adresmutatie. Dit gebeurt vaak op de website van de klant waar vervolgens deze gegevens aangeboden worden aan Pluriform Goede Doelen.
3. *Transparantie.* Organisaties die giften ontvangen van het publiek moeten daarover verantwoording afleggen. Er is een beweging in de sector waarbij steeds meer financiële gegevens met betrekking tot kosten en opbrengsten wordt gepubliceerd op het web.
4. *Online fondsenwerving.* Voor fondswervende organisaties wordt het steeds belangrijker om via internet fondsen te werven. Opkomende technieken zijn vooral sociale media.

Een aantal klanten van Matthat Software B.V. maakt nu gebruik van deze web-koppeling. Deze uitwisseling van gegevens is nu echter voor iedere klant op maat gedefinieerd, terwijl er een overlap is tussen wat de uitwisselingswensen van klanten zijn. De aanleiding tot de opdracht is dan ook de vraag die rijst: hoe en wat kan er gestandaardiseerd worden met betrekking tot de uitwisseling van gegevens tussen Pluriform Goede Doelen en een website van een klant?

2.2 DE UITGANGSSITUATIE

De huidige situatie met betrekking tot de uitwisseling tussen Pluriform Goede Doelen en een website van een klant is dat Pluriform Goede Doelen de volgende interacties met de website ondersteunt:

1. *Interactie door middel van door Pluriform gegenereerde webforms*. Deze webforms communiceren via een zogenaamde web-connector rechtstreeks met de Pluriform backend. Gegevens die in deze webforms worden ingevuld kunnen dus rechtstreeks in Pluriform worden opgeslagen.
2. *Interactie met Pluriform door middel van XML berichten*. Gegevens uit Pluriform kunnen worden verpakt in XML berichten die, Remote Procedure Call (hierna genoemd RPC), naar de website worden verstuurd en daar getoond. Andersom kunnen gegevens die ingevuld worden op de website verpakt worden in een XML bericht en verstuurd naar Pluriform. De XML berichten worden in Pluriform via een ‘transformatie script’ (semi-)automatisch verwerkt.
3. *Interactie met Pluriform door batch gestuurde uitwisseling van bijvoorbeeld CSV bestanden*. Deze bestanden kunnen op een FTP locatie worden neergezet en daaruit opgepakt en verwerkt.

De meest toegepaste uitwisselingsmethodiek is optie 2, dus via XML berichten. Naast deze opties is het sinds 2010 mogelijk om Pluriform Goede Doelen in te zetten als Simple Object Access Protocol (hierna genoemd SOAP) server. Hier is, tot de start van de afstudeeropdracht, nog geen kennis of implementatie van bij Matthat Software B.V.

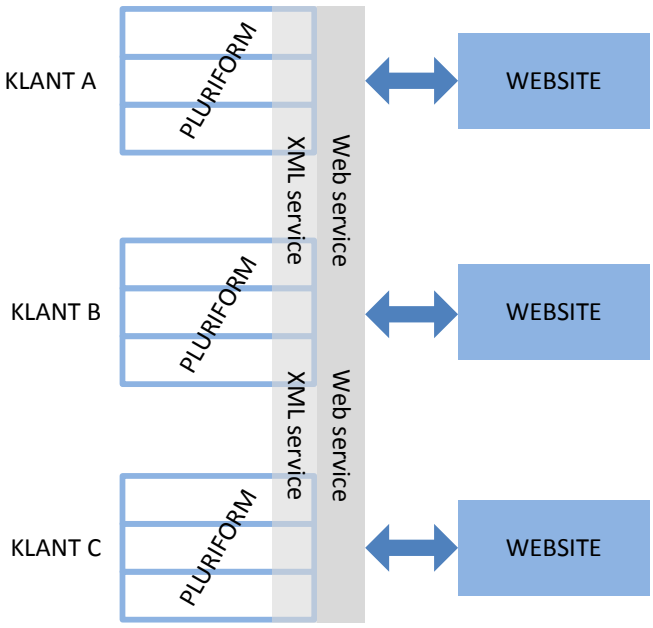
Naast uitwisseling tussen Pluriform Goede Doelen en een website is er ook een groeiende noodzaak tot het uitwisselen van informatie tussen Pluriform systemen van verschillende klanten. Een voorbeeld hiervan is dat NGO's die gezamenlijk subsidie ontvangen samen de resultaten moeten verantwoorden aan het Ministerie van Buitenlandse Zaken. Deze informatie wordt in de huidige situatie samengebracht in Excel bestanden.

2.3 DE PROBLEEMSTELLING

"Wat is er nodig om (betere, snellere, gestandaardiseerde enzovoort) uitwisseling mogelijk te maken tussen het Pluriform systeem en de klant website en eventueel verschillende Pluriform klantsystemen en hoe kan dat proces zodanig worden gestandaardiseerd dat klanten met minimale inzet van Matthat dit zelf in gebruik kunnen nemen?"

2.4 DE DOELSTELLING

Het ontwikkelen, binnen de Pluriform standaardoplossing, van generieke XML berichten die uitwisseling van gegevens tussen systeem en website mogelijk maken (zie ook Figuur 3). Hierbij zal gekeken worden naar de vereiste en beschikbare data voor fondsenwerving en kan er aan de hand van een analytisch onderzoek datasets en XML berichten worden gestandaardiseerd. Hiernaast wordt gekeken naar de mogelijkheden voor de standaardisatie van de interface die uitwisseling mogelijk maakt. Dit alles zal op basis zijn waarvan de klanten efficiënter de communicatie met hun website en eventueel met andere Pluriform systemen kunnen implementeren. De standaardisatie van deze onderdelen betekent dat ze ingezet kunnen worden bij alle klanten. Door dit aan te bieden zal deze uitwisseling niet meer als maatwerk voor iedere klant worden gemaakt, wat uiteindelijk moet leiden tot een besparing in de ontwikkel- en implementatiekosten.



Figuur 3 Tijdens de afstudeeropdracht is het doel de grijze laag te ontwikkelen

2.5 OP TE LEVEREN PRODUCTEN

Plan van Aanpak
De overeenkomst tussen de opdrachtgever en de afstudeerder. Hierin worden afspraken vastgelegd die vooraf gaan aan de opdracht inclusief een planning.

Onderzoeksplan
In het onderzoeksplan komen de volgende onderwerpen aan bod (volgens de methode uit 'wat is onderzoek? door Nel Verhoeven') (Verhoeven, 2010):

- Aanleiding
- Probleemstelling
- Doelstelling
- Afbakening
- Onderzoek ontwerp en verantwoording
- Tijdpad

Survey onderzoeks ontwerp
Een document waarin een ontwerp is opgenomen voor een vragenlijst die gericht wordt aan klanten van Matthat Software B.V.

Informatieplan inventarisatie en analyse document
Hierin komen de volgende onderwerpen aan bod:

- Aanleiding en inleiding
- Methode
- Onderzoeksresultaten
- Conclusie en vervolgstappen
- Bronverwijzingen

Architecture Description
Dit document moet inzicht geven in de beoogde architectuur voor de uitwisseling van informatie vanuit Pluriform. Er wordt nagedacht over de technische mogelijkheden voor de uitwisseling met de klant website enerzijds en een ander Pluriform systeem anderzijds. Het uitgangspunt voor de architectuur zijn de eisen en wensen van de opdrachtgever en klanten. Dit alles wordt benaderd vanuit verschillende views.

Functioneel ontwerp
Dit document geeft een conceptuele versie voor de functionele beschrijving van het te ontwikkelen systeem voor het uitwisselen van gegevens tussen Pluriform Goede Doelen en een website.

Technisch ontwerp
Dit document geeft een conceptuele versie voor de technische beschrijving van het te ontwikkelen systeem voor het uitwisselen van gegevens tussen Pluriform Goede Doelen en een website.

Set van XML berichten, een web- en XML service en een wijzigingsinterface
Naar aanleiding van het onderzoek en het doel van de opdracht zal er een set van generieke XML berichten worden ontwikkelt die aan de klant kunnen worden aangeboden. Om gebruik te maken van de berichten wordt een XML service en web service ingericht. Hiernaast wordt er een interface ontwikkeld die zorg draagt voor het afhandelen van wijzigingsverzoeken.

Beschrijvend document over de inhoud van de XML berichten
Beschrijvend document over de inhoud van de generieke XML berichten en de interface voor het afhandelen van wijzigingsverzoeken.

Handleiding voor het interpreteren en inpassen van de XML berichten en de interface

Document waarin uitleg wordt gegeven hoe de XML berichten onderhouden moeten worden

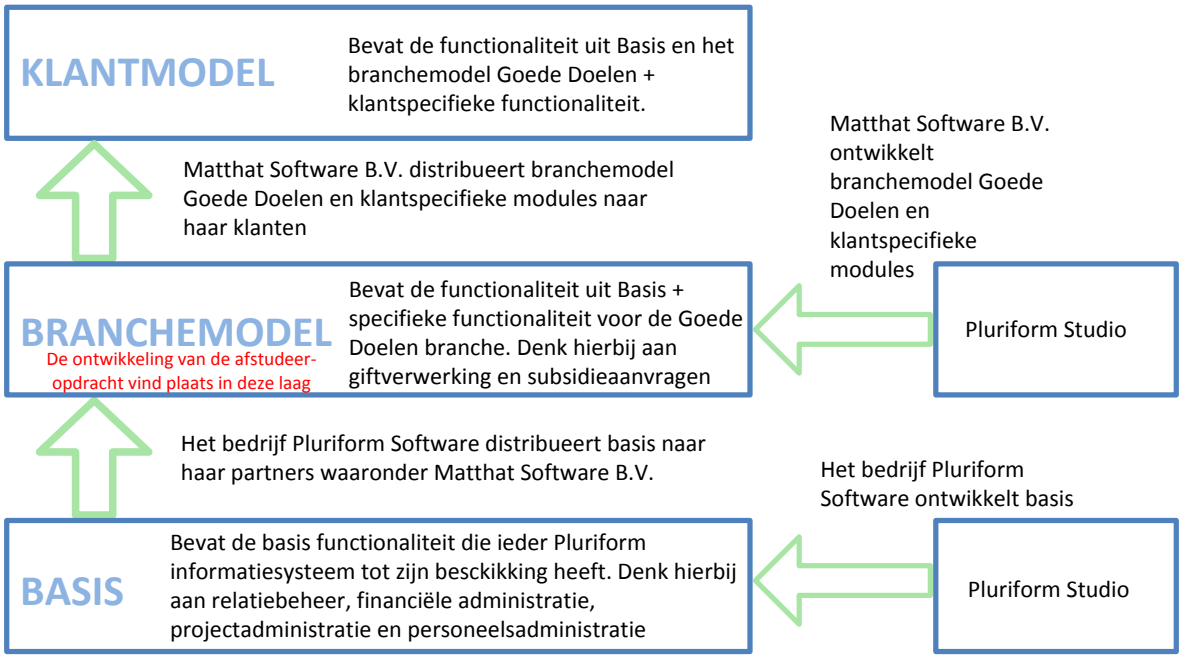
Testplan
Dit document beschrijft een plan om de berichten te kunnen testen, afhankelijk van de gebruikelijke methode binnen Matthat zal er ook een testscript opgesteld zijn.

2.6 MIJN ROL EN POSITIE TIJDENS DE AFSTUDEEROPDRACHT

Het ontwikkelen van generieke XML berichten die uitwisseling mogelijk maken tussen systeem en website, zoals beschreven wordt in paragraaf 9.2, heb ik ontwikkeld binnen de laag branchemodel (zie Figuur 4). Dit heeft ermee te maken dat de functionaliteit door mij gestandaardiseerd ontwikkeld werd, wat betekent dat het beschikbaar en bruikbaar moet zijn voor alle klanten van Matthat Software B.V. mits de klant het wil afnemen. Als klanten specifieke wensen hebben die niet in het branchemodel terecht mogen komen, dus niet beschikbaar mogen zijn voor andere klanten van Matthat Software B.V., dan zal de functionaliteit ontwikkeld worden in het klantmodel. Tijdens het ontwikkelen heb ik onder andere gebruik gemaakt van functionaliteit die aangeboden wordt uit de laag Basis, bijvoorbeeld de module XML Framework die functionaliteit aanbiedt om

XML frameworks te ontwikkelen en in te richten. Daarnaast heb ik in de laag branchemodel nieuwe functionaliteit ontwikkeld die betrekking heeft op de doelstelling zoals beschreven in paragraaf 2.4.

Mijn rol tijdens het afstudeerproject omvat alle aspecten die te maken hebben met een software ontwikkeltraject. Tijdens het inwerktraject was dit vooral een lerende en onderzoekende rol. In de inception fase ben ik door middel van een klantonderzoek dieper ingegaan op de opdracht en het doel van de opdracht. Hierbij probeerde ik het probleem en de wensen te inventariseren. In de elaboration fase werd het belangrijk om keuzes te maken uit de eisen die naar voren kwamen, gesprekken te voeren met verschillende stakeholders, een architectuur te ontwerpen en een functioneel en technisch ontwerp te maken. In de construction fase voerde ik, als ontwikkelaar, het ontwerp door naar een werkende functionaliteit in Pluriform. In de laatste fase, de transition fase, nam ik de rol aan van tester. Deze fase liep parallel aan de construction fase.



Figuur 4 De drie lagen

Knelpunten en leermomenten tijdens het inwerktraject
Tijdens het inwerktraject dat ik heb doorlopen met het afstuderen heb ik veel geleerd over vooral technische aspecten van Pluriform en de goede doelen branche. Reden hiertoe was een onbekend systeem en een nieuwe branche. Ik heb geleerd goed te luisteren naar een opdrachtgever voordat je je beeld over de opdracht al vormt. Ook heb ik geleerd om geduld te hebben met klanten waar je vragen aan hebt of feedback van verwacht. Ik heb leren werken met voor mijn beleving een nieuwe generatie van ontwikkelen. Het leertraject wat daaraan vooraf ging kostte mij erg veel moeite omdat het nodig was de filosofie achter Pluriform te begrijpen en het systeem in zijn functionaliteit te leren kennen. Tijdens het inwerktraject liep ik tegen een aantal knelpunten aan:

De scripttaal. Ik ben vanuit school gewend om mijn eigen syntax te kunnen bepalen. Omdat de syntax binnen Pluriform niet te vergelijken is met Java of C#, kostte het mij moeite om een stuk script te begrijpen. Het was dan ook een grote worsteling om te werken met het script. Het leren gebruiken van het script heeft mij ruim 2 weken bezig gehouden, en op het moment van schrijven leer ik nog steeds nieuwe onderdelen van het script. Het heeft mij geleerd om het Pluriform model te begrijpen en dat model ook te kunnen vertalen en toepassen naar de juiste context binnen het script. Ik kon de kennis die ik vanuit school had opgedaan over onder andere object georiënteerd denken en ontwikkelen hier goed toepassen.

Hergebruik van objecten. Uiteraard probeerde ik tijdens het afstuderen zo veel als mogelijk gebruik te maken van bestaande methoden en generics. Vanuit school had ik wel geleerd om dit te doen, maar dan waren er niet veel objecten en had je ze zelf ontwikkeld. Het kostte mij tijdens het afstuderen een hoop moeite en tijd vanwege de hoeveelheid aan standaard objecten en het uitzoeken wat elk object voor functie heeft. Ook al heeft Pluriform een hulp-functie voor elk object, er zijn bijna geen objecten waar de modelleur hulptekst bij heeft geplaatst. Het is voorgekomen dat ik na enig zoeken toch een eigen methode of generic schreef, maar achteraf bleek dat de functionaliteit al geïmplementeerd was in een andere methode of generic. Ik denk dat dit kwam omdat ik vooral in het begin probeerde om veel zelf te doen, in plaats van het te vragen aan een collega.

Verdiepen in de terminologie. Omdat ik bij aanvang van het afstuderen geen kennis had van XML, SOAP en XML-RPC heb ik mij hier in moeten verdiepen. Vervolgens kom ik erachter dat deze technieken een grote functionaliteit aanbieden, waardoor ik eigenlijk ‘verzonk’ in de hoeveelheid informatie en voorbeelden van implementatie mogelijkheden. Op dat moment heb ik geleerd dat je niet alles kan leren door het te lezen maar dat je het soms moet toepassen. Mede door dit knelpunt heb ik de keuze gemaakt om een afzonderlijke iteratie te doorlopen om de verschillende technieken en termen te leren.

Gebruik maken van het XML framework. De module XML framework uit de basis laag, waarin de SOAP technologie ondersteund wordt, werd bij aanvang van het afstuderen aangeboden in het Pluriform branchemodel van Matthat Software B.V. en verdween aan het begin van de tweede iteratie plotseling hieruit (zie paragraaf 7.4) Aangezien niemand bij Matthat Software B.V. kennis had van deze module, was het aan mij om daar in te verdiepen, te gebruiken en te testen. Ook dit kostte moeite vanwege mijn gebrek aan kennis met het Pluriform systeem. Ik heb hierdoor geleerd om mij aan de planning te houden, omdat ik hierdoor veel tijd zou verliezen. Daarnaast had ik ervoor gekozen om regelmatig sessies te plannen met collega’s om onderdelen uit Pluriform te bespreken.

3 AANPAK VAN DE OPDRACHT

Hoe de opdracht wordt aangepakt wordt beschreven in dit hoofdstuk. Denk hierbij aan de projectmethodiek (3.1) en de te gebruiken technieken (3.2, 3.3).

3.1 SOFTWARE ONTWIKKELMETHODE

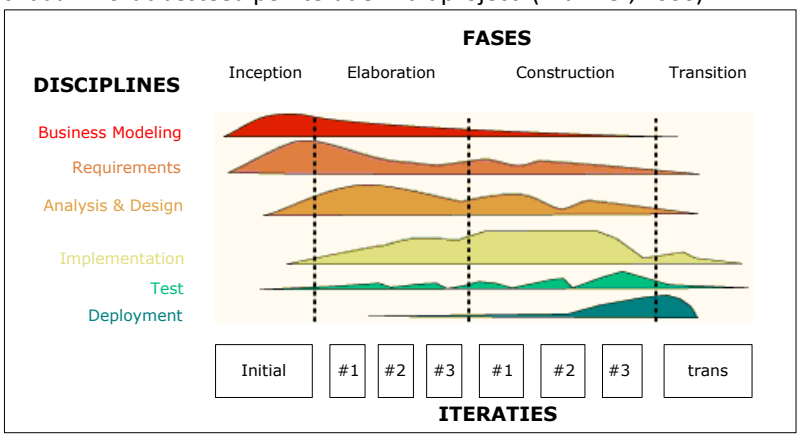
Voor het opzetten en uitvoeren van het project wordt er gewerkt volgens een ontwikkelmethode. Deze methode wordt vooraf gekozen en gebruikt om het softwareontwikkelproces (het geheel van activiteiten gericht op de ontwikkeling van software) in te richten en gestructureerd uit te voeren. Er zijn verschillende methoden die gebruikt kunnen worden in het project en zijn onder te verdelen in twee categorieën, namelijk watervalmethode (lineair) en iteratieve methoden (gefaseerd).
Matthat Software B.V. gebruikt de iteratieve methode 'Softwareontwikkeling volgens de Timebox techniek'. Deze methode toont veel overeenkomsten met de ontwikkelmethode Unified Process (hierna genoemd UP) (Kruchten, 2004):

- Iteratief ontwikkelen
- De scope (omvat de functionele- en niet functionele eisen van het project) is aanpasbaar tijdens het project
- Component gebaseerde architectuur, de software kan worden opgedeeld in stukjes.
- De kwaliteit van de software is te testen
- Het bijhouden van wijzigingen van de software tijdens het ontwikkelen

Wat in de Timebox methode niet zit maar wel in UP is het visueel modelleren met gebruik van de Unified Modeling Language (hierna genoemd UML). De watervalmethode is in vergelijking met UP ook niet goed voor dit project omdat dan veranderingen in de scope niet zijn toegestaan en fases niet naast elkaar mogen worden uitgevoerd. Een fase in de watervalmethode wordt eerst afgesloten voordat een andere fase van start gaat. Naar aanleiding van de voorgaande argumenten is er gekozen om dit project gebruik te maken van UP met UML. Pluriform biedt een aantal tools uit de ontwikkelomgeving voor het aanmaken van timeboxes. Hieraan kunnen use-cases, requirements en tests worden gehangen. Maar omdat ik modelleer in de testomgeving van Pluriform en niet in de live ontwikkelomgeving worden deze niet meegenomen bij de overgang naar productie. Ook schieten deze tools tekort als het gaat om gedetailleerd ontwerpen zoals dit kan met UML en kunnen timeboxes niet buiten het systeem worden gepubliceerd.

3.2 UNIFIED PROCESS

Tijdens het project wordt gebruik gemaakt van UP. Het verloop van het project is onderverdeeld in de vier fases van UP. Figuur 5 toont een afbeelding van deze fases en de verschillende disciplines waar per fase een hoeveelheid aandacht aan wordt besteed per iteratie in dit project. (Warmer, 2006)



Figuur 5 Unified Process met iteraties

Business modeling De discipline die vooral gebruikt wordt aan het begin van het project (inception fase). In de uitvoering van de business modeling discipline wordt er georiënteerd op het probleem van de opdracht om een visie te ontwikkelen en er wordt een plan voor aanpak gemaakt. Daarnaast wordt er onderzoek uitgevoerd naar de toe te passen technieken voor het ontwikkelen van het systeem.

Requirements Bij het uitvoeren van de requirements discipline is het doel het achterhalen en documenteren van de requirements, oftewel te bepalen wat het systeem zal moeten gaan doen. Om de requirements te achterhalen wordt er onder andere een klantonderzoek verricht.

Analysis & Design Vooral in de elaboration fase wordt de discipline analysis & design uitgevoerd. Hierbij wordt bepaald hoe de requirements worden gerealiseerd. Vanwege onervarenheid met Pluriform Goede Doelen en het moeten maken van een keuze tussen twee communicatieprotocollen is er voor gekozen om de elaboration fase op te delen in drie iteraties:

1. Het ontwerpen van een Proof Of Concept (hierna genoemd POC) en deze vervolgens te ontwikkelen. Dit heeft als doel een keuze te maken tussen communicatieprotocollen en kennis op te doen van het systeem Pluriform Goede Doelen.
2. Het ontwerpen van een wijzigingsinterface
3. Het ontwerpen van gestandaardiseerde XML berichten

Tijdens de iteraties wordt er een architectuur document, een functioneel ontwerp en een technisch ontwerp opgebouwd.

Implementation De implementation discipline wordt uitgevoerd in de construction fase. In de uitvoering wordt het ontworpen systeem gerealiseerd en getest (unittesten). De construction fase is opgedeeld in drie iteraties:

1. Het ontwikkelen van het POC.
2. Het ontwikkelen van een wijzigingsinterface
3. Het ontwikkelen van gestandaardiseerde XML berichten

Door de eerste iteratie te doorlopen wordt het mogelijk om een ontwerp te maken voor de tweede en derde iteratie. Het verslag zal de XML berichten, de XML service en de interface als ‘het systeem’ beschrijven.

Test De discipline test kan in UP in verschillende fases worden uitgevoerd. Hierdoor is het mogelijk om fouten in een vroeg stadium van het project op te sporen. Daarom wordt de uitvoering vooral in de construction- en transition fase gedaan. Tijdens de uitvoering wordt gekeken of de requirements juist zijn geïmplementeerd en de interactie tussen de verschillende objecten en de integratie van verschillende componenten gecontroleerd.

Tijdens het project wordt hiervoor een testplan en testscript opgebouwd.

Deployment De deployment discipline wordt toegepast in de laatste fase van UP namelijk de transition fase. Tijdens de uitvoering van de discipline worden de ontwikkelde software componenten, gestandaardiseerde XML berichten en een interface, geïnstalleerd en getest bij de klant. Eventueel wordt er hulp en assistentie verleent en worden fouten na oplevering opgelost.

3.3 UNIFIED MODELING LANGUAGE

De visuele modelleertaal UML (Warmer, 2006) wordt tijdens het project gebruikt voor een aantal onderdelen in de elaboration fase. Het wordt toegepast om de specificaties van de te ontwikkelen XML berichten en interface te visualiseren. Aan de hand van use-case diagrammen wordt er per bericht en voor de interface een domeinmodel opgesteld om de relaties en attributen van de benodigde klassen te visualiseren. Hiernaast wordt er een use-case diagram gemaakt met de benodigde use-case beschrijvingen en zal ik sequence diagrammen maken voor het technische ontwerp.

4 UITVOEREN INCEPTION FASE: BUSINESS MODELING EN REQUIREMENTS

Dit hoofdstuk beschrijft de eerste fase van UP zoals deze is uitgevoerd tijdens het project. In deze fase wordt een Plan van Aanpak opgesteld met planning (4.1) en een klantonderzoek opgezet en uitgevoerd (4.2) voor het achterhalen van de requirements. Daarna beschrijf ik de gesprekken met klanten (4.3) en het opstellen van een begroting voor deze klanten (4.4).

4.1 OPSTELLEN PLAN VAN AANPAK

De allereerste activiteit zoals deze is opgenomen in het afstudeerplan is het opstellen van een Plan van Aanpak. Dit document wordt in overleg met de opdrachtgever van Matthat Software B.V. opgesteld en afgesproken. In het document is een afbakening, globale planning en afspraken en randvoorwaarden opgenomen.

Een aantal punten uit de afbakening en de randvoorwaarden:

- Het onderzoek richt zich op organisaties die doen aan fondsenwerving of een leden- en/of activiteitenadministratie bijhouden.
- Uitwisseling verloopt volgens het XML-RPC protocol, tenzij na het SOAP onderzoek blijkt dat deze techniek efficiënt genoeg is om in te zetten.
- De opdracht richt zich op uitwisseling tussen een Pluriform systeem en een website en eventueel met andere Pluriform systemen.

- De opdrachtgever en collega's binnen Matthat Software B.V. zijn in staat om vragen te beantwoorden over de Goede Doelen sector en Pluriform systemen.
- De opdrachtgever is in staat om voortgangsmomenten in te plannen.

De totstandkoming van de planning (zie Tabel 1) is in samenwerking met de opdrachtgever die de op te leveren producten als eis heeft ingediend. Aan de hand van deze producten en in combinatie met de projectmethodiek UP wordt er een gefaseerde planning gemaakt met Microsoft Project.

Voor het volledige Plan van Aanpak wordt doorverwezen naar bijlage B.

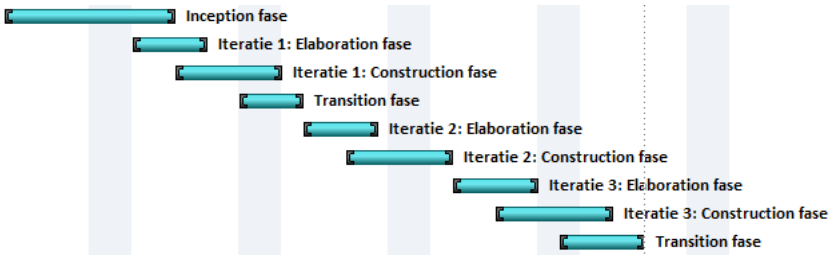
Activiteit	Duur	Begindatum	Einddatum
Inception fase	**28 dagen**	**maa 7-2-11**	**woe 16-3-11**
Plan van aanpak maken	5 dagen	maa 7-2-11	vri 11-2-11
Oriëntatie op probleem	3 dagen	maa 14-2-11	woe 16-2-11
Verdieping in Pluriform software	5 dagen	don 17-2-11	woe 23-2-11
Klantonderzoek	10 dagen	don 24-2-11	woe 9-3-11
SOAP onderzoek	5 dagen	don 10-3-11	woe 16-3-11
Elaboration fase	**12 dagen**	**don 17-3-11**	**vri 1-4-11**
Analyseren domeinmodel	3 dagen	don 17-3-11	maa 21-3-11
Architectuur beschrijven	3 dagen	din 22-3-11	don 24-3-11
Ontwerpen XML berichten, interface	6 dagen	vri 25-3-11	vri 1-4-11
Construction fase	**20 dagen**	**maa 4-4-11**	**vri 29-4-11**
Modelleren XML berichten, interface	5 dagen	maa 4-4-11	vri 8-4-11
Documenteren van de inhoud van de XML berichten	3 dagen	maa 11-4-11	woe 13-4-11
Documenteren van de manier waarop de XML berichten geïnterpreteerd en ingepast moeten worden	2 dagen	don 14-4-11	vri 15-4-11
In kaart brengen van het proces voor onderhoud voor deze XML berichten	10 dagen	maa 18-4-11	vri 29-4-11
Transition fase	**5 dagen**	**maa 2-5-11**	**vri 6-5-11**
Testen en implementeren van de XML berichten en evalueren of er sprake is van standaardisatie	5 dagen	maa 2-5-11	vri 6-5-11
Opstellen afstudeerdossier	15 dagen	maa 9-5-11	vri 27-5-11

Tabel 1 Planning

4.1.1 DE UITWERKING VAN DE PLANNING

Ik heb er voor gekozen om de elaboration en construction fase op te delen in drie iteraties per fase (zie ook Figuur 5). De eerste iteratie heeft als doel om inzicht te verkrijgen en kennis op te doen van het systeem Pluriform Goede Doelen en een keuze te maken tussen twee communicatieprotocollen door het opzetten van een POC. Vervolgens kan met de opgedane kennis en ervaring gewerkt worden aan het ontwikkelen van een wijzigingsinterface in iteratie 2 en de XML berichten in iteratie 3.

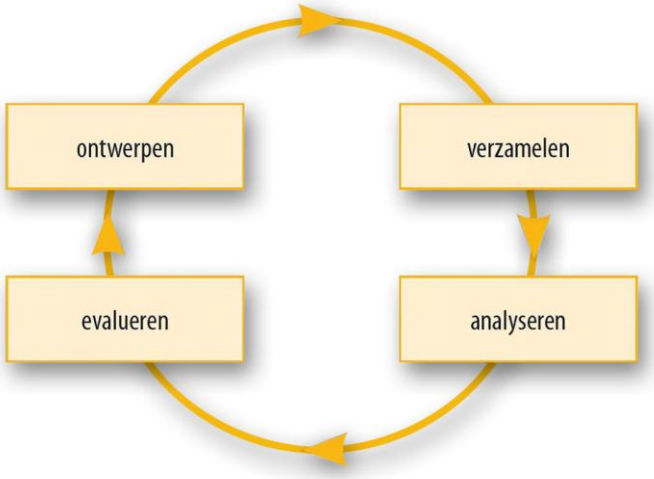
Ook al suggereert de planning zoals deze hierboven is opgenomen dat er gewerkt is volgens de watervalmethode (alle fases na elkaar), de uitwerking is exact volgens de UP methode waarin fases naast elkaar uitgevoerd worden (zie ook Figuur 6).



Figuur 6 Uitwerking UP fases

4.2 UITVOEREN VAN EEN KLANTONDERZOEK

Het onderzoeksplan beschrijft de werkwijze van het klantonderzoek binnen de afstudeeropdracht. Het geeft inzicht in de ontwerpkeuzes, strategie en inrichting van het onderzoek. Om het onderzoek gefaseerd te laten verlopen is er voor de kringloop van onderzoek (zie Figuur 7) gekozen zoals die is beschreven in het boek van Nel Verhoeven (Verhoeven, 2010).



Figuur 7 Onderzoeksfasen beschreven door Nel Verhoeven

In de fase ontwerpen wordt een onderzoeksplan en methode opgesteld. De fase verzamelen wordt in het klantonderzoek toegepast door middel van een survey. Hierna worden de verzamelde antwoorden gestructureerd en geanalyseerd. In de laatste fase worden er conclusies getrokken die moeten beantwoorden aan de probleemstelling uit het onderzoeksplan.

De aanleiding tot een klantonderzoek leidt vanuit de vraag van Matthat Software B.V. naar de mogelijkheden van standaardisatie voor uitwisseling tussen Pluriform Goede Doelen en de website van een klant. Het resultaat van het klantonderzoek heeft als doel leidend te zijn voor een inventarisatie van de wensen van de klanten met betrekking tot interactie met hun doelgroep(en). Uit de inventarisatie kunnen keuzes gemaakt worden over welke functionaliteit gestandaardiseerd kan worden. Voor het volledige onderzoeksplan wordt doorverwezen naar bijlage G.

4.2.1 ONTWERPEN

De probleemomschrijving
De centrale vraag die in het klantonderzoek beantwoord moet worden is: '*Welke interactie wensen klanten van Matthat Software B.V. met hun doelgroep(en) via hun website?*'. Om deze vraag te kunnen beantwoorden wordt deze opgedeeld in vijf deelvragen:

1. *Welke doelgroep(en) hebben de klanten?*
   Het antwoord op deze vraag schetst een beeld van de doelgroep(en) van een klant.
2. *Wat is de visie van klanten op het gebruik van internet om te communiceren met hun doelgroep(en)?*
   Het antwoord hierop geeft aan wat de beweegredenen zijn om een bepaalde boodschap wel of niet over te brengen via internet.
3. *Welke interactie bieden klanten via hun website aan hun doelgroep(en) op dit moment?*
   Het antwoord op deze vraag schetst de huidige situatie van de interactie tussen klant en hun doelgroep(en).
4. *Hoe en welk maatwerk, in relatie tot Pluriform Goede Doelen, is nu geïmplementeerd bij klanten?*
   Het antwoord op deze vraag brengt de huidige situatie in kaart hoe en welk maatwerk binnen Pluriform Goede Doelen door Matthat Software B.V. geïmplementeerd is bij de klant.

5. *Welke wensen van klanten zijn er om via hun website de interactie met hun doelgroep(en) te verhogen?* Het antwoord hierop kan gebruikt worden voor het maken van keuzes voor welke functionaliteit gestandaardiseerd moet worden.

De onderzoeksgroep
De eis van de opdrachtgever is om alle klanten mee te nemen in het onderzoek die een fondswervende organisatie zijn of een leden- en/of activiteitenadministratie hebben. Dit is een logische keuze aangezien dit de klanten zijn die een website hebben waar veel interactie is met de doelgroep van de organisatie. Er is gekozen om de contactpersonen van Matthat Software B.V. in de onderzoeksgroep op te nemen, dit zijn degenen die nauw contact hebben met Matthat Software B.V. en in het implementatieproces van Pluriform Goede Doelen een grote rol spelen. Hiernaast zijn er per klant vaak meerdere personen opgenomen in de onderzoeksgroep vanwege de verschillende functies die ze vervullen binnen de organisatie en dus allemaal vanuit een ander oogpunt de vragenlijst bekijken en beantwoorden. Naast deze onderzoeksgroep is er voor gekozen om een drietal niet-fondswervende organisaties mee te nemen in het onderzoek als controlegroep. Het uitgangspunt hierbij is dat de antwoorden van de controlegroep niet overeen mogen komen met de antwoorden van de onderzoeksgroep. Door deze twee met elkaar te vergelijken kan worden vastgesteld of de vragen thuishoren bij de onderzoeksgroep. Als blijkt dat de antwoorden overeenkomen moet het onderzoek opnieuw worden ingericht.

16 organisaties
29 personen

De vragenlijst
Omdat het klantonderzoek geen cijfermatige informatie verwacht en er geen statistische technieken gebruikt worden om beschrijvingen van de resultaten te geven is dit een kwalitatief onderzoek. Kwalitatief onderzoek kenmerkt zich door ‘onderzoekseenheden’ die als geheel worden onderzocht, men gaat op zoek naar betekenissen van situaties en verhalen van mensen. Er wordt in dit onderzoek geen vertaalslag gemaakt naar meetbare gegevens, hoewel resultaten vaak in procenten worden gepresenteerd (Verhoeven, 2010).

Het onderzoek is ingericht aan de hand van een surveyonderzoek (hierna vragenlijst genoemd). Deze methode is erg handig als meningen, opinies, houdingen en kennis bij groepen gevraagd worden en bevat veelal beschrijvings- en/of verklaringsvragen. De gesloten vragen worden gebruikt om de zogeheten respondent in te leiden op het onderwerp en doel van het onderzoek en een beeld te verkrijgen van zijn achtergrond. De open vragen dienen om antwoord te verkrijgen op de probleemstelling.

De inhoud van de vragenlijst is als volgt:
- Omschrijving van de opdrachtgever voor het aantonen van objectiviteit.
- Een onderzoek-verantwoording waarin staat aangegeven hoe het onderzoek is opgezet en uitgevoerd wordt om toevallige fouten te voorkomen en de betrouwbaarheid te waarborgen.
- De doelpopulatie.
- Het aantal respondenten om de omvang van het onderzoek aan te tonen.
- Open- en gesloten vragen.

Er zijn ook een aantal voorwaarden voor het onderzoek opgenomen in het onderzoeksplan:
- De vragenlijst is voor de start van het onderzoek getest om de betrouwbaarheid te verhogen.
- De volledige vragenlijst is opgenomen in de onderzoek-verantwoording om de betrouwbaarheid te verhogen.
- Het percentage respons is minimaal 50%, aangezien een lagere respons leidt tot selectiviteit.
- De resultaten van de onderzoeksgroep zijn niet overeenkomend met de resultaten van de controlegroep.

4.2.2 VERZAMELEN
De vragenlijst moet via e-mail worden verzonden naar de respondenten. Het nadeel hiervan is dat de respondent zijn antwoorden moeilijk kan invullen. Dus in plaats van een document als bijlage in een e-mail is er gekozen om gebruik te maken van een online enquête tool (NetQ, 2011). Het principe is als volgt: De vragenlijst wordt opgesteld in een document. Vervolgens worden alle vragen met antwoordmogelijkheden verwerkt in de tool. De tool geeft de mogelijkheid om de respondenten een uitnodiging via e-mail te versturen met een unieke

link naar de vragenlijst. Het voordeel van een online tool is dat alle antwoorden gemakkelijk te analyseren en rapporteren zijn. Voor een volledig overzicht van de vragen uit de vragenlijst wordt verwezen naar bijlage H.

4.2.3 ANALYSEREN

De resultaten van het klantonderzoek moeten worden geanalyseerd en gestructureerd om zo antwoord te kunnen geven op de deelvragen en de hoofdvraag uit de probleemstelling. Dit wordt gerealiseerd met de gebruikte tool (NetQ) bij het verzamelen van de antwoorden. Hieronder een overzicht van de kernpunten en resultaten per deelvraag:

11 organisaties
18 respondenten
62 % respons

1. *Welke doelgroep(en) hebben de klanten?*
   Leeftijd: tussen 19 en 65 jaar
   Vallen onder: donateurs, leden, vrijwilligers en deelnemers

2. *Wat is de visie van klanten op het gebruik van internet om te communiceren met hun doelgroep(en)?*
   De respondenten die reageerden geven aan dat internet ingezet wordt voor relatiebeheer, communicatie, informatieverstrekking en werving.

3. *Welke interactie bieden klanten via hun website aan hun doelgroep(en) op dit moment?*
   Antwoorden lopen hier erg uiteen. De respondenten die reageerden geven aan niet veel interactie te hebben met de doelgroep via de website. Als er interactie is gaat het vooral om standaard formulieren om wijzigingen door te geven en het aanbieden van nieuwsberichten en informatie.

4. *Hoe en welk maatwerk, in relatie tot Pluriform Goede Doelen, is nu geïmplementeerd bij klanten?*
   De helft van de respondenten die reageerden geeft aan dat er uitwisseling is tussen Pluriform Goede Doelen en de website van hun organisatie. Hier gaat het om de batch gestuurde import en RPC technieken (zie paragraaf **Fout! Verwijzingsbron niet gevonden.**).

5. *Welke wensen van klanten zijn er om via hun website de interactie met hun doelgroep(en) te verhogen?*
   13 respondenten geven aan dat er veel tot zeer veel interesse is voor een koppeling tussen Pluriform en de website van hun organisatie. Overeenkomende antwoorden voor wensen zijn uitwisseling van persoonsgegevens (NAW gegevens), giftgegevens en projectgegevens.

De analyse levert een rapportage document op waarin per onderwerp en per vraag duidelijk alle antwoorden worden getoond. Het document is vervolgens verspreid onder de medewerkers van Matthat Software B.V. en is terug te vinden in bijlage I.

4.2.4 EVALUEREN

Het onderzoek voldeed aan alle voorwaarden zoals deze opgegeven waren in het onderzoeksplan. Omdat het onderzoek een doorlooptijd had van maar twee weken bleef de respons uit. Hierdoor was het nodig om respondenten eerst een herinneringsemail te versturen en vervolgens ook te bellen en aan te manen om mee te werken aan het onderzoek.

Het eerste kritiekpunt betrouwbaarheid is met dit onderzoek moeilijk na te gaan, dit komt omdat er geen afgebakende setting is voor het onderzoek die bij herhaling dezelfde resultaten geeft. Als hetzelfde onderzoek na een aantal jaar herhaald wordt kan het zijn dat vragen niet meer relevant zijn. Toch is in het onderzoeksplan rekening gehouden met het verhogen van de betrouwbaarheid en kwaliteit door uitgebreid het probleem en de onderzoeksopzet te beargumenteren.
Het tweede kritiekpunt validiteit is bij dit kwalitatieve onderzoek geen hoofddoel. Dit komt omdat er een selecte steekproef is gedaan en de onderzoeksgroep in een bepaalde categorie, namelijk fondswervend, te onderscheiden valt. Hierdoor is de generaliseerbaarheid naar de gehele fondswervende tak van de resultaten te waarborgen.

Conclusie van het klantonderzoek
89% van de respondenten geeft aan dat er wensen zijn om de interactie met de doelgroep via de website te willen vergroten. Hierbij gaat het om nieuwe relaties vinden, meer doneermogelijkheden, mogelijkheden om wijzigingen van bijvoorbeeld NAW-gegevens door te geven en rapportages van werknemers in te dienen.

Ruim 75% van de respondenten geeft vervolgens aan dat hun website technisch beheerd en onderhouden wordt door werknemers binnen de organisatie. Zo is eenzelfde aantal het geheel eens dat deze website persoonlijke en gerichte informatie moet kunnen verstrekken en geeft 80% aan dat elke persoon een donatie moet kunnen doen via deze website.

Ruim 72% van de respondenten geeft aan veel tot zeer veel interesse te tonen in een koppeling tussen Pluriform en de website. Men geeft aan deze vorm van interactie te willen inzetten in hun eigen omgeving. Hierbij is het doel informatievoorziening, relaties onderhouden, doorgeven van wijzigingen van gegevens en vooral fondsenwerving.
Overeenkomende delen zijn het kunnen tonen en wijzigen van persoons/donateur gegevens en gift/toezeggingsgegevens. Ook al is niet in detail duidelijk om welke gegevens het gaat, kan duidelijk de scope van de te standaardiseren uitwisseling worden gesteld.

Aanbeveling naar aanleiding van het klantonderzoek
Ruim 60% van de respondenten geeft aan sociale media te willen integreren met de website. Maar men heeft geen helder beeld hoe sociale media een toegevoegde rol kunnen spelen binnen de organisatie. Daarom heb ik verschillende collega's van Matthat Software B.V. aanbevolen om over deze behoefte een visie te vormen en deze te delen met haar klanten. Naar aanleiding van mijn aanbeveling is een collega op een cursus gegaan die betrekking heeft op dit onderwerp.

4.3 GESPREKKEN MET KLANTEN NAAR AANLEIDING VAN KLANTONDERZOEK

Naar aanleiding van de resultaten van het onderzoek ben ik vervolgens met vier klanten, die veel interesse hebben, gesprekken aangegaan om te brainstormen en de wensen te bespreken. Door de respondenten van de klanten telefonisch te benaderen kan het onderzoek en de afstudeeropdracht nader worden toegelicht. Na deze gesprekken verklaarde er één klant op korte termijn, binnen de afstudeerperiode, mee te willen werken aan een implementatie van een koppeling tussen Pluriform Goede Doelen en de website. Andere klanten gaven aan uitstel te willen in verband met tijdnood, werkdruk, lopende projecten en het niet passen binnen de begroting van 2011.

De klant die mee wilde werken aan de implementatie van de koppeling in hun eigen omgeving gaf aan dat er een voorstel vanaf Matthat Software B.V. aangeleverd moest worden zodat dit intern besproken kon worden. Dit heb ik voorgelegd aan mijn begeleider en we kwamen tot de conclusie dat het voorstel opgedeeld moest worden in twee delen:

- Een architecture description. Hierin worden alle architectuur definities beschreven op een manier die begrijpelijk is voor alle stakeholders. Het adviseert en beschrijft de nieuwe architectuur voor het te ontwikkelen XML framework. Deze wordt gemaakt door de afstudeerder, die keuzes maakt, uitlegt en verantwoord. Het document moet de stakeholders tevreden stellen en laten zien dat er naar hen wordt geluisterd. Voor het volledige document wordt verwezen naar bijlage J.
- Een begrotings document. Hierin worden alle kosten voor het ontwikkelen, implementeren en de licenties, de meerwaarde voor de klant en tijdsbegrotingen opgenomen. Dit document wordt gemaakt door de afstudeerder in overleg met de manager en de begeleider. Na overleg met de manager heb ik besloten om dit document niet op te nemen vanwege gevoelige informatie. In paragraaf 4.4 wordt het proces beschreven over het opstellen van een begroting en maak in een voorbeeldberekening.

Mijn rol en doel tijdens de klantgesprekken

Na het afronden van het onderzoek heb ik besloten om gesprekken te voeren met klanten die in het onderzoek aangaven mee te willen denken, concrete wensen hadden of geïnformeerd wilden worden over de mogelijkheden die het XML framework biedt. Deze gesprekken waren voor mij essentieel voor het opstellen van de wensen en eisen. Het kan namelijk niet zo zijn dat de eisen die ik opstel gebaseerd zijn op de wensen en eisen van één klant (hierna genoemd stakeholder). Namens Matthat Software B.V. stelde ik mij voor en informeerde ik de klant over de afstudeeropdracht. Tijdens het gesprek waakte ik er steeds voor dat het doel van het gesprek werd nagestreefd, namelijk de stakeholder informeren en zijn wensen gedetailleerd achterhalen.

Knelpunten en leermomenten tijdens het onderzoek en de klantgesprekken
Het opzetten van het klantonderzoek verliep zonder knelpunten. Ik had veel kennis opgedaan in het I7 blok 'Actuele trends in ICT & Media' op de Haagse Hogeschool, en dan specifiek de lessen over onderzoek. Desondanks had ik toch erg moeite met het formuleren van de vragen, omdat ik niet bij het doel van het onderzoek bleef maar soms in de vragen mijn eigen wensen en eisen probeerde te sturen naar de klant. Daarnaast had ik moeite met vraagstelling zo te formuleren dat de klanten het begrepen. Dit heb ik opgelost door de vragenlijst in verschillende versies naar al mijn collega's te sturen, feedback te laten geven en dit vervolgens verwerkt in de vragenlijst. Ik heb geleerd om als onderzoeker mij te verplaatsen in de doelgroep en objectief een onderzoek te behandelen.

Een ander knelpunt tijdens het onderzoek was de klanten eraan herinneren om de vragenlijst in te vullen. Ik had ervoor gekozen om dit per telefoon te doen. Een gesprek verliep soms niet soepel als ik aan een klant, die het erg druk had, uit wilde leggen hoe groot het belang was voor mijn afstuderen. Ik probeerde dit dan tegenover de voordelen voor de klant te vertellen om zo uiteindelijk een respons te krijgen. Na twee herinneringen en een hoog respons percentage had ik besloten om het onderzoek af te sluiten. Het is voor mij belangrijk dat ik de klant ken en daarnaast zal ik mij zelf moeten ontwikkelen in de communicatie.

Tijdens een klantgesprek was het voor mij soms lastig om vragen te beantwoorden van een klant, vanwege de inhoudelijke kennis van Pluriform die op dat moment niet erg groot was. Maar omdat ik mij goed had ingelezen op de antwoorden van de klant in het onderzoek en bij het doel van het gesprek bleef, had elk gesprek grote waarde voor de afstudeeropdracht. Daarnaast had ik gekozen om een collega bij het gesprek te betrekken om eventueel in te springen op vragen die ik niet kon beantwoorden.

4.4 OPSTELLEN VAN EEN BEGROTING

Naar aanleiding van mijn gesprekken met verschillende klanten kreeg ik de vraag om een kostenoverzicht en het resultaat van deze kosten voor de klanten op te stellen. Helaas heb ik vanuit school weinig tot geen ervaring opgedaan met kostenberekeningen. Daarom heb ik gevraagd aan mijn manager wat er nodig is om een begroting te maken. Allereerst moest ik het aantal uren berekenen die nodig zijn om een volledige implementatie te doen voor een klant van de functionaliteit die ik ontwikkel. Ik heb hiervan een inschatting gemaakt na de eerste iteratie in het afstudeerproces, omdat ik hierna wist welke handelingen er nodig zijn voor een implementatie bij een klant. Alle handelingen die nodig zijn voor de implementatie komen ook naar voren in het AD. Denk hierbij aan het installeren van een webconnector en het updaten van het klantsysteem met de XML framework module. Ik moest navragen of Matthat Software B.V. ook de webservices aanmaakt in IIS in het klantsysteem, maar dat gebeurt altijd in overleg met de systeembeheerder van de klant. Mijn inschatting kwam hierdoor op 4 werkuren voor de implementatie. Hierbij mocht ik van mijn manager eventuele tijd voor het oplossen van problemen meerekenen.
Ik wist niet hoe ik de kosten in uren moest berekenen voor de ontwikkeling van het framework. Mijn collega raadde mij aan om het aantal uren op te delen in de tijd die de verschillende stakeholders erin stoppen. De stakeholders zijn tijdens het ontwikkelen mijn collega (40 uur begeleiding in totaal), Pluriform Software (40 uur in totaal) en ik zelf (17 werkweken maal 40 uur).
Samen met mijn manager hebben we het AD doorgenomen en konden we in de begroting laten zien wat de klant specifiek zou krijgen aan functionaliteit in zijn systeem (met als kernpunt de mogelijkheid om persoonsgegevens en donatiegegevens te ontsluiten). En wat hij nodig had om die te kunnen implementeren (bijv. een licentie voor de laatste versie van Pluriform waarin SOAP standaard ondersteund wordt). Doordat ik het aantal begrote uren had overgedragen aan mijn manager kon hij er een uurtarief aan koppelen.

Naar aanleiding van het klantonderzoek kon ik inschatten dat ongeveer 5 klanten binnen een jaar het XML framework zullen implementeren. Mijn manager heeft mij uitgelegd dat Matthat Software B.V. werkt op basis van kostendekking (kosten zoveel mogelijk drukken voor de klant: lagere kosten bieden meer financiën voor de projecten), wat betekent dat de ontwikkelkosten gedeeld worden door 5 en de klanten dus samen de ontwikkelkosten dragen. Het komt er dan op neer dat een klant 20% van de investering bijdraagt maar voor 100% profiteert van de ontwikkelingen binnen het XML framework. Hiernaast zullen specifieke klantwensen, dus die niet standaard in het door mij ontwikkelde XML framework zitten, apart worden berekend. Tabel 2 toont een voorbeeldberekening voor klant X waarin de kosten verdeeld worden onder 'ontwikkeling' en 'implementatie en gebruik', en die 2 methodes specifiek voor zijn organisatie wil.

Ontwikkeling		
Begeleiding: 40 uur X € 100	€ 4000	
Kosten aan Pluriform Software: 40 uur X € 175	€ 7000	
Kosten afstudeerder: 680 uur X € 6	€ 4000	

	€15000 / 5	
Totale kosten ontwikkeling		€ 3000
Implementatie en Gebruik		
Implementatiekosten: 4 uur X €100		€ 400
Ontwikkeling maatwerk 2 extra methodes		€ 400
Licentiekosten Pluriform Software: één module		€ 0,375
(kosten per timeslot van 10 minuten)		
		--------- +
Totale kosten		€ 3800

Tabel 2 Voorbeeldberekening kosten

5 VAN ONDERZOEK NAAR REQUIREMENTS

Dit hoofdstuk beschrijft de relatie tussen het klantonderzoek, de klantgesprekken en de requirements. In de eerste paragraaf (5.1) worden de stakeholders binnen de afstudeeropdracht bepaalt, de tweede paragraaf (5.2) beschrijft per stakeholder de requirements en de derde paragraaf (5.3) beschrijft de prioritering van deze requirements volgens de MoSCoW methode. Het doel van dit hoofdstuk is verklaren hoe tot de requirements is gekomen.

5.1 BEPALEN VAN DE STAKEHOLDERS

Binnen de afstudeeropdracht onderscheidt ik de volgende stakeholders:

- **De klanten van Matthat Software B.V.** De klant is de belangrijkste stakeholder tijdens dit project. De inbreng van klanten is dan ook belangrijk voor het kiezen van de functionaliteit die aanwezig moet zijn in het XML framework. In overleg met de klant kan de door te voeren standaardisatie dus een richting in gaan die vooraf niet bepaald was. Hiernaast is de klant de stakeholder waaraan de functionaliteit uiteindelijk verkocht zal worden.
- **De opdrachtgever Matthat Software B.V.** De opdrachtgever speelt een belangrijke rol in het opzetten van een architectuur voor dit project. Matthat Software B.V. heeft kennis en ervaring hiermee, wat zal worden meegenomen in overwegingen voor het opzetten van een architectuur. Daarnaast is de opdrachtgever de stakeholder die grote inspraak heeft over de kosten die gepaard gaan met de opdracht.
- **De afdeling Research & Development van Matthat Software B.V.** De werknemers op de ontwikkelafdeling van Matthat Software B.V. zijn degenen met technische ervaring binnen Pluriform wat kan helpen bij het maken van technische keuzes. Hierdoor zullen technische overwegingen altijd voorgelegd worden aan deze personen. Deze stakeholder is daarnaast ook belangrijk vanwege het feit dat die het te ontwikkelen systeem moeten kunnen onderhouden en aanpassen. Op de eisen en wensen van deze stakeholder moet dus extra gelet worden tijdens het ontwikkelen van het systeem

5.2 REQUIREMENTS PER STAKEHOLDER

Het is van essentieel belang voor de stakeholders dat in het te ontwikkelen systeem terug te vinden is dat er geluisterd is naar hun wensen en belangen. Daarom is het van belang om alle wensen en belangen te vertalen naar een set requirements voor het te ontwikkelen systeem. In deze fase van de afstudeeropdracht kan de set requirements nog veranderen.

5.2.1 DE KLANTEN UIT ONDERZOEK

Naar aanleiding van het klantonderzoek (zie bijlage I) en de gevoerde gesprekken met de klanten maak ik een vertaalslag van de wensen en eisen van deze klanten naar een set requirements.

Eisen uit onderzoek
Naar aanleiding van vraag 10 uit het klantonderzoek worden de volgende eisen gesteld:

- Het systeem mag alleen persoonlijke en gerichte informatie vanuit Pluriform aanbieden aan de gebruiker. Dit betekent dat iedere gebruiker alleen toegang heeft via het systeem op informatie uit Pluriform waar hij bij betrokken is.
- De gebruiker moet door het systeem eerst geauthentiseerd zijn voordat toegang tot informatie via het systeem mogelijk wordt voor de gebruiker.
- Uitwisseling van informatie tussen systeem en website verloopt via een HTTPS verbinding.
- Het systeem moet het aanbieden van informatie beperken tot de gewenste informatie van de klant. Dit betekent dat de gebruiker geen informatie krijgt van het systeem als hier niet om gevraagd wordt.

Naar aanleiding van vraag 12, 13 en 17 uit het klantonderzoek, waar 75% overeenkomende wensen had, worden de volgende eis gesteld:

- Het systeem biedt mogelijkheden voor het opvragen en wijzigen van persoons/donateurgegevens en gift/toezeggingsgegevens (Dit zijn de overeenkomende delen uit de antwoorden van de klanten, andere antwoorden worden niet meegenomen in deze set requirements anders kan niet worden voldaan aan de eis van standaardisatie).

Eisen uit gesprekken
Naar aanleiding van de gesprekken met 4 klanten die mee gewerkt hebben aan het onderzoek worden de volgende eisen gesteld:

- Het systeem heeft een beschikbaarheid van 100% met uitzondering van planned en unplanned downtime (Onder 'planned downtime' valt onder andere het installeren van updates en gepland onderhoud en onder 'unplanned downtime' vallen onder andere hardware- en software matige storingen. Afspraken over 'unplanned downtime' worden vastgelegd in het Service Level Agreement).
- Het implementeren van het systeem en het aansluiten op een website moet eenvoudig zijn. Dit wordt als volgt gerealiseerd:
    - Opleveren van het proces hoe de klant het systeem in gebruik kan nemen
    - Documenteren van de inhoud van de XML berichten zodat een website ontwikkelaar de informatie kan interpreteren en inpassen in de website
    - Een mogelijkheid aanleveren voor het testen van het systeem
- Het systeem biedt de mogelijkheid om wijzigingen van gegevens, die door de gebruiker geïnitieerd worden, eerst te plaatsen voor controle en accordatie

5.2.2 DE OPDRACHTGEVER MATTHAT SOFTWARE B.V.

De eisen van de opdrachtgever heb ik tijdens het formuleren van de afstudeeropdracht en het hij opvolgende overleg kunnen vastleggen:

- Het te ontwikkelen systeem levert een gestandaardiseerd set van XML berichten op, zodat het herbruikbaar is voor verschillende implementaties bij klanten.
- Het systeem wordt ontwikkeld als onderdeel van het Goede Doelen branchemodel.
- In Pluriform wordt voor het systeem gedocumenteerd hoe het systeem in- en toe te passen is.

5.2.3 DE AFDELING RESEARCH EN DEVELOPMENT

Na enkele brainstormsessies met twee collega's op de ontwikkelafdeling kunnen er een aantal eisen worden gesteld:

- Het systeem moet een functionaliteit aanbieden aan de gebruiker om zich te kunnen authentificeren met gebruikersnaam en wachtwoord zoals deze worden vastgelegd in de klasse Gebruikersaccount (extern).
- Het systeem biedt functionaliteit om gegevens in de volgende objecten binnen Pluriform te wijzigen, op te vragen en te plaatsen:
    - Persoon
    - Adres
    - Gift
    - Toezegging
- Het onderhouden en aanpassen van het systeem moet gedaan kunnen worden door de ontwikkelaars bij Matthat Software B.V. Dit wordt als volgt gerealiseerd:
    - Het te ontwikkelen systeem levert een gestandaardiseerd set van XML berichten op, zodat het herbruikbaar is voor verschillende implementaties bij klanten.
    - Het systeem is dusdanig gedocumenteerd in Pluriform zodat bijvoorbeeld de eerstelijns support van Matthat Software B.V. een call die betrekking heeft op het systeem kan afhandelen binnen de afgesproken reactie- en oplostijden (deze worden vastgelegd in een SLA).

Doordat ik de eisen van de klanten ook aan mijn collega's had voorgelegd kwamen we, naar aanleiding van de eis dat het systeem wijzigingen van gegevens eerst moet plaatsen voor controle en accordatie, op de volgende eisen:

- Het systeem biedt functionaliteit voor het maken van instellingen op de volgende onderdelen:

  - Van elke attribuut en elke klasse in Pluriform moet ingesteld kunnen worden of wijzigingen hierop direct gemuteerd mogen worden.
  - Het instellen welke rollen binnen Pluriform wijzigingen mogen doorvoeren en bovenstaande instellingen mogen instellen.
- Als er een wijziging wordt doorgegeven via het systeem op een attribuut die niet direct gemuteerd mag worden, dan maakt het systeem een wijzigingsvoorstel aan.
- Op een wijzigingsvoorstel kunnen de volgende acties worden ondernomen:
  - Bewerken
  - Verwijderen
  - Annuleren
  - Afwijzen
  - Doorvoeren
- Een wijzigingsvoorstel wordt gekenmerkt door een status die is vastgelegd in Pluriform en kan wijzigen naar aanleiding van bovenstaande acties.

5.3 ANALYSE EN PRIORITERING VAN DE REQUIREMENTS

Het analyseren en prioriteren van de requirements gebeurd op basis van de MoSCoW methode. De MoSCoW methode kent vier prioriteringen en zullen worden gebruikt om de requirements in te delen:

- **Must have this**. Regel: deze eis moet in het eindresultaat terugkomen, zonder deze eis is het product niet bruikbaar.
- **Should have this if at all possible**. Regel: deze eis is zeer gewenst, maar zonder is het product wel bruikbaar.
- **Could have this if it does not affect anything else**. Regel: deze eis mag alleen aan bod komen als er tijd genoeg is.
- **Won't have this but would like to have this in the future**. Regel: deze eis zal in dit project niet aan bod komen maar kan in de toekomst, bij een vervolg project, interessant zijn.

Bij de totstandkoming van de prioritering van de eisen heb ik bovenstaande regels in acht genomen. Bij de uitwerking van de eisen stel ik als regel dat eisen met prioritering Must of Should uitgewerkt moeten worden om het project als geslaagd te beschouwen.

Naast de prioritering van de requirements is het van belang om ze onder te verdelen onder twee onderwerpen. Namelijk het gedrag en de kwaliteit van het systeem, ook wel functionele en niet- functionele softwarerequirements genoemd. Hierbij hou ik rekening met de definities van deze twee termen zoals deze beschreven worden in ‘Handboek requirements’ van Nicole de Swart (de Swart). De niet- functionele softwarerequirements zeggen iets over de kwaliteit van het systeem in de volgende categorieën van ISO 9126:

- Functionaliteit
  - Juistheid
  - Koppelbaarheid
  - Beveiligbaarheid
- Betrouwbaarheid
  - Bedrijfszekerheid
  - Foutbestendigheid
- Bruikbaarheid
  - Begrijpelijkheid
  - Leerbaarheid
  - Bedienbaarheid
- Onderhoudbaarheid
  - Analyseerbaarheid
  - Wijzigbaarheid
  - stabiliteit

- Overdraagbaarheid
  - Aanpasbaarheid
  - Installeerbaarheid

De categorie Efficiëntie uit ISO 9126 komt niet aan de orde in de requirements. Geen van de stakeholders heeft op dit moment wensen of eisen die binnen deze categorie vallen. Ik kan me voorstellen dat na een implementatie van het systeem bij een klant er nieuwe eisen en wensen ontstaan die wel in deze categorie vallen en betrekking hebben op responsiesnelheid en middelenbeslag.

5.3.1 OPSTELLEN FUNCTIONELE REQUIREMENTS

Een functionele softwarerequirement is gedrag (functionaliteit) dat het systeem moet vertonen om in de behoefte te voorzien van een stakeholder.

Must have this

- Het systeem biedt mogelijkheden voor het opvragen en wijzigen van persoons/donateurgegevens en gift/toezeggingsgegevens (Dit zijn de overeenkomende delen uit de antwoorden van de klanten, andere antwoorden worden niet meegenomen in deze set requirements anders kan niet worden voldaan aan de eis van standaardisatie). *Deze eis moet doorgevoerd worden in het systeem om dat het essentieel is voor het behalen van het doel van de afstudeeropdracht*
- Het systeem biedt de mogelijkheid om wijzigingen van gegevens, die door de gebruiker geïnitieerd worden, eerst te plaatsen voor controle en accordatie. *Deze eis moet doorgevoerd worden in het systeem om de klant eist dat er een controle en accordatie over bepaalde wijzigingen heen moet*
- Het te ontwikkelen systeem levert een gestandaardiseerd set van XML berichten op, zodat het herbruikbaar is voor verschillende implementaties bij klanten. *Deze eis moet doorgevoerd worden in het systeem om dat het essentieel is voor het behalen van het doel van de afstudeeropdracht*
- Het systeem wordt ontwikkeld als onderdeel van het Goede Doelen branchemodel. *Deze eis moet doorgevoerd worden in het systeem om dat het essentieel is voor het behalen van het doel van de afstudeeropdracht en beschikbaar moet zijn voor alle klanten van Matthat Software B.V.*
- Het systeem biedt functionaliteit om gegevens in de volgende objecten binnen Pluriform te wijzigen, op te vragen en te plaatsen:
  - Persoon
  - Adres
  - Gift
  - Toezegging

  Deze eis moet worden doorgevoerd in het systeem om te voldoen aan de eerste eis
- Het systeem biedt functionaliteit voor het maken van instellingen op het volgende onderdeel:
  - Van elke attribuut en elke klasse in Pluriform moet ingesteld kunnen worden of wijzigingen hierop direct gemuteerd mogen worden.

  Deze eis moet doorgevoerd worden in het systeem om de klant eist dat er een controle en accordatie over bepaalde wijzigingen heen moet en Matthat Software B.V. eist dat de klant dit zelf kan instellen
- Als er een wijziging wordt doorgegeven via het systeem op een attribuut die niet direct gemuteerd mag worden, dan maakt het systeem een wijzigingsvoorstel aan.
  Deze eis moet worden doorgevoerd in het systeem om te voldoen aan de tweede eis
- Op een wijzigingsvoorstel kunnen de volgende acties worden ondernomen:
  - Afwijzen
  - Doorvoeren

  Deze eis moet worden doorgevoerd in het systeem om te voldoen aan de tweede eis

Should have this if at all possible

- De gebruiker moet door het systeem eerst geauthentiseerd zijn voordat toegang tot informatie via het systeem mogelijk wordt voor de gebruiker. (vanuit login oogpunt) *Uit veiligheid zou deze eis een must zijn maar het systeem is bruikbaar zonder deze eis*
- Het systeem moet het aanbieden van informatie beperken tot de gewenste informatie van de klant. Dit betekent dat de gebruiker geen informatie krijgt van het systeem als hier niet om gevraagd wordt.

In sommige gevallen kan het zijn dat er extra informatie verstuurd wordt door het systeem, maar de websiteontwikkelaar kan dit afvangen. Daarom is het should

Could have this if it does not affect anything else

- Het systeem moet een functionaliteit aanbieden aan de gebruiker om zich te kunnen authentificeren met gebruikersnaam en wachtwoord zoals deze worden vastgelegd in de klasse Gebruikersaccount (extern). *Deze eis stellen niet alle klanten, dus wordt niet hoog geprioriteerd*
- Op een wijzigingsvoorstel kunnen ook de volgende acties worden ondernomen:
    - Bewerken
    - Verwijderen
    - Annuleren

  Deze eis heeft minder prioriteit dan het doorvoeren of afwijzen, het systeem functioneert ook zonder deze eis
- Een wijzigingsvoorstel wordt gekenmerkt door een status die is vastgelegd in Pluriform en kan wijzigen naar aanleiding van het uitvoeren van bovenstaande acties.
  Zonder deze eis is het systeem bruikbaar, maar heeft lage prioriteit vanwege de functionaliteit die het biedt

Won't have this but would like to have this in the future

- Het systeem biedt functionaliteit voor het maken van instellingen op het volgende onderdeel:
    - Het instellen welke rollen binnen Pluriform wijzigingen mogen doorvoeren en bovenstaande instellingen mogen instellen.

  Zonder deze eis is het systeem bruikbaar, maar kan later worden doorgevoerd

5.3.2 OPSTELLEN NIET-FUNCTIONELE REQUIREMENTS

Een niet functionele softwarerequirement is een kwaliteitseis waaraan het systeem moet voldoen om in een behoefte te voorzien van een stakeholder. Per requirement geef ik aan in welke categorie van kwaliteitskenmerken hij valt volgens de internationale standaard over softwarekwaliteit ISO 9126.

Must have this

- Juistheid, beveiligbaarheid: Het systeem mag alleen persoonlijke en gerichte informatie vanuit Pluriform aanbieden aan de gebruiker. Dit betekent dat iedere gebruiker alleen toegang heeft via het systeem op informatie uit Pluriform waar hij bij betrokken is. *Deze eis is zeer belangrijk vanwege de gevoeligheid van gegevens uit Pluriform*
- Aanpasbaarheid: Het onderhouden en aanpassen van het systeem moet gedaan kunnen worden door de ontwikkelaars bij Matthat Software B.V. Dit wordt als volgt gerealiseerd:
    - Het te ontwikkelen systeem levert een gestandaardiseerd set van XML berichten op, zodat het herbruikbaar is voor verschillende implementaties bij klanten.

  Deze eis moet doorgevoerd worden in het systeem om dat het essentieel is voor het behalen van het doel van de afstudeeropdracht en het zijn de ontwikkelaars die het systeem moeten kunnen beheren
- Bedrijfszekerheid, foutbestendigheid, stabiliteit: Het systeem heeft een beschikbaarheid van 100% met uitzondering van planned en unplanned downtime (Onder ‘planned downtime’ valt onder andere het installeren van updates en gepland onderhoud en onder ‘unplanned downtime’ vallen onder andere hardware- en software matige storingen. Afspraken over ‘unplanned downtime’ worden vastgelegd in het Service Level Agreement). *Deze eis is essentieel voor het functioneren van het systeem en krijgt de hoogste prioriteit*

Should have this if at all possible

- Beveiligbaarheid: Uitwisseling van informatie tussen systeem en website verloopt via een HTTPS verbinding. *Zonder een beveiligde verbinding is het systeem bruikbaar, maar het heeft prioriteit vanwege de gevoeligheid van gegevens uit Pluriform*
- Analyseerbaarheid, wijzigbaarheid: Het onderhouden en aanpassen van het systeem moet gedaan kunnen worden door de ontwikkelaars bij Matthat Software B.V. Dit wordt als volgt gerealiseerd:
    - Het systeem is dusdanig gedocumenteerd in Pluriform zodat bijvoorbeeld de eerstelijns support van Matthat Software B.V. een call die betrekking heeft op het systeem kan

afhandelen binnen de afgesproken reactie- en oplostijden (deze worden vastgelegd in een SLA).

Deze eis draagt niet bij aan het functioneren van het systeem, maar heeft prioriteit vanwege de afspraken in een SLA

Could have this if it does not affect anything else

- Aanpasbaarheid, installeerbaarheid, koppelbaarheid: Het implementeren van het systeem en het aansluiten op een website moet eenvoudig zijn. Dit wordt als volgt gerealiseerd:
  - Opleveren van het proces hoe de klant het systeem in gebruik kan nemen
  - Documenteren van de inhoud van de XML berichten zodat een website ontwikkelaar de informatie kan interpreteren en inpassen in de website
  - Een mogelijkheid aanleveren voor het testen van het systeem

  Zonder deze eis is het systeem bruikbaar maar wordt het lastiger voor een websiteontwikkelaar om het te implementeren
- Begrijpelijkheid, leerbaarheid, bedienbaarheid: In Pluriform wordt voor het systeem gedocumenteerd hoe het systeem in- en toe te passen is. *Deze eis heeft betrekking op de Hulp-functie van Pluriform. Het is niet meer dan handig als dit gedocumenteerd is, maar het draagt niet bij aan de functionaliteit van het systeem.*

6 OPSTELLEN VAN EEN ARCHITECTUUR

Voor de communicatie richting klanten en intern bij Matthat Software B.V. heb ik een architectuur document opgesteld. Dit document beschrijft de architectuur van uitwisseling van gegevens tussen Pluriform Goede Doelen en een website van een klant vanuit verschillende oogpunten, ook wel viewpoints genoemd (Rozanski, 2005). Deze viewpoints worden ieder vertegenwoordigd door 1 of meerdere stakeholders. Per viewpoint wordt een stuk van de architectuur beschreven die voor het betreffende viewpoint van belang is. Het document is dus bedoeld om duidelijkheid te scheppen over de architectuur van het systeem, wat is gebeurd op basis van de functionele eisen en de verschillende kwaliteitsattributen (zie hoofdstuk 5). Voor het architectuur document wordt verwezen naar bijlage J.

6.1 SYSTEM SCENARIOS

Vanuit de requirements in hoofdstuk 5 worden use-cases geformuleerd (zie Figuur 8).Omdat de architectuur van het systeem begrijpelijk moet zijn voor alle stakeholders, heb ik vanuit de bezoeker van een website en een gebruiker van Pluriform gedacht om tot de use-cases te komen. Vanuit de eis om persoons/donateurgegevens en gift/toezeggingsgegevens te kunnen ophalen en wijzigen door een bezoeker van een website kom ik tot de volgende use-cases:

1. Geef overzicht van mijn persoonsgegevens
2. Wijzig mijn persoonsgegevens
3. Geef overzicht van mijn adres(sen)
4. Voeg een nieuw adres toe
5. Wijzig mijn adresgegevens
6. Geef overzicht van mijn donaties
7. Geef overzicht van mijn toezeggingen

Omdat er verschillende eisen waren met betrekking tot authenticatie door het systeem en de mogelijkheid om wijzigingen eerst klaar te zetten voor controle en accordatie kom ik vervolgens tot de volgende use-cases:

8. Log mij in bij Pluriform
9. Maak een wijzigingsvoorstel aan

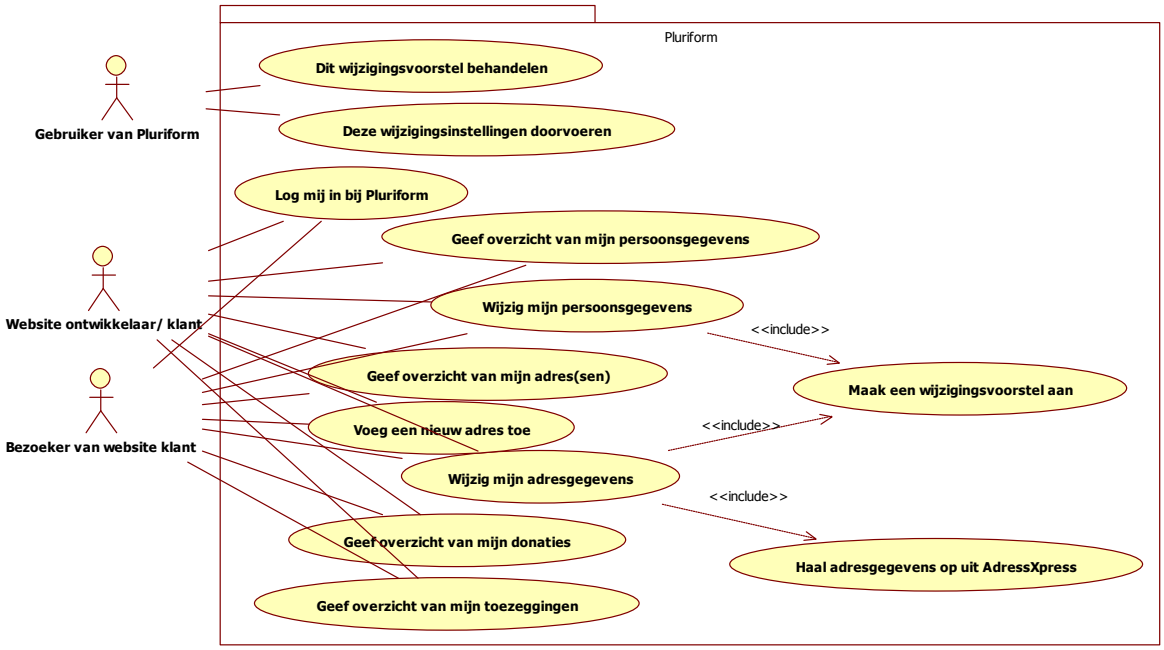
Omdat ik er voor kies om het mogelijk te maken om adresgegevens toe te voegen en te wijzigen via het systeem, moet ik het voorkomen dat er verkeerde adressen in Pluriform komen. Hierdoor kom ik tot de volgende use-case:

10. Haal adresgegevens op uit AdressXpress

Omdat er vanuit de afdeling Research en Development eisen kwamen om een wijzigingsvoorstel te kunnen behandelen en instellingen te maken in het systeem wanneer er wel en wanneer er geen wijzigingsvoorstel gemaakt wordt, kom ik tot de volgende use-cases door een gebruiker van Pluriform:

11. Dit wijzigingsvoorstel behandelen
12. Deze wijzigingsinstellingen doorvoeren

Figuur 8 toont een het use-case diagram.



Figuur 8 Use-cases

Uit bovenstaand figuur heb ik vervolgens scenario's afgeleid. Deze scenario's heb ik beschreven om alle stakeholders te kunnen verzekeren dat de opgezette architectuur de functionaliteit en het gedrag vertoont die er vereist is. Hiernaast zijn ze zeer bruikbaar op het moment dat het systeem zal worden getest:

Scenario 1#
Overview: Hoe het systeem omgaat met het ophalen van gegevens via de website uit Pluriform door een bezoeker van de website
System state: De op te halen gegevens zijn aanwezig in Pluriform en het systeem is beschikbaar om binnenkomende XML berichten af te handelen
System environment: De web connector is operationeel, de web service is te benaderen vanaf de website en het systeem is operationeel
External stimulus: Een XML bericht komt binnen via de web service en wordt doorgezonden naar Pluriform
Required system response: Het systeem stuurt na het ophalen van de gegevens een XML bericht terug naar de website met daarin de gegevens die werden opgevraagd door de website

Scenario 2#
Overview: Hoe het systeem omgaat met het wijzigen van gegevens via de website in Pluriform door een bezoeker van de website
System state: De te wijzigen gegevens zijn aanwezig in Pluriform, de te wijzigen gegevens mogen direct gewijzigd worden en het systeem is beschikbaar om binnenkomende XML berichten af te handelen
System environment: De web connector is operationeel, de web service is te benaderen vanaf de website en het systeem is operationeel
External stimulus: Een XML bericht komt binnen via de web service en wordt doorgezonden naar Pluriform
Required system response: Het systeem stuurt na het wijzigen van de gegevens een XML bericht terug naar de website met daarin een bevestiging van het doorvoeren van de wijziging

Scenario 3#
Overview: Hoe het systeem omgaat met het verwerken van een wijzigingsvoorstel in het Pluriform systeem door een gebruiker van het Pluriform systeem

System state: Het wijzigingsvoorstel is aanwezig in Pluriform en het systeem biedt de mogelijkheid om een wijzigingsvoorstel te verwerken
System environment: Het systeem is operationeel en de gebruiker heeft de juiste rol om wijzigingsvoorstellen af te handelen
External stimulus: De gebruiker selecteert een wijzigingsvoorstel en drukt op bijvoorbeeld 'Doorvoeren'
Required system response: Het systeem voert het wijzigingsvoorstel door en laat door bepaalde interactie met de gebruiker zien dat dit is gelukt

Het systeem zal in elk scenario anders functioneren en ander gedrag vertonen. Deze scenario's van het systeem heb ik op architectuur niveau beschreven en zal ik later bij de system qualities uitbreiden. De aanleiding tot het maken van scenario's waren verschillende klanten die met vragen en generieke problemen naar mij toe kwamen waarvoor ik een oplossing moest bedenken.

6.2 VOORGESTELDE ARCHITECTUUR

Bij het opstellen van de voorgestelde architectuur heb ik zoveel mogelijk rekening gehouden met de eisen en wensen van de betrokken stakeholders. Dit leidde tot een aantal verschillende toevoegingen in de infrastructuur en een aantal afspraken over de te ontwikkelen functionaliteiten en kwaliteiten waaraan het systeem dient te voldoen.

6.2.1 FUNCTIONAL VIEW

In de functional view kijk ik naar de functionaliteiten die het systeem moet bieden en welke componenten hierbij betrokken zijn.

Funtional capabilities
Er wordt een webserver ingericht waar IIS op zal worden geïnstalleerd. Binnen IIS wordt een web service aangemaakt die binnenkomende en uitgaande XML berichten afhandelt. Deze web service zal een service contract bieden die leidend zal zijn bij implementatie van het systeem.

- *Dit heb ik als eis opgenomen in voorstellen richting klanten. Het geeft klanten ook inzicht in de benodigdheden voor het realiseren en implementeren van het systeem. Hierbij heb ik duidelijk aangegeven dat Matthat Software B.V. leidend is in het vastleggen van voorwaarden over de structuur van de service messages door middel van een WSDL.*

De web service wordt beschikbaar gesteld aan websites voor productiedoeleinden en aan testtools zoals bijvoorbeeld soapUI voor testdoeleinden. Communicatie zal verlopen via de communicatietechnieken SOAP of XML-RPC over de transportprotocollen HTTP of HTTPS. Beide technieken en protocollen zullen worden aangeboden. Als gekozen wordt voor een HTTPS verbinding zullen de gegevens veilig getransporteerd worden. Hiernaast zorgt het systeem voor bescherming van gegevens. Dit zal betekenen dat er geen gegevens aangeboden zullen worden door de web service en het systeem waar een actor geen rechten toe heeft.

- *Dit sluit aan op het eerste gedeelte over het aanbieden van een web service vanuit Pluriform. Het beantwoordt vragen van klanten over de manier waarop de communicatie kan verlopen tussen website en Pluriform. Deze functionaliteit komt tegemoet aan de eisen van de klant over een beveiligde verbinding en het systeem eenvoudig kunnen implementeren en aansluiten op een website.*

Het systeem zal de volgende functionaliteit aanbieden:

1. Een gebruiker authentiseren op gebruikersnaam en wachtwoord
2. Het ophalen en versturen van persoonsgegevens voor een specifiek persoon
3. Het wijzigen van persoonsgegevens voor een specifiek persoon
4. Het ophalen en versturen van adresgegevens voor een specifiek persoon
5. Het toevoegen van een adres aan een specifiek persoon
6. Het wijzigen van adresgegevens van een specifiek persoon

7. Aan de hand van een postcode en huisnummer adresgegevens opvragen uit AdressXpress en versturen
8. Het ophalen en versturen van donatiegegevens van een specifiek persoon
9. Het ophalen en versturen van toezeggingen van een specifiek persoon
10. Een wijzigingsvoorstel aanmaken en opslaan
11. Een specifiek wijzigingsvoorstel doorvoeren, verwijderen of annuleren
12. Instellingen op wijzigingsvoorstellen maken en opslaan
13. Een antwoord XML bericht maken en versturen

➢ *Door aan te geven wat de functionaliteiten van het systeem worden, kan de klant zich een beeld vormen over het systeem. Dit is belangrijk op het moment dat de klant interesse toont in het systeem. Hierbij kom ik tevens tegemoet aan de eisen van de klant en de afdeling research & development. Daarnaast zal deze functionaliteit een standaard opleveren die beschikbaar gesteld kan worden aan alle klanten.*

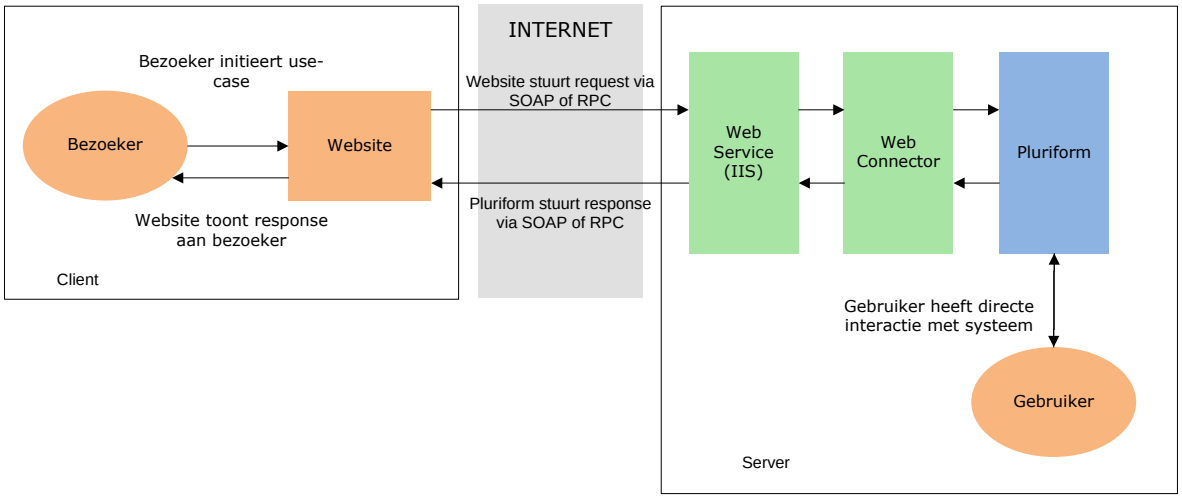
Binnen het systeem wordt de mogelijkheid ingevoerd om wijzigingsverzoeken vanaf een website eerst klaar te zetten in Pluriform als wijzigingsvoorstel. Voor het behandelen van een wijzigingsvoorstel zal er een knop in Pluriform aangemaakt worden. Als de gebruiker hier op klikt wordt een actiescherm geopend die functionaliteit aanbiedt voor het behandelen van een wijzigingsvoorstel. Alle wijzigingsvoorstellen zullen bewaard blijven in het systeem zodat deze altijd terug zijn te vinden. Hiernaast zal er een status worden toegevoegd aan een wijzigingsvoorstel.

➢ *Met het toevoegen van deze functionaliteit aan het systeem wordt er rekening gehouden met de eis van de klant en de afdeling research & development om wijzigingen te kunnen controleren en accorderen voordat deze worden doorgevoerd. Hiernaast wordt rekening gehouden met de eis om een wijzigingsvoorstel te kenmerken met een status.*

External interfaces

Voor het systeem wordt de client - server architectuurstijl toegepast. Hierbij dient de website van de klant als client en het systeem als server. Zoals eerder aangegeven zal de infrastructuur worden uitgebreid met een webserver waar IIS op draait. Binnen IIS zal er een web service worden ingericht. Daarnaast zal er een web connector worden geïnstalleerd die de communicatie tussen systeem en web service.

Om duidelijk te maken wie er bij een use-case betrokken is, hoe een use case geïnitieerd wordt en hoe de communicatie verloopt tussen systeem en actor heb ik een communicatie schema gemaakt. In het schema worden de componenten aangegeven die betrokken zullen zijn met het systeem:



Figuur 9 Communicatie tussen client en server

De bezoeker van de website initieert een van de use cases waardoor de website een request bericht verpakt in XML en over een HTTP of beveiligde HTTPS verbinding naar de web service stuurt. De web service verwerkt het bericht naar de web connector die het bericht direct doorstuurt naar Pluriform. In het Pluriform systeem wordt

vervolgens een verwerking gedaan, het antwoord gegenereerd dat ook verpakt is als XML bericht. Via de web connector en de web service wordt het antwoord bericht over HTTP of HTTPS verstuurd naar de website waar de website het antwoord bericht kan tonen aan de bezoeker.
Daarnaast heeft de gebruiker van Pluriform, om de use cases uit te voeren, directe interactie met het Pluriform systeem.

In Figuur 8 is te zien dat ik onderscheidt maak tussen bezoeker van een website en gebruiker van Pluriform. Beide actoren hebben hun eigen use-cases en daarom is het belangrijk om deze twee actoren apart te noemen. In dit verslag wordt onderstaande definitie aangehouden:

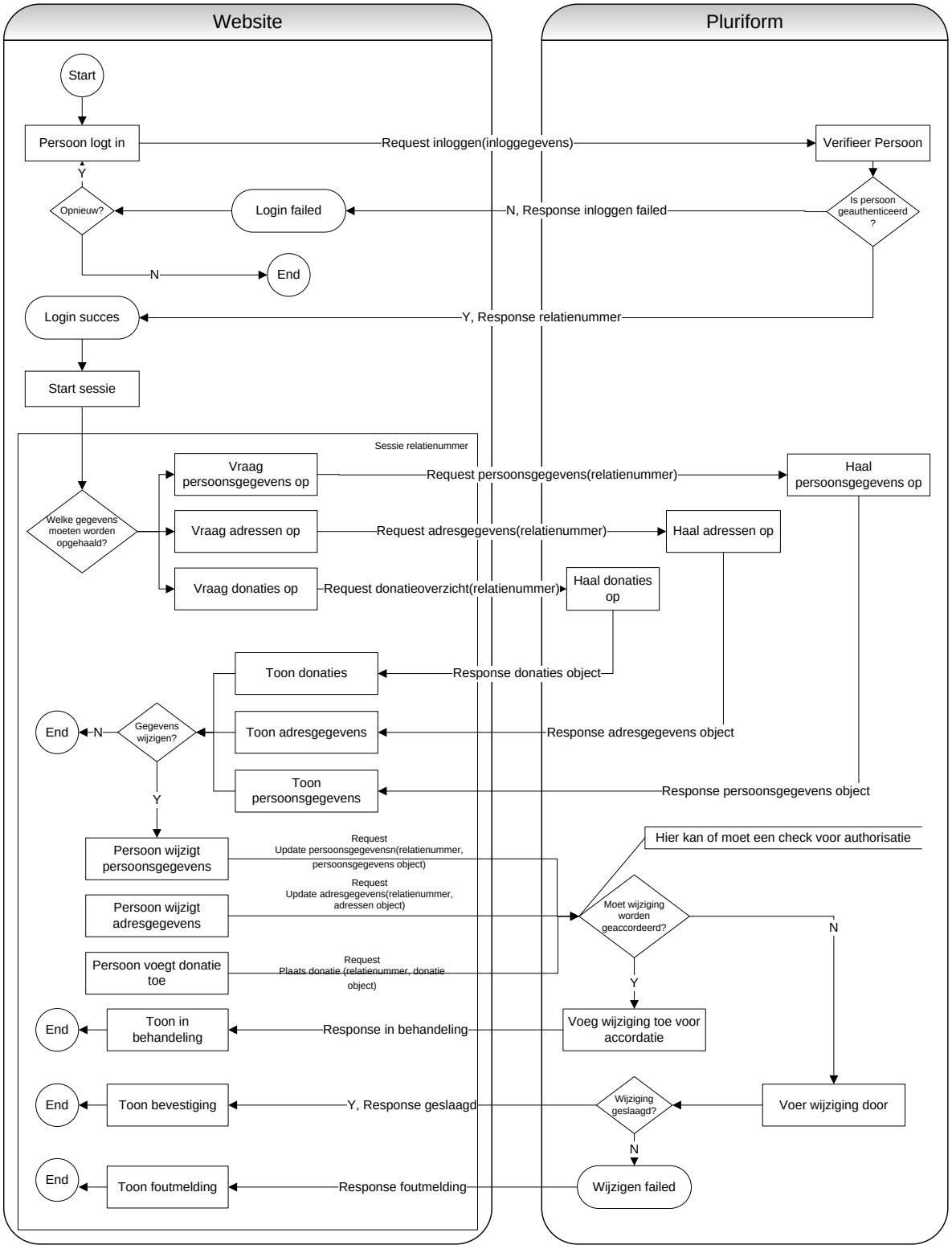
- **Bezoeker**
  De actor bezoeker is een persoon die geregistreerd staat in het Pluriform systeem van een klant van Matthat Software B.V. en inloggegevens heeft. Vaak is deze persoon iemand uit de doelgroep van een klant van Matthat Software B.V. Deze actor initieert use-cases naar het systeem door gebruik te maken van de website.

- **Gebruiker**
  De actor gebruiker is een werknemer/vrijwilliger van een klant van Matthat Software B.V. Deze actor initieert use-cases vanuit Pluriform direct met het systeem. Denk hierbij aan bijvoorbeeld het behandelen van een wijzigingsvoorstel.

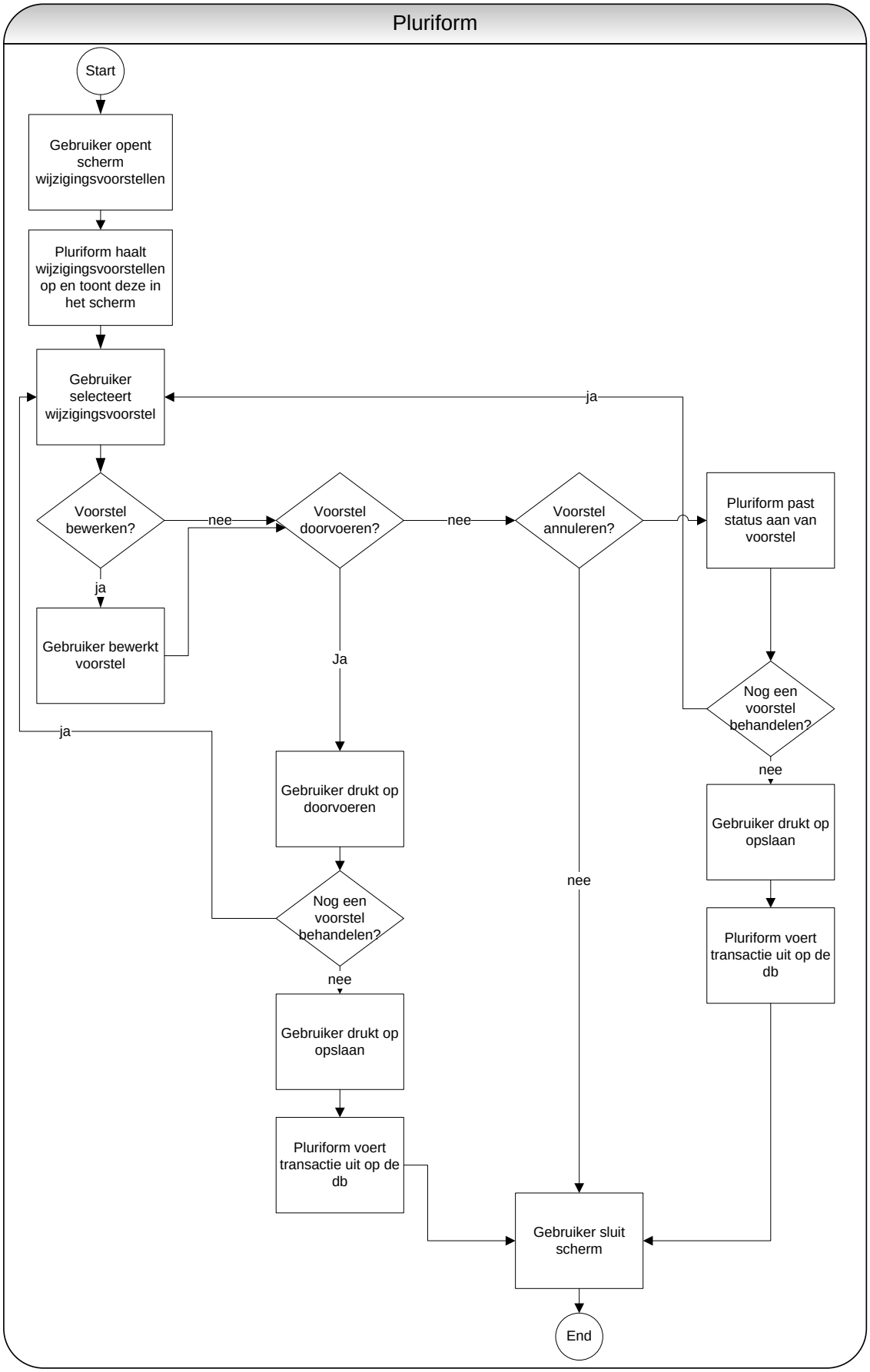
Internal Structure
Het Pluriform systeem is modulair opgebouwd. Dit betekent de module ‘XML Framework’ benodigd is voor het functioneren van het systeem. Deze module wordt geleverd uit Pluriform Basis en zal worden aangevuld met een stuk functionaliteit die wordt ondergebracht in een nieuwe module in het branchemodel van Matthat Software B.V. Zodra de functionaliteit van het systeem en de module ‘XML framework’ zal worden uitgebreid zullen beide modules dus groeien.

6.2.2 INFORMATION VIEW

In deze paragraaf wordt beschreven hoe de informatiestructuur geregeld wordt. De nadruk ligt hierbij op de drie scenario's uit paragraaf 6.1. In onderstaande figuren heb ik workflows gemaakt van client en server.



Figuur 10 Workflow inloggen, opvragen en wijzigen



Figuur 11 Workflow wijzigingsvoorstel behandelen

Naar aanleiding van de workflows en de scenario's uit pararaaf 6.1 heb ik gekeken naar de data die tussen client en server verstuurd wordt, hoe deze veranderen kan in het systeem en waar de data wordt opgeslagen.

Het ophalen van gegevens
In figuur 8 zijn de verschillende use-cases vastgelegd. Er zijn verschillende use-cases die als doel hebben om bepaalde gegevens uit Pluriform op te halen. Als voorbeeld wordt de use-case 'Geef overzicht van mijn persoonsgegevens'. Het voorbeeld is relevant voor de use-cases:

- Geef overzicht van mijn adres(sen)
- Geef overzicht van mijn donaties
- Geef overzicht van mijn toezeggingen

Beschrijving
1. Een bezoeker van de website geeft aan dat hij een overzicht van zijn persoonsgegevens wil inzien en initieert daarmee de website om een request naar Pluriform te versturen die zal bestaan uit een XML bericht waarin bijvoorbeeld zijn relatienummer is opgenomen
2. Het systeem ontvangt het XML bericht
3. Het systeem haalt, door gebruik te maken van het XML framework, het relatienummer uit het XML bericht
4. Het systeem haalt de persoonsgegevens data op uit de database van de persoon bij het opgegeven relatienummer uit het XML bericht
5. Het systeem stopt deze data in een nieuw XML bericht, door gebruik te maken van het XML framework
6. Het systeem stuurt als antwoord het nieuwe XML bericht (die de persoonsgegevens bevat) terug naar de website
7. Het systeem verwijdert zowel het request als antwoord XML bericht

Het wijzigen van gegevens
In figuur 8 zijn de verschillende use-cases vastgelegd. Er zijn verschillende use-cases die als doel hebben om bepaalde gegevens in Pluriform te wijzigen. Als voorbeeld wordt de use-case 'Wijzig mijn persoonsgegevens'. Het voorbeeld is relevant voor de use-cases:

- Wijzig mijn persoonsgegevens
- Wijzig mijn adresgegevens

Beschrijving
1. Een bezoeker van de website geeft aan dat hij zijn persoonsgegevens wil wijzigen en wijzigt deze op de website
2. Deze bezoeker zal op enig moment zijn wijzigingen willen doorvoeren en zal bijvoorbeeld drukken op een knop 'Opslaan'. Hiermee initieert de bezoeker de website om een request naar Pluriform te versturen die zal bestaan uit een XML bericht waarin zijn nieuwe persoonsgegevens staan en bijvoorbeeld zijn relatienummer
3. Het systeem ontvangt het XML bericht
4. Het systeem haalt, door gebruik te maken van het XML framework, het relatienummer uit het XML bericht
5. Het systeem haalt de persoonsgegevens data op van de persoon bij het opgegeven relatienummer uit het XML bericht en zet een lock op de data
6. Het systeem controleert elk persoonsgegeven uit het XML bericht of de waarde ervan daadwerkelijk een nieuwe waarde is. Dit doet het systeem door via het XML framework de waarde uit het XML bericht te halen en te vergelijken met de huidige persoonsgegevens
7. Hierna controleert het systeem elk persoonsgegeven uit het XML bericht of deze direct gemuteerd mag worden. Dit doet het systeem door de wijzigingsinstellingen op te halen uit de database en te controleren of het persoonsgegeven toestemming is verleend.
8. Het systeem kan nu drie dingen doen:

   a. Het persoonsgegeven direct muteren, als het een wijziging betreft en deze toestemming heeft. Hierdoor zal Pluriform direct een transactie uitvoeren op de database en de oude waarde van het persoonsgegeven overschrijven met de nieuwe waarde uit het XML bericht.
   b. Een wijzigingsvoorstel aanmaken voor het persoonsgegeven, als het een wijziging betreft en het geen toestemming heeft om direct gemuteerd te worden. Hierdoor zal Pluriform direct een nieuw object creëren en opslaan in de database als wijzigingsvoorstel.
   c. Het persoonsgegeven negeren, als het geen wijziging betreft.
9. Het systeem genereert door middel van het XML framework een XML bericht als antwoord waarin bijvoorbeeld een bevestiging wordt gestopt
10. Het systeem stuurt als antwoord het nieuwe XML bericht terug naar de website
11. Het systeem haalt de lock van de data
12. Het systeem verwijdert zowel het request als antwoord XML bericht

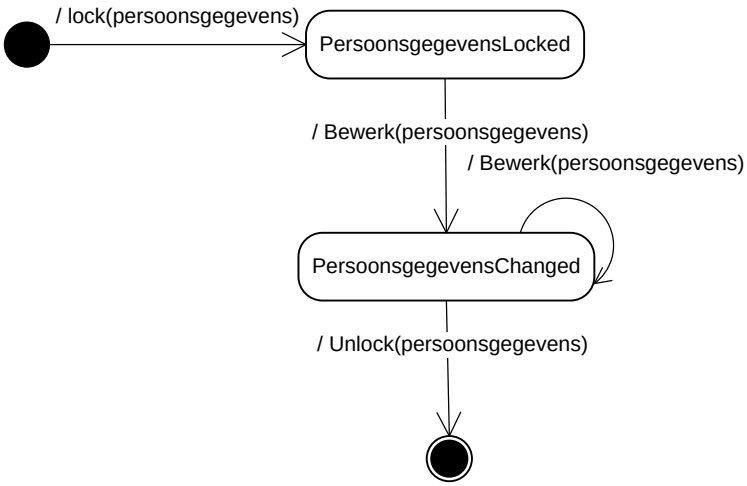
Het verwerken van een wijziging
In figuur 8 zijn de verschillende use-cases vastgelegd. De use-case 'Dit wijzigingsvoorstel behandelen' heeft betrekking op het verwerken van een wijziging. Het voorbeeld is relevant voor de use-cases:

- Dit wijzigingsvoorstel behandelen
- Wijzigingsinstellingen doorvoeren

Beschrijving
1. Een gebruiker van Pluriform opent het wijzigingsvoorstel scherm met bijvoorbeeld een knop
2. Het systeem selecteert uit de database alle wijzigingsvoorstellen met bijvoorbeeld een bepaalde status en zet een lock op de data.
3. Het systeem genereert een scherm en toont het object van wijzigingsvoorstellen bijvoorbeeld in een lijst.
4. Het systeem start een transactie op de database
5. De gebruiker selecteert een wijzigingsvoorstel uit de lijst en drukt op een knop bijvoorbeeld 'Verwerk'
6. Het systeem start een parallelle transactie en voert het wijzigingsvoorstel direct door in de database
7. De gebruiker selecteert een wijzigingsvoorstel en bewerkt bijvoorbeeld de nieuwe waarde van het voorstel
8. Het systeem slaat de uit te voeren actie op de database (de bewerking door de gebruiker) op
9. De gebruiker selecteert een wijzigingsvoorstel en bewerkt bijvoorbeeld de status
10. Het systeem slaat de uit te voeren actie op de database (de bewerking van de gebruiker) op
11. De gebruiker drukt op een knop bijvoorbeeld 'Toepassen'
12. Het systeem voert alle acties op de database uit in de openstaande transactie, stopt de transactie, haalt de lock van de data en sluit het scherm

Backup & recovery
Op het moment dat er een storing optreedt in een van de componenten van het systeem kan het voorkomen dat er verlies van data plaatsvindt. Om dit op te vangen wordt afgesproken dat het systeem elke 24 uur een back-up zal maken van de database. Deze back-up zal altijd plaats vinden in de nacht. Afspraken over de frequentie van het back-up proces zullen vastgelegd worden in een SLA.
Daarnaast zullen back-ups 6 maanden lang worden bewaard en zal het systeem recovery van deze back-ups mogelijk maken.

6.2.3 CONCURRENCY VIEW
Op het moment dat het systeem wijzigingen wil doorvoeren op bepaalde data uit de database zal het systeem zorgen voor een lock op deze data. Hierdoor wordt voorkomen dat twee clients bewerkingen uitvoeren op dezelfde data (zie figuur 12).



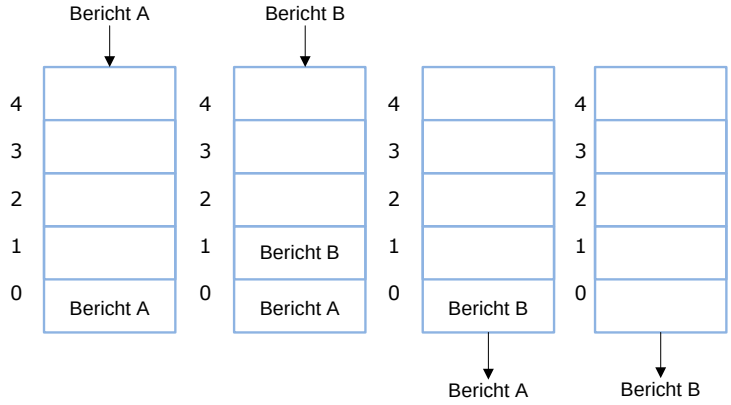
Figuur 12 Toestandsdiagram persoonsgegevens object

Er zijn twee scenario's waarin het voor kan komen dat twee clients data willen wijzigen in de database van Pluriform. Voor beide geldt de voorwaarde dat in de instellingen is opgenomen dat de betreffende wijziging direct gemuteerd mag worden:

1. Het kan op enig moment zijn dat een bezoeker op de website zijn gegevens wijzigt en verstuurd naar Pluriform en dat op hetzelfde moment een gebruiker van Pluriform deze gegevens al aan het bewerken is en de data dus gelockt is voor de gebruiker.
2. Op het moment dat er een back-up wordt gemaakt van de database en er een bezoeker op de website zijn gegevens wijzigt en verstuurd naar Pluriform.

Ook al is het risico erg klein dat een van deze scenario's optreedt, toch moet ik hier rekening mee houden en het systeem dit laten afvangen. Normaal leveren beide scenario's een foutmelding op in Pluriform, maar dit zal het systeem opvangen door op het moment van optreden de wijzigingen die de bezoeker wil doorvoeren op te slaan als wijzigingsvoorstel.

Om te voorkomen dat bezoekers niet hoeven te wachten tot de web service en het systeem een XML bericht van andere bezoekers hebben afgehandeld voordat ze een XML bericht kunnen indienen via de website zullen de XML berichten asynchroon worden uitgevoerd. Het systeem zal er voor zorgen dat XML berichten terecht komen in een wachtrij waar een First In First Out principe geldt. Dit betekent dat de XML berichten worden afgehandeld op volgorde van binnenkomen (zie figuur 13).



Figuur 13 First in First out principe

6.2.4 DEVELOPMENT VIEW

Het systeem wordt ontwikkeld in het branchemodel Goede Doelen van Matthat Software B.V. Hierbij zal het systeem gebruik maken van componenten uit de module 'XML framework' die is opgenomen in Pluriform basis. Daarnaast wordt gebruik gemaakt van de geïntegreerde modelleeromgeving Pluriform Studio om het systeem te ontwikkelen en te testen.

Ontwikkelproces
Het ontwikkelproces zal gebeuren in twee iteraties. In de eerste iteratie zal de architectuur worden getest en zal er een Proof of Concept ontwikkelt worden waarin zowel de SOAP- als de XML-RPC techniek getest zullen worden. Aan het eind van de eerste iteratie zal er 1 van deze technieken gekozen worden. In de tweede iteratie wordt met de gekozen techniek het systeem ontwikkelt. Na de iteraties wordt het systeem als update ingelezen in de testomgeving van de klant die vervolgens de ontwikkelde functionaliteit zelf kan testen.

Versiebeheer
Klanten hebben twee mogelijkheden:

- Van release naar release. Het voordeel hiervan is een stabiele versie, een nadeel is dat je niet beschikt over de meest recente versie.
- Meegaan in de zogeheten kopgroep/innovatierelease. Het voordeel hiervan is dat je beschikt over de meest recente versie, een nadeel kan zijn dat onderdelen in ontwikkeling niet altijd stabiel zijn. Als een klant in de kopgroep zit, kan het zijn dat er zelfs dagelijks een update van het systeem wordt ingelezen. Daarnaast is vastgesteld dat er minimaal 1 keer in de 2 weken een update wordt ingelezen bij de klant.

6.2.5 DEPLOYMENT VIEW

Omdat het de test/ontwikkelomgeving van de klant een exacte kopie is van de productieomgeving zal het uitrollen van het systeem geen verrassingen opleveren. Fouten in het systeem zullen al in de testomgeving worden ontdekt en verholpen worden. Een randvoorwaarde die ik stel aan het uitrollen van het systeem is dat de productieomgeving wordt geupdate met dezelfde versie als waar mee is getest.

Systeemeisen
Pluriform heeft een client-server architectuur waarin meerdere werkstations (de clients) en eventuele webservers via een snel netwerk toegang hebben tot een gemeenschappelijke database server. Welke hardware er nodig is voor de server en het netwerk hangt sterk af van het aantal clients en de intensiteit van gebruik. In elk geval is het aan te bevelen dat de database server hardware alleen voor Pluriform Server (of eventueel Pluriform clients) wordt gebruikt, en dus niet voor andere applicaties.

Daarnaast zijn de volgende softwarecomponenten vereist:

- IIS (Internet Information Services) 6.0 of hoger en een web service
- Pluriform Goede Doelen versie 2011.2*pre* of later waarin de module *XML Framework* is opgenomen.
- Pluriform web connector.

Technology compatibility
Servers gebaseerd op virtualisatie software worden niet ondersteund door het systeem. Hiernaast worden Microsoft Windows 2000 en Office 2000 niet meer ondersteund.

Installation en updates
De web connector wordt geïnstalleerd met behulp van een installer. Deze connector wordt afzonderlijk geupdate. Zodra Pluriform wordt geupdate wordt de module 'XML framework' en het systeem automatisch meegenomen in de nieuwe versie van Pluriform. Het updateproces wordt verzorgd door Matthat Software B.V. Ook de installatie van de web service kan verzorgd worden door Pluriform, maar gebeurt vaak door de systeembeheerder van de klant. Het uitbrengen van updates voor de web service software wordt wel standaard verzorgd door Matthat Software B.V.

Licensing
Matthat Software B.V. biedt twee verschillende licentie mogelijkheden aan:

- *Naar ratio van gebruik*. Als een gebruiker of bezoeker het systeem één of meer keer gebruikt in een periode van 10 minuten (een 'timeslot') wordt voor deze periode een bepaald bedrag gerekend. Afrekening per maand achteraf. Deze methode is met name interessant bij minder intensief gebruik van het systeem. Dit betekent weinig verzoeken vanaf de client naar de server.

- *Voor een maximum aantal gelijktijdige sessies*. Het licentietarief is gebaseerd op het maximaal aantal gelijktijdige systeem sessies dat actief kan zijn. Afrekening per maand achteraf. Deze methode is met name aantrekkelijk bij intensief gebruik van het systeem. Dit betekent veel verzoeken vanaf de client naar de server.

6.2.6 OPERATIONAL VIEW
Bij het verwerken van binnenkomende XML berichten zal het systeem gebeurtenissen en fouten loggen. Hiermee kunnen oorzaken van fouten snel worden opgespoord en kan de communicatie tussen client en server in de gaten worden gehouden.

Voor het monitoren van de webserver kan gebruik worden gemaakt van de monitoring tools binnen IIS.

Voor het monitoren van de web connector zal het systeem een zogenoemde control center aanbieden.

Omdat elke 24 uur een back-up wordt gemaakt van het systeem is het mogelijk om een rollback te doen van oudere back-ups. Het inlezen van een back-up duurt gemiddeld 30 minuten waarin het systeem niet operationeel is. Hierom worden updates en back-ups altijd in de nacht ingelezen.

De beschikbaarheid van het systeem in productie omgevingen is getest onder verschillende condities. Meer hierover is te vinden in hoofdstuk 7, 8 en 9.

6.3 SYSTEM QUALITIES

6.3.1 SECURITY

Scenario 4#
Overview: Hoe het systeem omgaat met de beveiliging van de web server en gegevens
System state: De web server is ingericht om een HTTPS verbinding op te zetten met een client
System environment: Het systeem is operationeel en biedt een authenticatie op SOAP niveau
External stimulus: Een bezoeker initieert via de website een van de XML berichten uit het systeem
Required system response: Het systeem zorgt ervoor dat de bezoeker geauthentiseerd wordt en de web server zorgt ervoor dat er een HTTPS verbinding actief is tussen server en client

Het systeem biedt mogelijkheden om via een beveiligde verbinding (HTTPS) gegevens uit te wisselen. Hiernaast zal het systeem een authenticatie functionaliteit aanbieden waardoor gebruikers zich moeten authentificeren met gebruikersnaam en wachtwoord. Daarnaast is er ook een verantwoordelijkheid van de website om bijvoorbeeld in een beveiligde sessie het systeem aan te roepen.
Het systeem ondersteund ook authenticatie als er gebruik wordt gemaakt van het SOAP protocol, wat duidelijk zal worden na de eerste iteratie. Dit houdt in dat elke keer als er een SOAP XML bericht naar de web service wordt gestuurd, het bericht een gebruikersnaam en wachtwoord moet bevatten die het systeem in staat stelt de gebruiker te authentificeren.

6.3.2 PERFORMANCE AND SCALABILITY

Scenario 5#
Overview: Hoe het systeem omgaat met piekbelasting van de web connector
System state: De web connector is operationeel
System environment: Het systeem is operationeel
External stimulus: Een grote hoeveelheid bezoekers initieert via de website een van de XML berichten uit het systeem
Required system response: De web connector zorgt voor een afhandeling van de aanvragen vanaf de client met een maximale afhandelingstijd van een seconde per aanvraag

Onder andere response tijden van de server zijn afhankelijk van verschillende factoren waaronder hardware en breedband snelheden. Binnen Pluriform is het mogelijk om meerdere web connectors naast elkaar te draaien. Zo kan het aantal calls naar de server verdeelt worden. Ik heb besloten om in de testfase de performance van de web connector te testen (zie paragraaf 7.3).

6.3.3 AVAILABILITY

Scenario 6#
Overview: Hoe het systeem omgaat met het stilvallen van een server, web server, hosting provider of breedband verbinding
System state: De web server, server, hosting provider en breedbandverbinding zijn operationeel
System environment: Het systeem is operationeel
External stimulus: Door verschillende omstandigheden valt een van de componenten stil.
Required system response: Het systeem heeft geen invloed op al deze componenten. De website zou een cache kunnen aanmaken.

De web connector, IIS en het Pluriform systeem moeten 24 uur per dag, 7 dagen per week beschikbaar zijn om het systeem operationeel te houden. Uitzondering hierop zijn geplande werkzaamheden zoals updates of onderhoud, of storingen.
Als er storingen ontstaan kan de website verantwoording nemen en een cache aanmaken die, als het systeem weer operationeel is, verstuurd wordt naar Pluriform om alsnog afgehandeld te worden. Daarnaast kan de website meldingen tonen op het moment dat er sprake is van storingen.

6.4 VAN ARCHITECTUUR NAAR ITERATIE 1

Het Pluriform systeem biedt uit de basis laag de modules ‘Web – basis’ en ’ ‘XML framework’ aan. Beide frameworks bevatten bouwstenen voor het ontwikkelen van het systeem. Eén van de grootste verschillen tussen beide frameworks is het protocol voor communicatie tussen client en server. De module ‘Web - basis’ ondersteunt het XML-RPC protocol en de module ‘XML framework’ ondersteund het protocol SOAP. Voordat het systeem volledig ontwikkeld kan worden ga ik een keuze maken tussen beide protocollen.

Om een keuze te maken tussen de communicatieprotocollen SOAP en XML-RPC heb ik besloten om een Proof of Concept te ontwikkelen van het systeem waarin ik beide protocollen zal gebruiken. Door beide protocollen te gebruiken kan ik de voor- en nadelen van beide tegenover elkaar zetten, de werking van de protocollen leren en bewijs leveren voor de haalbaarheid van het gekozen protocol. Daarnaast heb ik de objecten in beide modules toegepast en kan ik die kennis in de tweede en derde iteratie gebruiken bij het technisch ontwerp van het systeem.

Bovenstaand heb ik gedaan in een eerste iteratie van het project. In deze iteratie doorloop ik een elaboration- en construction fase, om vervolgens een conclusie te kunnen trekken op basis van deze uitgevoerde fases. In de elaboration fase heb ik eerst een functioneel en technisch ontwerp voor het POC gemaakt. Aan het eind van

de eerste iteratie zullen, op basis van mijn conclusie, deze ontwerpen, de requirements en de architectuur groeien.

6.4.1 EVALUATIE

Tijdens het opstellen van de architectuur heb ik veel kennis van het I6 blok 'Software architectuur' van de Haagse Hogeschool kunnen toepassen. Door gebruik te maken van het boek 'software systems architecture' van Nick Rozanski en Eoin Woods (Rozanski) kon ik duidelijk stap voor stap het ontwerp van het systeem vanuit een architectuur oogpunt behandelen. Ik heb geleerd om de functionele aspecten van het systeem te scheiden van het architectuur ontwerp. In eerste instantie dacht ik veel te functioneel na en stopte daarom heel veel functionele aspecten in het architectuur ontwerp. Ik heb daar samen met een collega over gesproken en hij heeft mij geholpen om vanuit het architectuur oogpunt te ontwerpen. Daarnaast heb ik geleerd om vanuit verschillende viewpoints het systeem te ontwerpen, waarbij ik rekening hield met de functionele eisen en de verschillende kwaliteitsattributen. Doordat er klanten waren die met generieke problemen naar mij toe kwamen kon ik die verwerken in een scenario en een oplossing bedenken. Uiteindelijk heb ik een duidelijke architectuur kunnen opstellen die begrijpelijk is voor alle stakeholders. Kort samengevat:

- Een client – server architectuur die ontwikkeld wordt in korte iteraties
- Externe componenten zijn een web server met web service, een web connector en een client in de vorm van een website
- Interne componenten zijn de onderdelen uit de module 'XML framework'
- Het systeem ondersteunt straks wijzigen, ophalen en plaatsen van gegevens in het Pluriform systeem
- Er is sprake van concurrency in twee scenario's, die afgevangen gaan worden door het systeem
- Het systeem biedt beveiligingsmogelijkheden
- De werking van het systeem wordt afhankelijk van verschillende componenten waaronder een web server, hosting provider en breedbandverbinding
- Het systeem kent de actoren bezoeker website en gebruiker Pluriform

7 ITERATIE 1# ONTWIKKELEN VAN EEN PROOF OF CONCEPT

Na het opstellen van de requirements is het essentieel om te achterhalen welk communicatieprotocol toegepast zal worden bij het ontwikkelen van het systeem. Hierbij moet ik kiezen tussen XML-RPC of SOAP, de twee protocollen die ondersteund worden door Pluriform. Daarom zal ik in deze eerste iteratie een Proof of Concept ontwikkelen. In deze iteratie is het doel om beide communicatieprotocollen te gebruiken, de voor- en nadelen van beide tegenover elkaar te zetten, een keuze te maken en de haalbaarheid van het gekozen protocol aantonen voor de afstudeeropdracht. Door beide protocollen te testen weet ik straks welke objecten ik kan gebruiken bij het technisch ontwerp van het systeem.

Het hoofdstuk is opgedeeld in drie fases van Unified Process, namelijk elaboration fase (7.1), construction fase (7.2) en transition fase (7.3).

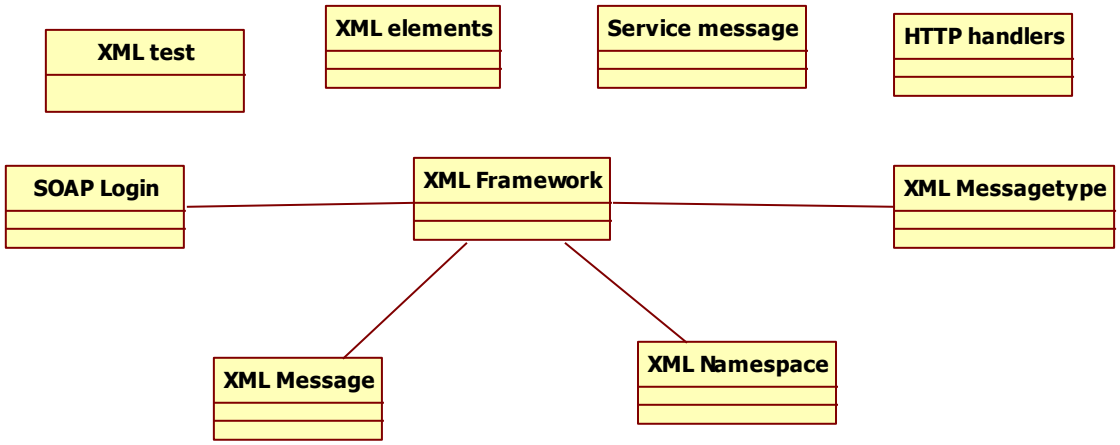
7.1 UITVOEREN ELABORATION FASE: ANALYSIS AND DESIGN

Voor het Proof of Concept wil ik de functionaliteit ‘Het ophalen en versturen van persoonsgegevens voor een specifiek persoon’ ontwerpen en ontwikkelen. Deze functionaliteit heb ik gekozen omdat het als één van de meest gewenste functionaliteiten werd genoemd in het klantonderzoek.

7.1.1 DE BESCHIKBARE PLURIFORM COMPONENTEN VOOR SOAP EN XML-RPC

Voordat ik een functioneel ontwerp maak voor het POC ga ik de componenten uit de twee modules ‘Web - Basis’ en ‘XML framework’ analyseren. Zo weet ik wat elke module aanbiedt en wat ik kan gebruiken voor de ontwikkeling van het POC.

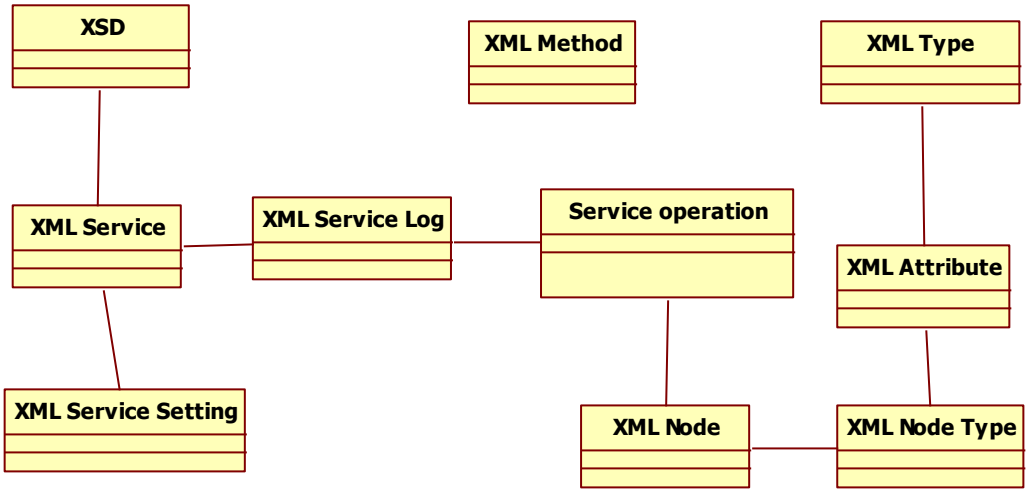
De module Web – Basis



Figuur 14 Domeinmodel module 'Web - Basis'

De module omvat alle klassen die opgenomen zijn in figuur 14, De module biedt daarnaast het XML-RPC protocol voor het ontvangen en versturen van XML berichten.

De module XML framework



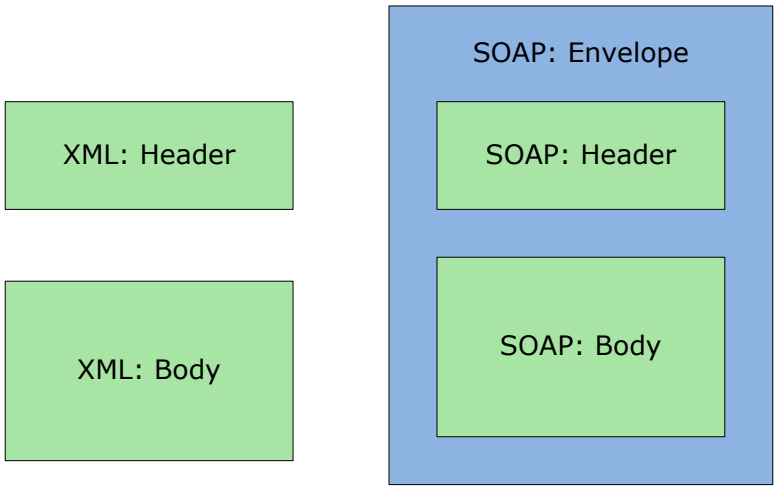
Figuur 15 Domeinmodel module 'XML framework'

De module omvat alle klassen die opgenomen zijn in figuur 15. De module biedt daarnaast het SOAP protocol voor het ontvangen en versturen van XML berichten.

SOAP is een protocol dat wordt gebruikt voor communicatie tussen verschillende componenten en is de opvolger van XML-RPC. Het idee achter RPC komt uit het jaar 1976, hanteert een eenvoudigere structuur dan SOAP maar is een verouderd protocol. Beide protocollen sturen XML-berichten, meestal over HTTP, maar ook over SMTP, HTTPS of FTP. Het SOAP protocol bestaan uit drie onderdelen:

- Een 'Envelope' element dat het XML document identificeert als een SOAP message. Dit moet altijd aangegeven worden met een namespace: xmlns=http://www.w3.org/2003/05/soap-envelope.
- Een 'Header' element, waarin credentials kunnen worden ingegeven.
- Een 'Body' element die alle request en response informatie bevat.
- Een 'Fault' element die error en status informatie bevat

SOAP-implementaties zijn beschikbaar voor vele verschillende talen en omgevingen, zodat ontwikkelaars zich niet hoeven te bekommeren over het formaat van SOAP-berichten, over de wijze van versturen en hoe foutcorrectie moet worden toegepast.



Figuur 16 RPC en SOAP structuur

Als een XML bericht is ontwikkeld zal Pluriform automatisch een WSDL genereren. Deze Web Service Description Language (WSDL) is een document geschreven in XML. Het document beschrijft de web service, specificeert de locatie van de web service en beschrijft de inhoud van de XML berichten die het systeem aanbiedt. In de WSDL zijn vier elementen te onderscheiden:

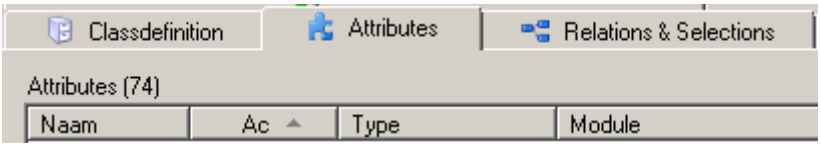
- Types. De datatypes die gebruikt worden door de web service
- Message. De inhoud van de messages die gebruikt worden door de web service
- portType. De operaties die uitgevoerd worden door de web service
- Binding. De communicatie protocollen die gebruikt worden door de web service

Het voordeel van een WSDL is dat ontwikkelaars uit de WSDL kunnen aflezen hoe een client ontwikkelt kan worden om het systeem te benaderen.

7.1.2 HET FUNCTIONELE ONTWERP VAN HET PROOF OF CONCEPT

Een persoon moet via een website zijn of haar specifieke persoonsgegevens kunnen opvragen. Deze persoonsgegevens bestaan minimaal uit de volgende onderdelen: initialen, voornaam, achternaam, aanspreektitel, geslacht, geboortedatum, telefoonnummer(s), emailadres en nationaliteit. Het systeem zal hiervoor een service moeten aanbieden waardoor de website het systeem kan aanroepen. Het is van belang dat het systeem de persoon kan authentificeren. Het systeem moet vervolgens na authenticatie zorgen voor een antwoord in de vorm van een XML bericht op de aanvraag van de website.

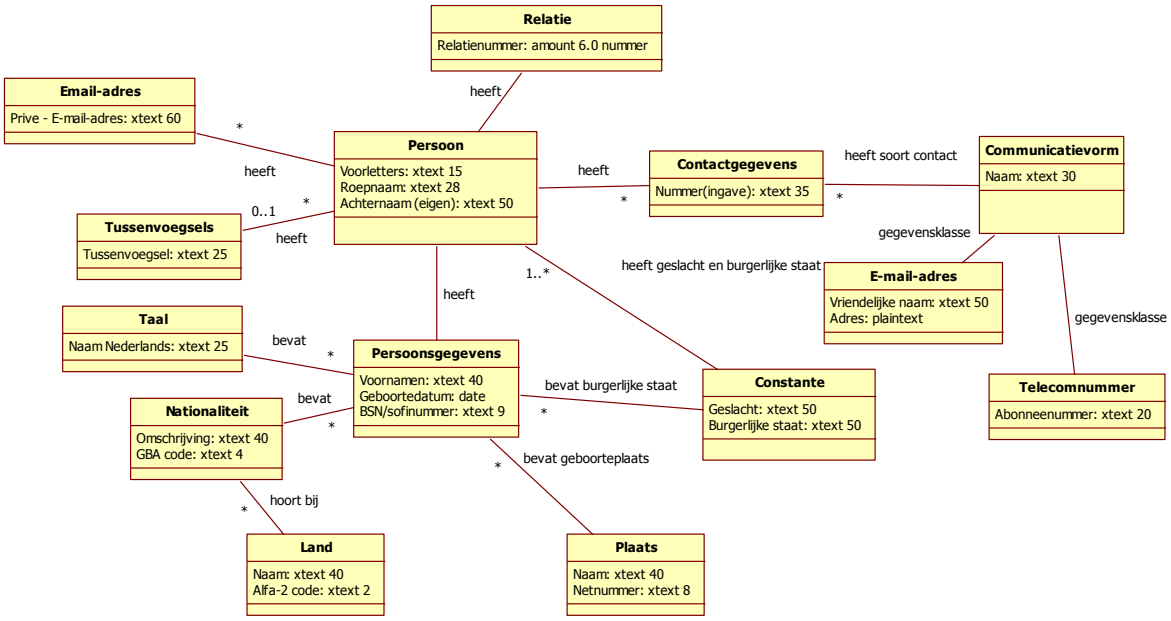
Bovenstaande functionele beschrijving voor het POC heb ik gebruikt om een domeinmodel op te stellen voor de case 'Het ophalen en versturen van persoonsgegevens voor een specifiek persoon'. Voor deze case volstaat het om binnen Pluriform reverse engineering toe te passen op de bestaande klasse Persoon. Reverse engineering in Pluriform kan ik doen door naar een specifieke klasse te navigeren en daarvan de definitie op te vragen. Dit geeft een duidelijk beeld van de klasse, zijn attributen met types en relaties met andere klassen (zie figuur 17).



Figuur 17 Reverse engineering in Pluriform

Door vervolgens gebruik te maken van de tool StarUML kon ik een model opstellen die het huidige beeld in Pluriform weergeeft. In het model leidt ik af dat een persoon een uniek relatienummer heeft. Het systeem zou dus aan de hand van dit relatienummer een persoon kunnen authentificeren. In het model (zie figuur 18) heb ik methodes en constructors weg gelaten omdat ik hier in het technisch ontwerp over zal nadenken (later in dit hoofdstuk).

Hieronder is het domeinmodel voor persoonsgegevens met de verschillende klassen en hun onderlinge relaties.



Figuur 18 Domeinmodel persoonsgegevens

Bij het maken van een use-case voor ‘geef overzicht van mijn persoonsgegevens’ ga ik uit van de gebruiker van het systeem. Hierbij probeer ik te beschrijven hoe de gebruiker om wil gaan met het systeem. De gebruiker in deze use-case is de bezoeker van de website. De website is het onderdeel wat het systeem aanroept. In dit geval is de actor de bezoeker van de website. Zie tabel 3 voor de use-case ‘geef overzicht van mijn persoonsgegevens’

Naam	Geef overzicht van mijn persoonsgegevens
Samenvatting	De functionaliteit zorgt voor het ophalen van persoonsgegevens uit Pluriform aan de hand van een relatienummer en het versturen van een XML bericht met deze persoonsgegevens.
Aannamen	- Service is bereikbaar over HTTP - Relatienummer van actor is bekend in de sessie van de website
Beschrijving	(1) Actor geeft aan zijn persoonsgegevens te willen ophalen via de website (2) De website stuurt een XML bericht naar het systeem met als input een relatienummer (3) Het systeem checkt of het relatienummer bestaat, anders treedt uitzondering [relatienummer onbekend] op. (4) Het systeem zoekt de desbetreffende persoon op (5) Het systeem plaatst alle persoonsgegevens uit het opgehaalde object zoals bekend in Pluriform in vooraf gedefinieerde XML velden van het antwoord XML bericht (6) Het systeem stuurt het antwoord XML bericht van de operation getPerson terug naar de website
Uitzonderingen	[relatienummer onbekend] Als het opgegeven relatienummer niet bestaat stuurt het systeem een foutmelding als antwoord XML bericht terug naar de website
Resultaat	Website ontvangt het antwoord XML bericht en actor heeft inzicht in zijn persoonsgegevens zoals deze bekend zijn binnen Pluriform

Tabel 3 Use-case beschrijving ‘Geef overzicht van mijn persoonsgegevens

Het uiterlijk van het binnenkomende en uitgaande XML bericht op het systeem zal er als volgt uit moeten zien:

Binnenkomend XML bericht:

```
<methodCall>
 <methodName>getPerson</methodName>
 <params>
  <param>
   <value><int></int></value>
  </param>
 </params>
</methodCall>
```

Uitgaand XML bericht:

```
<methodResponse>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>Voorletters</name>
            <value>
              <string></string>
            </value>
          </member>
          <member>
            <name>Achternaam</name>
            <value>
              <string></string>
            </value>
          </member>
          <member>
            <name>Voornaam</name>
            <value>
              <string></string>
            </value>
          </member>
          <member>
            <name>Aanspreektitel</name>
            <value>
              <string></string>
            </value>
          </member>
          <member>
            <name>Geslacht</name>
            <value>
              <string></string>
            </value>
          </member>
          <member>
            <name>Geboortedatum</name>
            <value>
              <string></string>
            </value>
          </member>
          <member>
            <name>Telefoonnummer</name>
            <value>
              <string></string>
            </value>
          </member>
          <member>
            <name>Emailadres</name>
            <value>
              <string></string>
            </value>
          </member>
```

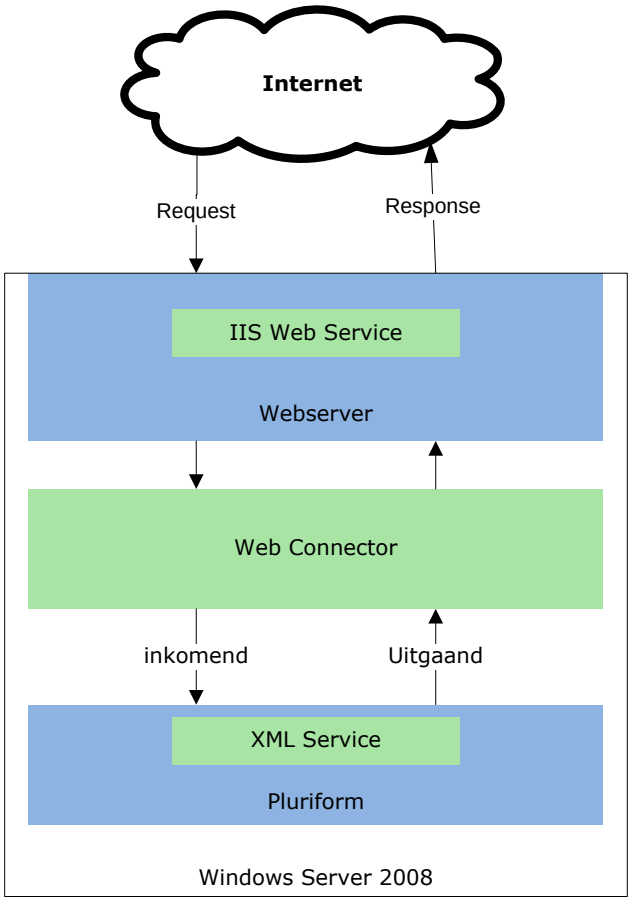
```
        <member>
          <name>Nationaliteit</name>
          <value>
            <string></string>
          </value>
        </member>
      </struct>
    </value>
  </param>
 </params>
</methodResponse>
```

7.2 UITVOEREN CONSTRUCTION FASE: DESIGN AND IMPLEMENTATION

7.2.1 HET UITROLLEN VAN EEN WEB SERVICE EN EEN WEB CONNECTOR

Pluriform maakt het mogelijk om installers te exporteren naar een gewenste locatie. Hierdoor is het mogelijk om een web service en een web connector uit te rollen. Voor het uitrollen heb ik gebruik gemaakt van de webserver van Matthat Software B.V. en Internet Information Services (hierna genoemd IIS).

De installers creëren een aantal Dynamic-Link Library (hierna genoemd DLL) files die ervoor zorgen dat verzoeken vanaf het web op de juiste manier gerouteerd worden door IIS naar het systeem. Figuur 19 toont hiervan een schematische weergave. In IIS heb ik een site aangemaakt, de DLLs toegevoegd aan de site en toestemming verleend om deze DLLs uit te voeren. Bij het aanmaken van een site in IIS op de webserver krijgt de site een HTTP poort toegewezen, waardoor deze bereikbaar wordt vanaf het internet.



Figuur 19 Architectuur web- service en connector

7.2.2 HET MODELLEREN VAN HET PROOF OF CONCEPT MET XML-RPC

Voor het ontwikkelen van XML-RPC berichten wordt gebruik gemaakt van de klasse Service Messages. Een nieuwe Service Message verwacht een aantal onderdelen:

- ServiceID. De naam van de message.
- Parameter struct. Dit is een object van het type struct waarin de input parameters opgegeven moeten worden.
- Result struct. Dit is ook een object van het type struct waarin de output, het resultaat van de aanvraag, moet worden opgegeven.
- Transformationscript. Een Pluriform script waarin de logica zit. Dit zorgt ervoor dat een aanvraag op de juiste manier wordt afgehandeld.
- Developer description. Een beschrijving van de Service Message.
- Authentication type. Hierbij zijn drie opties mogelijk: Anonymous, Basic of Integrated Windows.

Een struct is een gestructureerd object die verschillende typen objecten kan bevatten. Voor het parameter struct is het de bedoeling dat de aanvrager een relatienummer als parameter ingeeft. Uit het ontwerp blijkt dat relatienummer in de klasse Relatie zit en een relatie heeft met de klasse Persoon. Daarom kan er straks in het transformation script gezocht worden naar een persoon. Ik heb in de klasse struct een nieuw struct aangemaakt met de naam getPerson en bevat één object van het type amount genaamd relatienummer:

Struct getPerson
relatienummer integer

Het parameter struct heb ik ontworpen aan de hand van de XML die ik verwacht bij binnenkomst. Dit betekent dat ik dus door middel van het definiëren van structs het uiterlijk kan bepalen van een XML bericht.

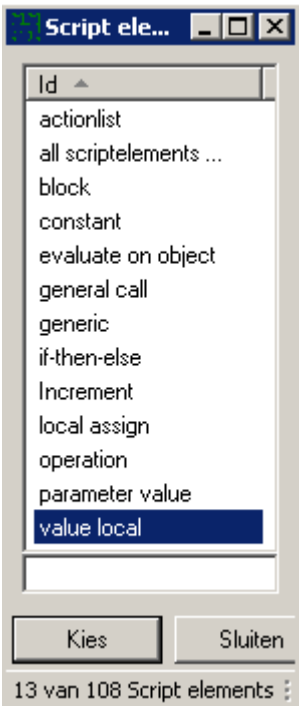
Het result struct bevat één object genaamd personResult (dit is het object dat er gevraagd wordt) ook van het type struct. Het struct persoon bevat een substruct persoonsgegevens met vervolgens verschillende objecten zoals opgegeven in het ontwerp zoals voorletters, roepnaam en achternaam:

Struct personResult
Substruct gegevens
structmember Initials string
structmember Nickname string
structmember Surname_own string
structmember Address_as string
structmember Sex string
structmember Date_of_birth date
structmember Civil_status string
structmember Private_e-mail_address string
structmember Nationality string

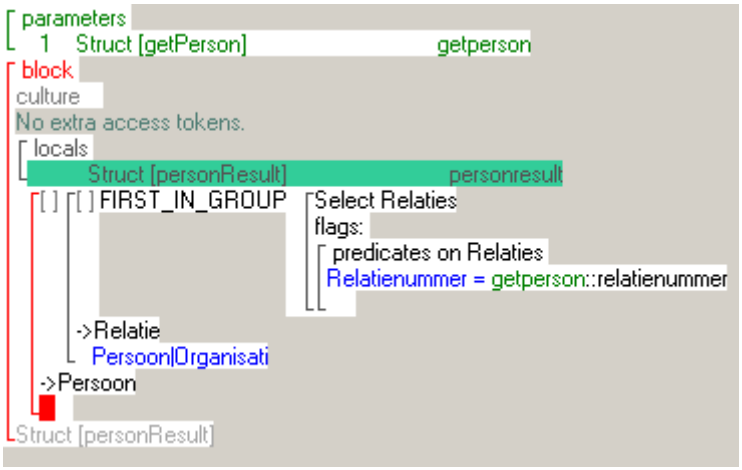
Nu de input en output van de Service Message gedefinieerd zijn, moet ik gebruik maken van een Transformation script om het binnenkomende XML bericht te kunnen afhandelen en het uitgaande bericht te kunnen genereren. Het Pluriform script is een volledig unieke taal en is anders van aard als dat ik geleerd heb op school.

Pluriform script

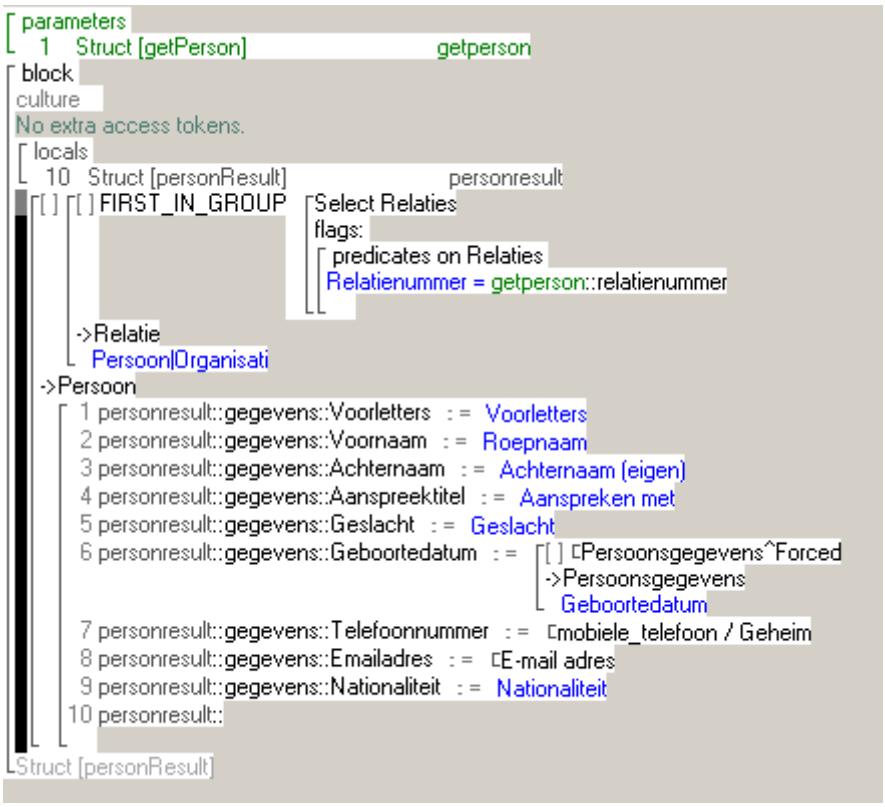
Het opstellen van een script in Pluriform gebeurt in een dialoog scherm waar gekozen kan worden uit voor gedefinieerde script elementen (zie Figuur 20). Het is de bedoeling dat eerst de juiste persoon wordt opgezocht aan de hand van een relatienummer:



Figuur 20 Script elementen



Op dit punt in het script zit ik in de context van een persoonsobject die gerelateerd is aan een relatienummer. Vervolgens kan ik alle attributen, oftewel persoonsgegevens, van deze persoon ophalen en in het result struct plaatsen:



Het transformation script zorgt er nu voor dat aan de hand van een relatienummer alle gewenste gegevens worden opgehaald en in het struct persoon worden gestopt. Het resultaat is een gevuld struct die geretourneerd wordt.

7.2.3 HET MODELLEREN VAN HET PROOF OF CONCEPT MET SOAP

Voor het ontwikkelen van SOAP berichten wordt gebruik gemaakt van de klasse Service Operations. Een Service Operation valt onder een zogeheten 'XML Service'. Het definiëren van een XML service gebeurt in Pluriform in de klasse XML Service. Een XML service is een service die verschillende service operations aanbiedt via een URL. Bij het aanmaken van deze service moeten een aantal dingen worden ingesteld:

- Een service ID
- De mogelijkheid om de service actief te maken door het aanvinken ervan.
- Opgeven van te ondersteunen web service bindings
- Opgeven van te ondersteunen autorisatie mogelijkheden
- De URL naar de WSDL opgeven
- De WSDL namespace opgeven

Ik heb gekozen voor een service ID genaamd 'SOAPXMLService' met ondersteuning voor SOAP binding. Dit houdt in dat de XML berichten opgezet worden volgens een opgezet XML schema. Dit schema beschrijft de structuur van een service operation en wordt vastgelegd in een WSDL. Vervolgens heb ik een URL aangegeven in de WSDL namespace waar de WSDL te bereiken is via het web.
Autorisatie mogelijkheden zijn basic, integrated windows, Anonymous en SOAP header. Zoals ik heb opgenomen in de architectuur van het systeem zal autorisatie mogelijk zijn op SOAP header niveau. Dit vereist dat in een binnenkomend XML bericht een username en password moet worden meegegeven. Pluriform zal dan aan de hand van de klasse Gebruikersaccount kijken of de opgegeven user bestaat. Dit vereist van klanten van Matthat Software B.V die het systeem willen implementeren dat van alle bezoekers die via hun website inloggen een account bestaat in de klasse Gebruikersaccount. Als men dit niet wil zal de autorisatie op Anonymous worden gezet.

Nu het systeem een XML service heeft die bereikbaar is over het internet kan ik een nieuwe Service Operation ontwikkelen die een aantal onderdelen verwacht:

- Operation ID. Dit wordt 'getPerson'.
- Een XML Service. Dit wordt de zojuist aangemaakte service 'SOAPXMLService'.
- Type. Het type kan inkomend of uitgaand zijn. Ik heb gekozen voor Request-Response operations, omdat er altijd eerst een aanvraag vanaf een website wordt gedaan naar de XML service.
- Input message. Hier moet het binnenkomende XML bericht worden gedefinieerd. Wat hier wordt aangegeven bepaald dus hoe een aanvraag eruit komt te zien.
- Output message. Hier wordt het antwoord XML bericht gedefinieerd.
- Handle request script. Een Pluriform script waarin de input message wordt afgehandeld en waar het resultaat de output message is.

Voor de input message is het de bedoeling dat de aanvrager een relatienummer als parameter ingeeft. Uit het ontwerp blijkt dat relatienummer in de klasse Relatie zit en een relatie heeft met de klasse Persoon. Daarom kan er straks in het handle request script gezocht worden naar een persoon. Anders als het opzetten van een Service Message (zie vorige paragraaf) is dat in de input geen struct worden gezet maar een XML Node. Een XML Node is een element in een XML tekst die omsloten wordt door de tags < en >. Dit betekent dus dat ik nog letterlijker een XML bericht aan het ontwikkelen ben omdat de naamgeving eigenlijk al zegt wat je ontwikkelt. Het XML bericht hieronder toont de naamgeving van de verschillende elementen in een XML bericht:

```
<ditiseenXMLNode>
   <ditiseenSubXMLNode ditiseenXMLattribuut=""></ditiseenSubXMLNode>
</ditiseenXMLNode>
```

De input message zal dus bestaan uit het volgende:

XML Node requestWithRelationNumber
        XML attribute relationnumber

En zal er zo uit moeten zien:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:plur="pluriform:SOAPXMLService">
  <soap:Header/>
  <soap:Body>
    <plur:requestWithRelationnumber relationnumber=""/>
  </soap:Body>
</soap:Envelope>
```

De output message wordt als volgt:

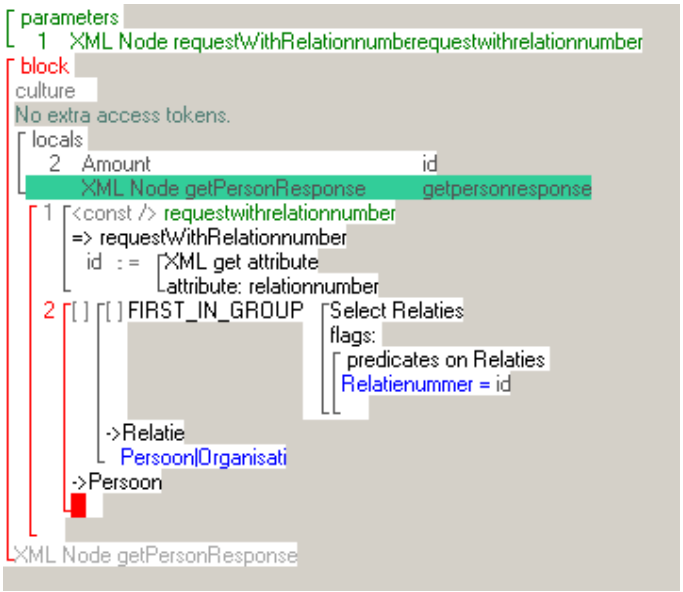
XML Node getPersonResponse
Sub XML Node person

XML attribute Initials	string
XML attribute Nickname	string
XML attribute Surname_own	string
XML attribute Address_as	string
XML attribute Sex	string
XML attribute Date_of_birth	date
XML attribute Civil_status	string
XML attribute Private_e-mail_address	string
XML attribute Nationality	string

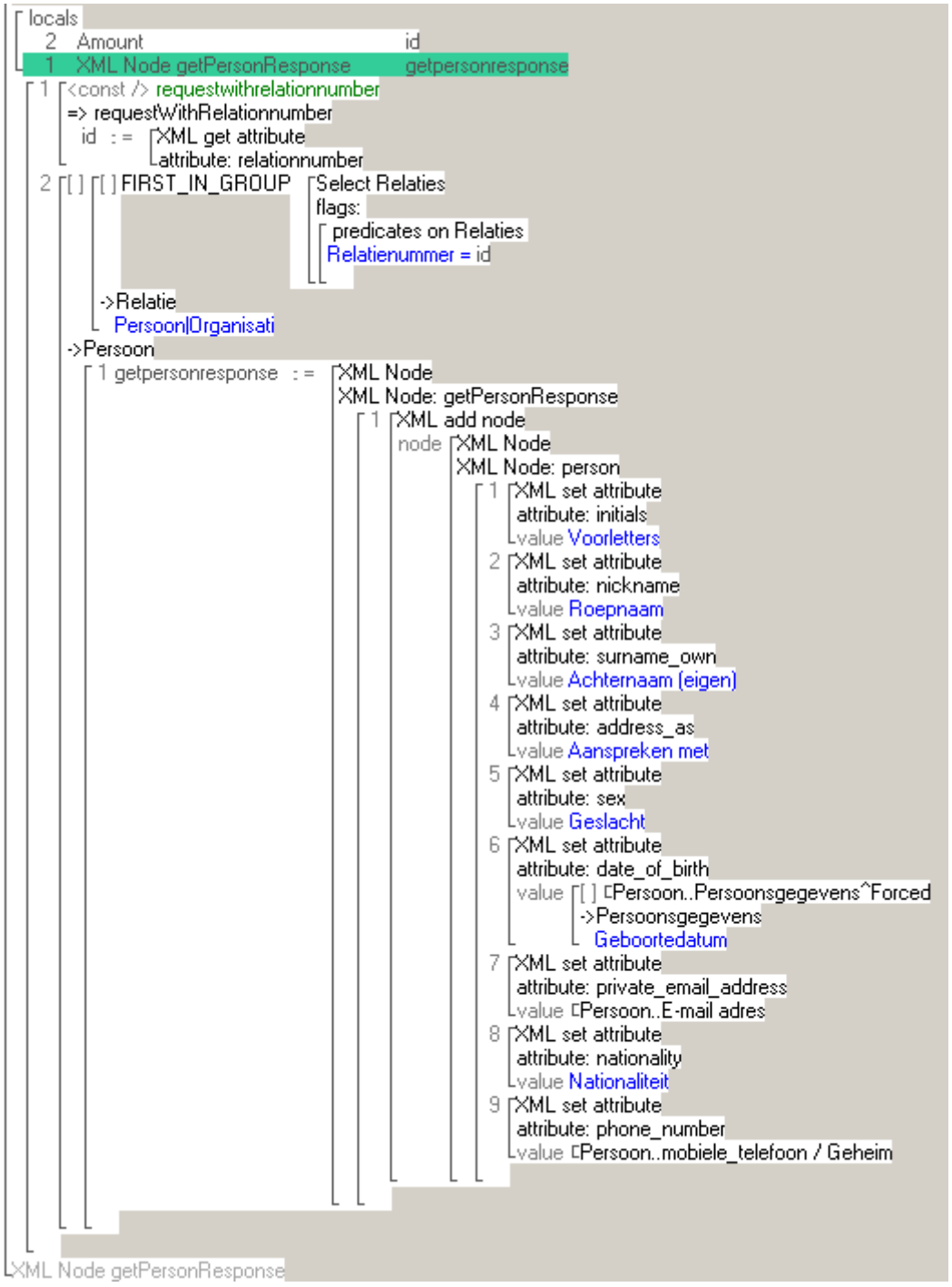
En zal er zo uit moeten zien:

```
<Envelope xmlns="http://www.w3.org/2003/05/soap-envelope">
  <Body>
    <getPersonResponse xmlns="pluriform:SOAPXMLService">
      <person initials="" nickname="" surname_own="" address_as="" sex="" date_of_birth="" private_email_address="" nationality=""
            phone_number=""/>
    </getPersonResponse>
  </Body>
</Envelope>
```

Vervolgens zal ik in het Handle response script ervoor moeten zorgen dat input message op een juiste manier wordt afgehandeld en de juiste output message met de juiste gegevens retourneert. Allereerst maak ik een lokale variabele id van het type amount. Deze variabele zal gevuld moeten worden met het opgegeven relatienummer uit het binnenkomende XML bericht. In de tweede stap zoek ik de juiste persoon bij het relatienummer die nu in de variabele id staat.



Op dit punt in het script zit ik in de context van een persoonsobject die gerelateerd is aan het relatienummer. Vervolgens kan ik alle attributen, oftewel persoonsgegevens, van deze persoon ophalen en in de output message plaatsen:



7.3 UITVOEREN TRANSITION FASE: IMPLEMENTATION AND TEST

Het POC moet vervolgens getest worden. Om deze te kunnen testen moet ik de mogelijkheid hebben om naar het systeem een aantal XML berichten te sturen. Hierdoor wordt er getest op binnenkomende XML berichten (hierna genoemd request) en uitgaande XML berichten (hierna genoemd response).

7.3.1 DE TEST TOOLS

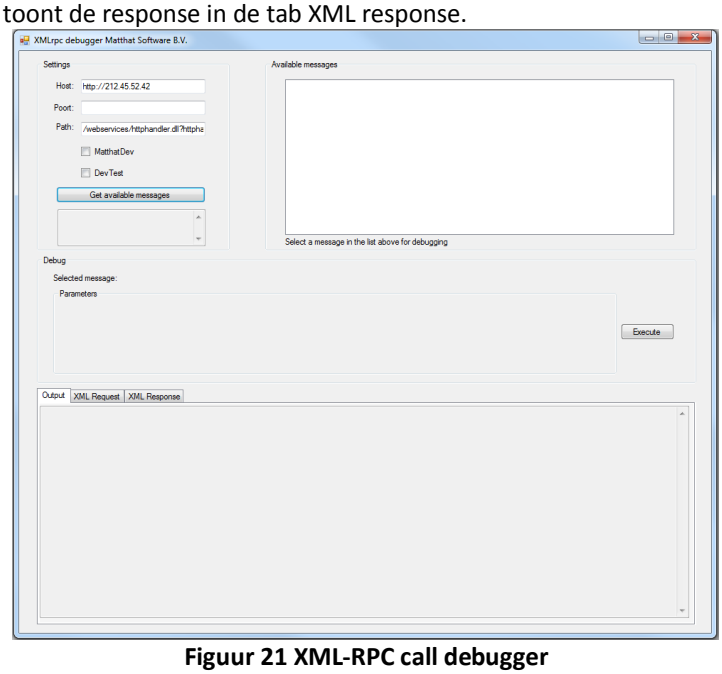
Bij het testen van het POC moet ik een client kunnen simuleren die XML berichten verstuurd naar het systeem. Bij het testen van het POC had ik niet genoeg kennis van het script om gebruik te kunnen maken van de ingebouwde testfunctionaliteit in Pluriform. Deze testfunctionaliteit biedt mogelijkheden om een client te simuleren die een XML bericht verstuurd naar het systeem. Om toch te kunnen testen moest ik op zoek naar externe tools die zowel XML-RPC- als SOAP berichten kan versturen.

XML-RPC.NET library

Ik ben gaan zoeken op internet voor mogelijke tools van derden om XML-RPC berichten te simuleren maar die zijn niet aanwezig. Na wat dieper onderzoek op internet bleek er een open source library te bestaan genaamd XML-RPC.NET (XML) die het mogelijk maakt om een XML-RPC server en client te implementeren in een .NET omgeving.

Door deze library te importeren in een nieuw Windows project in Visual Studio 2010 kon ik gebruik maken van verschillende klassen die het mogelijk maken om een client op te zetten. Figuur 21 toont een overzicht van de applicatie. De applicatie werkt als volgt:

1. De applicatie maakt verbinding met het systeem en stuurt een XML bericht ‘get available messages’.
2. Het systeem stuurt als reactie een XML bericht terug naar de applicatie met alle service messages uit het systeem.
3. De applicatie doorloopt het XML bericht en toont deze in de lijst.
4. De gebruiker selecteert een service message
5. De applicatie maakt een XML bericht aan en toont deze in de tab XML request
6. De applicatie maakt voor alle verwachte parameters in het te sturen XML bericht een textbox aan.
7. De gebruiker vult de parameters met een waarde en drukt op execute.
8. Het systeem stuurt het XML bericht met de parameters naar het systeem.
9. Het systeem stuurt een XML bericht terug naar de applicatie.
10. De applicatie toont de response in de tab XML response.



Figuur 21 XML-RPC call debugger

Hierdoor wordt het mogelijk om de service messages in het systeem te testen op request en response.

soapUI
Voor het testen van SOAP XML berichten heb ik gekozen om de gratis versie van soapUI te gebruiken (soapUI). Dit is een grafische Windows applicatie die aan de hand van een WSDL voorbeeld XML berichten creert. Bij het uitvoeren van een gegenereerd XML bericht door soapUI wordt de response ook als XML weergegeven.

7.3.2 TESTEN PERFORMANCE WEB-CONNECTOR

Het POC levert nu twee manieren voor het ophalen van persoonsgegevens van een specifiek persoon uit Pluriform. Ik vind het belangrijk om op dit punt een performance test te doen van de web connector. Hierbij maak ik gebruik van de testtool loadUI. De web connector is namelijk een belangrijk component in het systeem die niet overbelast mag raken door bijvoorbeeld teveel binnenkomende XML berichten. Daarom test ik de performance van de connector en wil ik erachter komen hoe lang de connector erover doet om een XML bericht af te handelen en hoeveel binnenkomende XML berichten hij per seconde aankan zonder dat de queue volloopt:

- TimeTaken: De tijd die de web connector nodig heeft om een binnenkomend XML bericht af te handelen. Het afhandelen betekent in dit geval een antwoord bericht terug sturen naar de client.
- Rate: Het hoogst aantal gelijktijdige aanvragen per seconde naar de server. Dit is vergelijkbaar met het aantal bezoekers van een website die op een bepaald moment dezelfde actie uitvoeren op het systeem.

Bovenstaande testsituaties ga ik uitvoeren met behulp van het ontwikkelde POC. Het eerste testgeval zal XML berichten versturen via het RPC protocol en het tweede testgeval stuur XML berichten met het SOAP protocol. Elk testgeval zal persoonsgegevens voor één specifiek persoon ophalen uit Pluriform. Als in de uitkomsten belangrijke verschillen te ontdekken zijn, zal dit meewegen in mijn keuze tussen de twee protocollen.

Test #1: getPerson met gebruik van XML-RPC

- Rate: 16
- TimeTaken: 200,42 miliseconden (het gemiddelde over 1000 XML berichten)

Test #2: getPerson met gebruik van SOAP

- Rate: 8
- TimeTaken: 620,71 miliseconden (het gemiddelde over 1000 XML berichten)

7.3.3 TESTEN PROOF OF CONCEPT MET XML-RPC

Met gebruik van de debugger roep ik de service message getPerson aan met het onderstaande XML bericht. Als parameter geef ik mijn eigen relatienummer op. Ik verwacht nu dat het systeem mijn persoonsgegevens terug stuurt.

```
<methodCall>
 <methodName>getPerson</methodName>
 <params>
  <param>
   <value><int>12483</int></value>
  </param>
 </params>
</methodCall>
```

De response van het systeem is het onderstaande XML bericht. Hierin is te zien dat alle velden die ik verwacht aanwezig zijn en ook een waarde bevatten!

```
<methodResponse>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>Voorletters</name>
            <value>
              <string>D.</string>
            </value>
          </member>
          <member>
            <name>Achternaam</name>
            <value>
              <string>Roos</string>
            </value>
          </member>
          <member>
            <name>Voornaam</name>
            <value>
              <string>David</string>
            </value>
          </member>
          <member>
            <name>Aanspreektitel</name>
            <value>
              <string>achternaam eigen</string>
            </value>
          </member>
          <member>
            <name>Geslacht</name>
            <value>
              <string>man</string>
            </value>
          </member>
          <member>
            <name>Geboortedatum</name>
            <value>
              <string>01-05-1989</string>
            </value>
          </member>
          <member>
            <name>Telefoonnummer</name>
            <value>
              <string>06 - 54 37 69 40</string>
            </value>
          </member>
```

```
          <member>
            <name>Emailadres</name>
            <value>
              <string>roos.david@gmail.com</string>
            </value>
          </member>
          <member>
            <name>Nationaliteit</name>
            <value>
              <string>Nederlandse</string>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodResponse>
```

Hieruit kan ik concluderen dat de juiste gegevens zijn opgehaald door het systeem en dat de service message naar behoren werkt.

7.3.4 TESTEN PROOF OF CONCEPT MET SOAP

Ook met gebruik van soapUI kan ik de service operation getPerson aanroepen met het onderstaande XML bericht. Het XML bericht heeft als parameter mijn relatienummer. Ook hier verwacht ik dat het systeem mijn persoonsgegevens terug stuurt.

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:plur="pluriform:SOAPXMLService">
  <soap:Header/>
  <soap:Body>
    <plur:requestWithRelationnumber relationnumber="12483"/>
  </soap:Body>
</soap:Envelope>
```

De response van het systeem is het onderstaande XML bericht. Hierin is te zien dat alle velden die ik verwacht aanwezig zijn en ook een waarde bevatten!

```
<Envelope xmlns="http://www.w3.org/2003/05/soap-envelope">
  <Body>
    <getPersonResponse xmlns="pluriform:SOAPXMLService">
      <person initials="D." nickname="David" surname_own="Roos" address_as="achternaam eigen" sex="man" date_of_birth="1989-05-
            01" private_email_address="roos.david@gmail.com" nationality="Nederlandse" phone_number="06 - 54 37 69 40"/>
    </getPersonResponse>
  </Body>
</Envelope>
```

Hieruit kan ik concluderen dat de juiste gegevens zijn opgehaald door het systeem en dat de service operation naar behoren werkt.

7.4 CONCLUSIE

7.4.1 DE KEUZE VAN HET PROTOCOL

	RPC	SOAP
1. Web Service Description Language (hierna genoemd WSDL)	no	Yes
2. HTTP en HTTPS	no	Yes
3. XML	Yes	Yes
4. Enveloppe/header/body	no	Yes
5. Developer gedefinieerde datatypes	no	Yes
6. Named Data structures	no	Yes
7. Vereist kennis van de client	no	no
8. Bericht specifieke verwerkingsinstructies	no	Yes
9. Gedetailleerde foutafhandeling	Yes	Yes
10. Opgeven van de ontvanger	no	Yes

Tabel 4 XML-RPC versus SOAP

Nu het Proof of Concept is ontwikkeld kan ik een keuze maken tussen de twee communicatieprotocollen XML-RPC en SOAP. In tabel 4 heb ik de belangrijkste eigenschappen met elkaar vergeleken. Hierdoor concludeer ik dat SOAP meer vrijheid biedt aan de ontwikkelaar. Hierbij doel ik vooral op het geven van eigen namen aan velden, ondersteuning van HTTPS en zelf de datatypes kunnen definiëren. Toch kwam uit de test dat XML-RPC gemiddeld per 'call' drie keer sneller wordt afgehandeld dan een SOAP 'call'. Dit weegt niet op tegen voordelen die opgenomen zijn in tabel 4 en is elke TimeTaken onder de 1 seconde acceptabel voor Matthat Software B.V. en de klant.

Na de eerste iteratie heb ik samen met een collega een demo gehad van Pluriform Software over het opzetten van SOAP services in Pluriform. Ik heb gevraagd welke techniek hun voorkeur heeft. Het antwoord was dat SOAP beter te gebruiken is voor website ontwikkelaars vanwege de uitgebreide WSDL. Hierbij zei Pluriform Software dat er steeds meer behoefte en vraag kwam van hun klanten naar SOAP services. Daartoe heeft men besloten om in januari 2012 XML-RPC outdated te verklaren, wat betekent dat het in de kopgroep versie (zie paragraaf 1.5) niet meer beschikbaar is in het systeem en dat een jaar later er ook geen support meer voor is. Mede hierdoor is de conclusie dat het systeem verder ontwikkelt wordt met gebruik van het SOAP protocol. Hierbij zal ik gebruik maken van de eigenschappen uit tabel 4.

Hieronder heb ik een lijstje met voordelen van SOAP opgenomen wat benadrukt dat SOAP een goed communicatiemiddel is voor het doel van dit project (zie paragraaf 2.4**Fout! Verwijzingsbron niet gevonden.**).

- Communicatie via internet
- Platformonafhankelijk
- Taalonafhankelijk
- Eenvoudig en uitbreidbaar
- Wordt toegelaten door firewalls
- Aanbevolen door W3C (W3C)

7.4.2 EVALUATIE

Dat XML-RPC outdated werd verklaart door Pluriform Software was enigszins een meevaller voor mij omdat de keuze tussen beide protocollen nu sowieso zou uitdraaien op het inzetten van het SOAP protocol, maar als ik dit voor het begin van de eerste iteratie wist zou ik de ontwikkeltijd voor het POC met het XML-RPC protocol niet als 'verloren' hoeven beschouwen. Voor Matthat Software B.V. is het minder goed nieuws aangezien klanten gebruik maken van XML-RPC en dus op termijn gedwongen worden over te stappen naar SOAP.

In de eerste iteratie heb ik een aantal onderdelen geleerd:

1. De werking van het XML-RPC en SOAP protocol
2. De opzet en het slim gebruik maken van de klassen in Pluriform
3. Gebruik maken van een WSDL
4. Het inrichten van web services in IIS
5. Gebruik maken van script elementen uit Pluriform

Voor het ontwikkelen van de XML-RPC debugger heb ik veel opgedane kennis kunnen gebruiken uit het I6 blok 'Software architectuur' van de Haagse Hogeschool. Denk hierbij specifiek aan de .Net lessen.

7.4.3 VAN ITERATIE 1 NAAR ITERATIE 2

Nu de basis voor het systeem gelegd is met de service operation GetPerson kan ik verder gaan met ontwikkelen in een tweede iteratie. Hierbij kan ik door de opgedane kennis uit de eerste iteratie een uitgebreid functioneel en technisch ontwerp maken voor het systeem en de overige, nog niet behaalde, requirements onderbrengen in het systeem.

Het set van requirements wordt door de eerste iteratie aangevuld met onderstaand requirement. Dit requirement is zowel een functioneel als niet functioneel requirement. Het is een niet-functionele eis vanwege de kwaliteitsattributen 'nakomen van betrouwbaarheidsnormen en bruikbaarheidsnormen' waaraan het SOAP protocol moet voldoen volgens W3C (W3C). Hiernaast wordt deze eis volgens de MoSCoW methode geprioriteerd als een must have eis vanwege het feit dat zonder deze eis het systeem niet kan functioneren.

- Het systeem wordt ontwikkeld met gebruik van het SOAP protocol.

8 ITERATIE 2# ONTWIKKELEN VAN EEN WIJZIGINGSINTERFACE

Op het moment dat het de te ontwikkelen XML berichten gebruikt worden om gegevens in Pluriform te wijzigen of toe te voegen kom ik op een lastig vraagstuk. Want welke wijzigingen en toevoegingen mogen zonder controle en accordatie gelijk worden doorgevoerd in de database van Pluriform? Ik ben tot de conclusie gekomen dat dit per klant kan verschillen. Maar als dit per klant anders is hoe zorg ik er dan voor dat ik geen maatwerk moet ontwikkelen om dit vraagstuk op te lossen?
De oplossing voor dit vraagstuk is het ontwikkelen van een wijzigingsinterface die het mogelijk maakt om in te stellen welke gegevens (lees attributen uit een specifieke klasse in Pluriform) direct te muteren en toe te voegen zijn aan de database. Alle andere gegevens zullen eerst gecontroleerd en geaccordeerd moeten worden door een gebruiker van het systeem voordat deze gemuteerd of toegevoegd mogen worden aan de database van Pluriform. Daarom zal ik in deze tweede iteratie een wijzigingsinterface ontwerpen en ontwikkelen.

Het hoofdstuk is opgedeeld in drie fases van Unified Process, namelijk elaboration fase (8.1), construction fase (8.2) en transition fase (8.3).

8.1 UITVOEREN ELABORATION FASE: ANALYSIS AND DESIGN

8.1.1 HET FUNCTIONELE ONTWERP VOOR DE WIJZIGINGSINTERFACE

Het is de bedoeling dat de web service een deel van de functionaliteit in Pluriform geautomatiseerd kan laten verlopen. Denk hierbij bijvoorbeeld aan het wijzigen van adresgegevens of een toezegging of gift aanmaken. Binnen het systeem ga ik het mogelijk maken om wijzigingsverzoeken vanaf de website eerst klaar te zetten in Pluriform als wijzigingsvoorstel. Een wijzigingsvoorstel kan een wijziging bevatten in elke in Pluriform beschikbare klasse en attribuut.

Figuur 22 Domeinmodel interface wijzigingen of toevoegingen

Binnen Pluriform ga ik hiervoor twee nieuwe klassen ontwikkelen, namelijk *wijzigingsvoorstel* en *wijzigingsinstelling*. Deze twee klassen zijn uitsluitend bedoeld voor verzoeken om gegevens te wijzigen in Pluriform vanaf een website en via de te ontwikkelen service operations. Met de klasse *wijzigingsinstelling* wordt het mogelijk om aan te geven welke attributen direct van buitenaf gewijzigd mogen worden in de database. Alle wijzigingsverzoeken van attributen die dus niet zijn opgenomen in deze klasse zullen als wijzigingsvoorstel aangemaakt worden in de klasse *wijzigingsvoorstel*. Door het kiezen van twee nieuwe klassen kan het probleem van wijzigingen wel of niet gelijk doorvoeren in de database worden opgelost. De twee klassen hebben geen directie relatie met elkaar maar de klasse *wijzigingsinstelling* zal benaderd worden in een te ontwikkelen generic om tot een *wijzigingsvoorstel* te komen (het ontwerp van de generic is te vinden in paragraaf 8.2).

Bij de twee klassen zal sprake zijn van low coupling omdat ze niet afhankelijk zijn van andere klassen. Het voordeel hiervan is dat wijzigingen in de klassen geen gevolgen heeft op andere klassen, de klassen eenvoudig te begrijpen en goed herbruikbaar zijn.

De eerste klasse wijzigingsvoorstel bevat een aantal attributen:

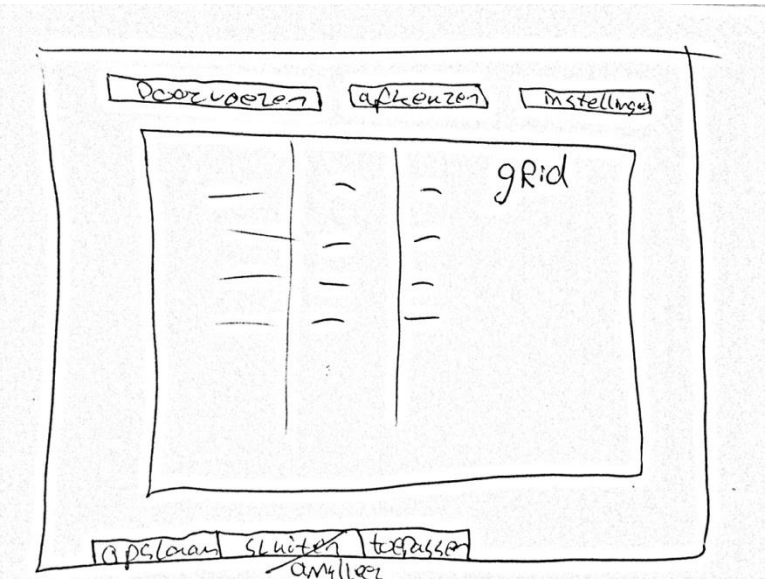
- Context. Dit wordt een verwijzing naar de context waar de wijziging of toevoeging betrekking op heeft.
- Attribuut. Dit wordt het te wijzigen attribuut.
- Waarde. Dit wordt de waarde van het attribuut zoals opgegeven door degene die zijn gegevens wijzigt.
- Status. De statussen die ik gekozen heb voor een wijzigingsvoorstel zijn:
  - **Doorvoeren.** Het voorstel is nieuw en moet nog behandeld worden.
  - **Afgewezen.** Het voorstel is afgewezen door een gebruiker om een bepaalde reden.
  - **Geannuleerd.** Het voorstel is geannuleerd door Pluriform. Dit gebeurd als er twee overeenkomende wijzigingsvoorstellen zijn met status doorvoeren. Het eerst binnen gekomen voorstel zal geannuleerd worden.
  - **Doorgevoerd.** Het voorstel is doorgevoerd in Pluriform en afgehandeld door de gebruiker.

Ik heb hiernaast gekozen voor het bewaren van alle wijzigingsvoorstellen ongeacht hun status, omdat hierdoor Pluriform een 'historie' bijhoudt wat handig kan zijn bij het verwerken van door te voeren wijzigingsvoorstellen. Ik stond hierbij voor een dilemma aangezien het bewaren van data kan leiden tot vervuiling van de database. Ik heb daarvoor navraag gedaan bij collega's of hiervoor regels zijn opgesteld. Het bleek dat Matthat Software B.V. de regel hanteert dat in principe data nooit wordt weggegooid. Daarom heb ik besloten om deze regel ook te hanteren voor het XML framework, maar hou ik wel in gedachte dat klanten hier anders over denken. Voor klanten die willen dat wijzigingsvoorstellen verwijderd worden na het verwerken zal later deze mogelijkheid ingebouwd kunnen worden. Op het moment dat data gewijzigd wordt in nieuwe waarden zal de oude data niet behouden blijven (niet persistent).

De tweede klasse wijzigingsinstelling bevat ook een aantal attributen:

- Klasse. Een verwijzing naar een klasse in Pluriform.
- Attributen direct muteerbaar. Een groep van attributen uit de opgegeven klasse.

Voor het behandelen van een wijzigingsvoorstel ga ik een zogeheten control center in Pluriform ontwikkelen. Hier kan op geklikt worden, waardoor er een actie scherm wordt geopend. Let op, dit zal alleen mogelijk zijn als de gebruiker in de rol Gebruiker bedrijfsprocessen zit. Ik heb hiervoor gekozen omdat het een eis was vanuit het opzetten van een GUI. Het behandelen van voorstellen moet namelijk mogelijk zijn in Pluriform en vanuit het hoofdscherm. De keuze voor een control center (knop op het hoofdscherm) in plaats van een taak heeft meerdere argumenten. Een control center heeft een soort 'cockpit-functie', dit houdt in dat de gebruiker met één klik op de knop het actiescherm kan openen. Bij een taak is dit anders, de gebruiker moet eerst zijn gebruikersgroep kiezen en vervolgens in het menu taken de taak selecteren. Als je bedenkt dat het verwerken van wijzigingsvoorstellen een dagelijkse taak kan worden binnen een organisatie is de keuze voor een control center terecht. Het tweede argument die ik vond is dat bij het starten van Pluriform de gebruiker de mogelijkheid heeft om het actiescherm voor het



Figuur 23 Schets actiescherm wijzigingsvoorstellen

verwerken van wijzigingsvoorstellen automatisch op te laten starten. Als de gebruiker als enige rol heeft om wijzigingsvoorstellen te verwerken zal dus een control center een logische keuze zijn.

Voor dit control center moet ik ook een Graphical User Interface (GUI, hierna genoemd actiescherm, figuur 23) ontwerpen. Ik heb daarom besloten om samen met een collega van Matthat Software B.V. te overleggen aan welke requirements dit actiescherm moet voldoen. Deze requirements zal ik allemaal verwerken in het actiescherm.

Requirements actiescherm

- Het actiescherm toont bij openen door te voeren wijzigingsvoorstellen, zichtbaar aan de status.
- De gebruiker heeft de mogelijkheid om meerdere voorstellen te selecteren.
- De gebruiker heeft de mogelijkheid om meerdere wijzigingsvoorstellen ineens door te voeren met een knop 'Wijziging doorvoeren' of af te keuren met een knop 'Wijziging afkeuren'.
- De gebruiker heeft de mogelijkheid om voor elk wijzigingsvoorstel de status en de nieuwe waarde te wijzigen.
- De gebruiker heeft de mogelijkheid om opmerkingen toe te voegen aan een wijzigingsvoorstel.
- Afhankelijk van het geselecteerde wijzigingsvoorstel kan de gebruiker de instellingen bekijken van de context van het wijzigingsvoorstel met een knop 'Bekijk instellingen'.
- Elk wijzigingsvoorstel kan een knop bevatten die verwijst naar wijzigingen in dezelfde context, met hetzelfde attribuut en die de status doorgevoerd, geannuleerd of afgekeurd hebben. Het doel hiervan is de historie te kunnen bekijken.
- Het scherm bevat een knop Opslaan. De gebruiker kan hierop klikken als er wijzigingen gedaan zijn in het actiescherm en op wijzigingsvoorstellen. Als er op gedrukt wordt worden de wijzigingen doorgevoerd en het scherm gesloten.
- Het scherm bevat een knop Toepassen. De gebruiker kan hierop klikken als er wijzigingen gedaan zijn in het actiescherm en op wijzigingsvoorstellen. Als er op gedrukt wordt worden de wijzigingen doorgevoerd.
- Het scherm bevat een knop Sluiten/Annuleer. Standaard heet de knop Sluiten en kan het scherm worden afgesloten door erop te klikken. Als er wijzigingen in het actiescherm en op wijzigingsvoorstellen worden gedaan moet de knop veranderen in Annuleer.
- Het actiescherm is te benaderen met een knop vanaf het hoofdscherm in Pluriform goede doelen.
- Alleen als de gebruiker in de rol Gebruiker bedrijfsprocessen zit zal deze knop zichtbaar zijn.
- Voor elk wijzigingsvoorstel waarvan de context betrekking heeft op een andere klasse wordt een nieuw tabblad gemaakt in het actiescherm.
-

De use-cases *maak een wijzigingsvoorstel aan, dit wijzigingsvoorstel behandelen* en *deze wijzigingsinstellingen doorvoeren* horen allemaal bij de wijzigingsinterface. Elke use-case beschrijving beschrijft de wijze waarop het systeem dient te functioneren.

Naam	Maak een wijzigingsvoorstel aan
Samenvatting	Aan de hand van een context, te wijzigen attribuut en de nieuwe waarde kan er in Pluriform een wijzigingsvoorstel worden aangemaakt
Aannamen	- Klasse wijzigingsvoorstel en wijzigingsinstelling bestaan in het Pluriform systeem - De context, te wijzigen attribuut en de waarde van de wijziging zijn bekend
Beschrijving	(1) Het systeem controleert of de combinatie van context, attribuut en waarde al bestaat in de wijzigingsvoorstellen die de status doorvoeren hebben in de klasse wijzigingsvoorstel, als dit het geval is treedt uitzondering [wijziging bestaat al] op. (2) Het systeem maakt een nieuw wijzigingsvoorstel aan met de opgegeven context, attribuut en waarde en zet de status van het voorstel op doorvoeren.
Uitzonderingen	[Wijziging bestaat al] Het systeem negeert de aanvraag om een wijziging door te voeren
Resultaat	Er is een nieuw wijzigingsvoorstel aangemaakt in Pluriform door het systeem

Naam	Dit wijzigingsvoorstel behandelen
Samenvatting	In Pluriform kan elk wijzigingsvoorstel worden behandeld. Dit houdt in status wijzigen,

	waarde wijzigen, opmerking toevoegen of doorvoeren.
Actor	Gebruiker van het systeem
Aannamen	- Klasse wijzigingsvoorstel en wijzigingsinstelling bestaan in het Pluriform systeem
Beschrijving	(1) Actor opent het scherm 'Verwerk wijzigingsvoorstellen' door op het zogeheten control center hiervoor te drukken (2) Het systeem haalt alle wijzigingsvoorstellen op met de status doorvoeren, zet een lock op de data en presenteert de wijzigingsvoorstellen in het scherm (3) Actor behandeld wijzigingsvoorstel, als hij de status wijzigt treedt uitzondering [Status gewijzigd] op. Als de actor de waarde wijzigt treedt uitzondering [Waarde gewijzigd] op. (4) Actor selecteert het wijzigingsvoorstel en drukt op de knop wijziging doorvoeren. (5) Het systeem controleert of de status van het voorstel op doorvoeren staat. Als de status van het voorstel niet op doorvoeren staat treedt uitzondering [Status niet op doorvoeren] op. (6) Het systeem start een transactie en voert de wijzigingen door in het systeem. (7) Het systeem wijzigt de status van het wijzigingsvoorstel naar doorgevoerd. (8) Het systeem haalt de lock van de data af. (9) Actor sluit het scherm door op de knop 'Sluiten' te klikken.
Uitzonderingen	[Status gewijzigd] Het systeem veranderd de status van het wijzigingsvoorstel, het wijzigingsvoorstel kan niet meer worden doorgevoerd als de status niet doorvoeren is. De actor moet dit bevestigen door op de knop 'Toepassen' of 'Opslaan' te klikken. [Waarde gewijzigd] Het systeem start een transactie en veranderd de waarde van het wijzigingsvoorstel in de nieuw ingegeven waarde. Als de combinatie context, attribuut en waarde vervolgens als bestaat in één van de wijzigingsvoorstellen met status doorvoeren zet het systeem de status van het wijzigingsvoorstel op geannuleerd. Het wijzigingsvoorstel kan niet meer worden doorgevoerd als de status niet doorvoeren is. De actor moet dit bevestigen door op de knop 'Toepassen' of 'Opslaan' te klikken. [Status niet op doorvoeren] Het systeem disabled de knop 'wijziging doorvoeren', dus kan de actor de taak niet uitvoeren.
Resultaat	Er is een wijzigingsvoorstel behandeld en doorgevoerd.

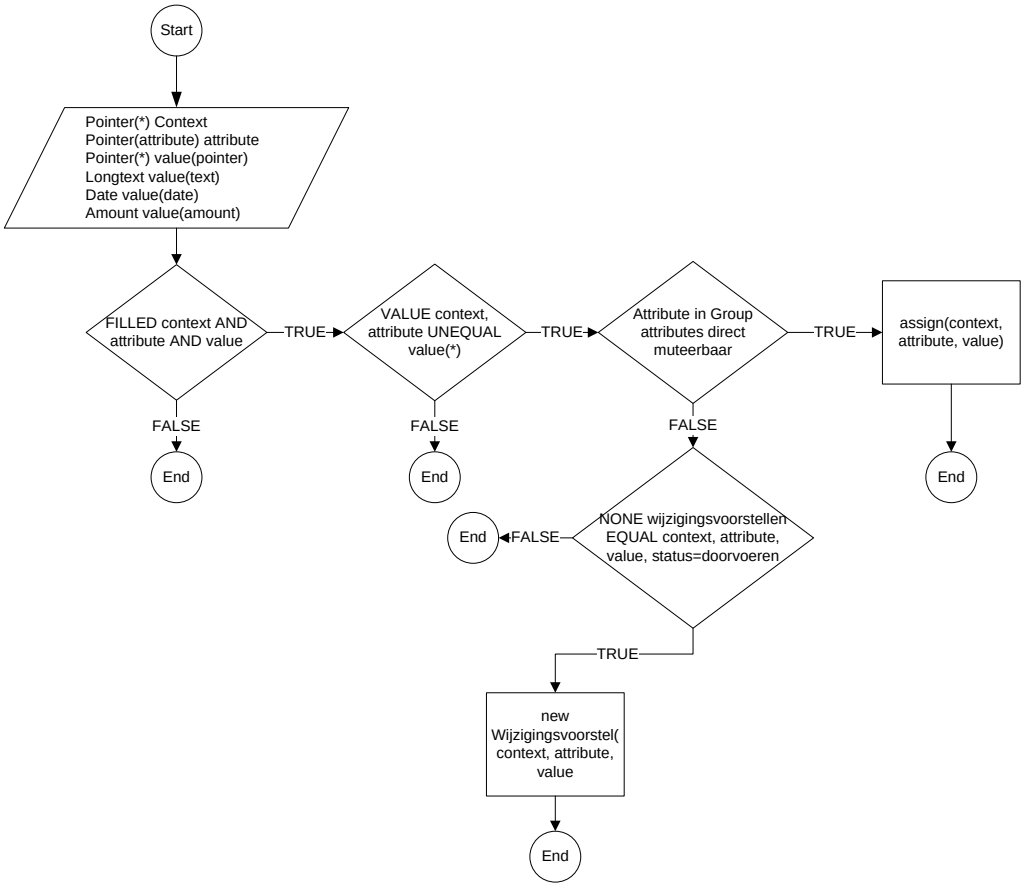
Naam	Deze wijzigingsinstellingen doorvoeren
Samenvatting	In Pluriform kan voor elk attribuut in elke klasse ingesteld worden of deze direct gemuteerd mag worden.
Actor	Gebruiker van het systeem
Aannamen	- Klasse wijzigingsvoorstel en wijzigingsinstelling bestaan in het Pluriform systeem
Beschrijving	(1) Actor opent het scherm 'Wijzigingsinstellingen' door op achtereenvolgens op F8 te drukken, 'wijzigingsinstellingen' te typen en ENTER te drukken. (2) Het systeem start een transactie, haalt alle wijzigingsinstellingen op, zet een lock op de data en presenteert de wijzigingsinstellingen in het scherm. (3) Actor drukt op de knop 'Nieuw', selecteert achtereenvolgens de klasse waarop de instellingen van toepassing zijn en voegt alle attributen hieraan toe die direct gemuteerd mogen worden. (4) Het systeem voegt alle attributen toe aan een groep 'attributes direct muteerbaar' (5) Het systeem eindigt de transactie en geeft daarmee de lock vrij op de data (6) Actor drukt achtereenvolgens op 'Opslaan' en 'Sluiten'.
Uitzonderingen	[]
Resultaat	Er is een nieuwe wijzigingsinstelling gemaakt waarin is aangegeven in welke klasse welke attributen direct gemuteerd mogen worden.

8.2 UITVOEREN CONSTRUCTION FASE: DESIGN AND IMPLEMENTATION

8.2.1 HET TECHNISCHE ONTWERP VAN DE WIJZIGINGSINTERFACE

Ik wil functionaliteit ontwikkelen die het mogelijk maakt om een controle te doen of een attribuut direct gewijzigd mag worden of dat er een wijzigingsvoorstel aangemaakt moet worden. Deze functionaliteit zal op vele plekken in het systeem gebruikt gaan worden. Om te voorkomen dat dezelfde bewerkingen in het systeem steeds opnieuw worden geprogrammeerd heb ik besloten om een generic (lees: methode) te ontwikkelen die deze functionaliteit afvangt. Op dat moment is er sprake van decompositie. Het voordeel van een generic is dat er geen sprake is van encapsulation (inkapseling) van de functionaliteit in een bepaalde klasse, maar dat de generic beschikbaar is door het hele systeem. Om een beeld te krijgen van de werking van deze generic heb ik stappenplan opgesteld (zie ook figuur 24):

1. De generic wordt ingezet bij het toewijzen van een waarde aan een attribuut.
2. De generic wordt gevuld met een context, attribuut en de nieuwe waarde.
3. In de generic wordt gekeken of de waarde van het opgegeven attribuut in een context overeen komt met de nieuwe waarde. Zo niet, dan betreft het inderdaad een wijziging.
4. Vervolgens wordt er gecontroleerd of het opgegeven attribuut is opgenomen in de klasse wijzigingsinstelling. Als dit het geval is wordt de wijziging direct doorgevoerd anders moet er een wijzigingsvoorstel worden aangemaakt.
5. Dan controleert de generic of er al een overeenkomend wijzigingsvoorstel is waarvan de status op doorvoeren staat. Als dit het geval is negeert de generic het te maken wijzigingsvoorstel.
6. De generic maakt een nieuw object wijzigingsvoorstel met de opgegeven context, attribuut en de nieuwe waarde.

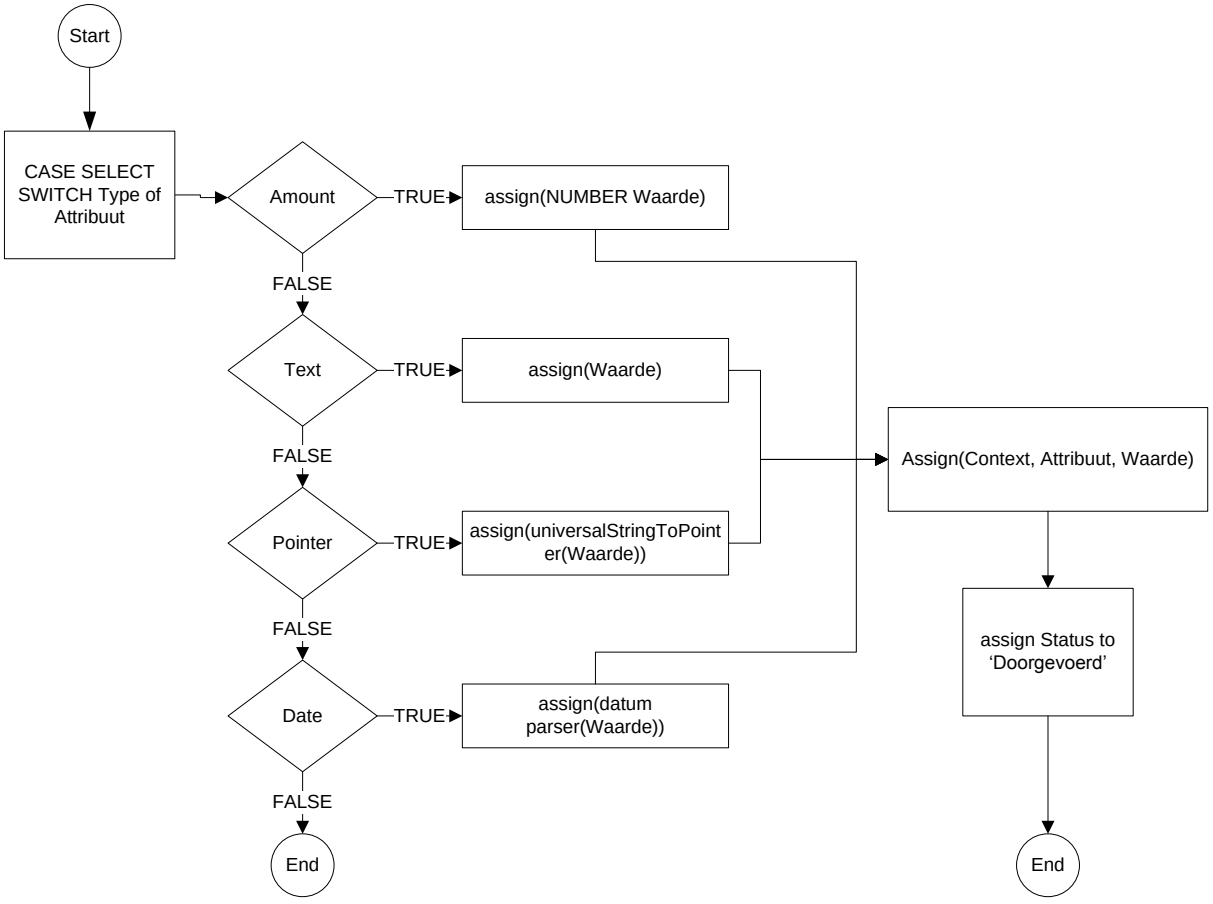


Figuur 24 Werking generic update or change request

Hiernaast heb ik besloten om een zogeheten taak te ontwikkelen in het systeem die in een specifieke rol uit Pluriform opgenomen kan worden. Deze taak *wijziging doorvoeren* moet onder een knopje komen in de user interface van Pluriform. Hierdoor wordt het mogelijk om een gebruiker van de interface door een druk op het knopje één of meerdere wijzigingen (afhankelijk van het aantal geselecteerden wijzigingsvoorstellen) door te voeren. Dit betekent dat de wijziging gecontroleerd en geaccordeerd wordt voordat deze doorgevoerd wordt in de database van Pluriform.

Om een beeld te krijgen van de werking van deze taak heb ik een stappenplan opgesteld (zie ook figuur 25):

1. De taak kan worden uitgevoerd door een gebruiker van het systeem.
2. De taak wordt getriggerd op het moment dat een gebruiker van het systeem op de knop ‘verwerk geselecteerde wijzigingen’ klikt.
3. De taak controleert van welk type het te wijzigen attribuut is.
4. Als dit duidelijk is vormt de taak de waarde van het wijzigingsvoorstel om naar het juiste type.
5. Daarna voert de taak de wijziging door op het attribuut in een context met een bepaalde waarde.
6. Daarna zet de taak de status van het wijzigingsvoorstel op doorgevoerd.



Figuur 25 Werking taak wijziging doorvoeren

8.2.2 ONTWIKKELEN VAN DE WIJZIGINGSINTERFACE

Ontwikkelen van de GUI

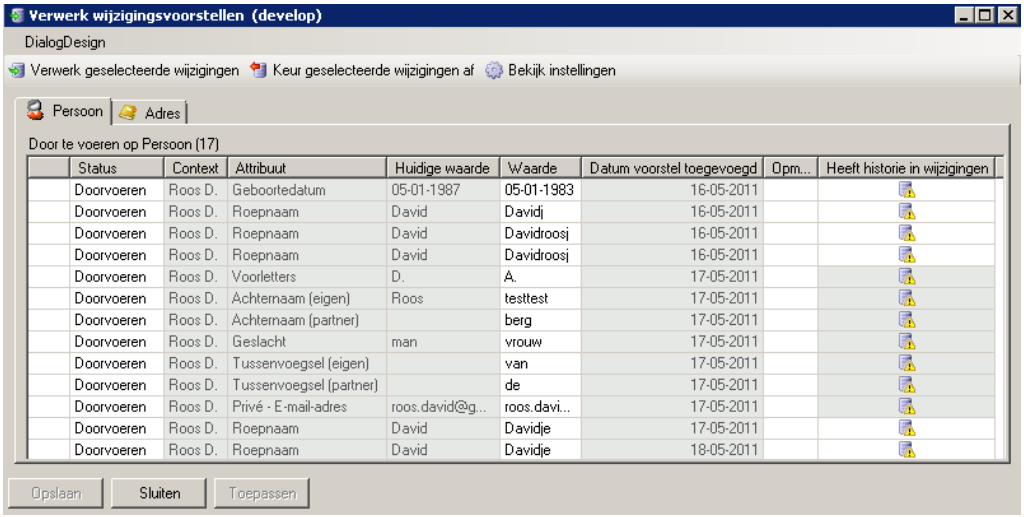
Zoals beschreven in de requirements voor het actiescherm heb ik besloten om een control center te maken waarmee het actiescherm aangeroepen kan worden. Door in de klasse Control centers een nieuw control center aan te maken kan ik vervolgens een zogeheten dialog aanmaken. Deze dialog, *verwerk wijzigingsvoorstellen,* heeft de mogelijkheid om event scripts aan te maken. Uiteraard is de dialog ingesteld als Singleton, zodat er maar één instantie kan draaien in runtime. Het is de bedoeling dat bij het openen van de dialog alle wijzigingsvoorstellen worden getoond met de status doorvoeren. Hierdoor heb ik besloten om een OnLoadDialog event script te maken om zo deze wijzigingsvoorstellen in de dialog te tonen. Maar ik had het probleem dat de dialog niet gevuld kan worden in het event script. Daarom heb ik besloten om een struct aan te maken en die te vullen in het OnLoadDialog event script. In het struct heb ik verschillende members van het type group opgenomen voor de verschillende contexten waarin een wijzigingsvoorstel kan bestaan. De contexten heb ik afgeleid uit de selectie van objecten zoals die zijn opgenomen in de requirements.

Figuur 26 Selectie van wijzigingsvoorstellen bij het openen van het actiescherm

Vervolgens heb ik binnen de dialog een aantal dialogcontrols toegevoegd die allemaal volgens de richtlijnen van Matthat Software B.V. voor een GUI zijn:

- Een tabcontainer. Deze control bevat tabbladen voor de verschillende contexten waarin een wijzigingsvoorstel kan bestaan (dat zijn op dit moment *Persoon, Adres, Gift* en *Toezegging*). Op ieder tabblad heb ik een editable grid toegevoegd waarin de wijzigingsvoorstellen getoond zullen worden. De editable grid bestaat uit de kolommen *status, context, attribuut, huidige waarde, waarde, datum voorstel toegevoegd* en *opmerkingen.* In de grid is het mogelijk om de waarden in de kolommen *status, waarde* en *opmerkingen* te wijzigen.
- Een toolbar. Deze control heb ik op de dialog geplaatst waarin ik de knoppen *verwerk geselecteerde wijzigingen, keur geselecteerde wijzigingen af* en *bekijk instellingen* heb toegevoegd.
- Een buttonbox. Deze control bevat de knoppen *Opslaan, Sluiten/Annuleer* en *Toepassen*.

Door aan elke grid aan te geven welk struct er ingeladen moet worden kunnen de wijzigingsvoorstellen getoond worden onder de verschillende tabbladen en in de verschillende contexten. Het resultaat is te zien in Figuur 27.



Figuur 27 GUI verwerk wijzigingsvoorstellen

Ontwikkelen van de generic update or change request

Zoals aangegeven in het ontwerp zijn de binnenkomende parameters op de generic een context, attribute en value. Ik heb besloten om eerst een aantal controles te laten uitvoeren door de generic om het volgende na te gaan:

1. Zijn alle parameters gevuld?
2. Is de huidige waarde van het opgegeven attribuut anders als de nieuwe waarde? Zo ja dan mag je concluderen dat het om een wijziging gaat.
3. Zit het te wijzigen attribuut in de groep ‘Attributen direct muteerbaar’? Zo ja dan mag de generic direct een mutatie doen. Zo niet, dan zal de generic een wijzigingsvoorstel aanmaken nadat is gecontroleerd of er niet al eenzelfde wijzigingsvoorstel bestaat in het systeem.

De generic gaat een wijziging direct muteren met het script:

```
1 // direct muteren met operation "assign"
2 assign
    object context
  attribute attribute
    value Case select
          Switch [ ] attribute
                 ->Attribute
                   Type
          [ Amount=> value (amount) [opt]
          | Text=> value (text) [opt]
          | Pointer=> value (pointer) [opt]
          | Date=> value(date) [opt]
          |
            Else skip
```

De generic gaat een wijzigingsvoorstel aanmaken met het script:

```
1 // maak wijzigingsvoorstel
2 discard new Wijzigingsvoorstel
        constructor
          context context
          attribute attribute
          waarde ( value (text) [opt] OTHERWISE ( Universal pointer to string: +class OTHERWISE ( value(date) [opt] OTHERWISE value (amount) [opt]))
                                                  ptr value (pointer) [opt]
```

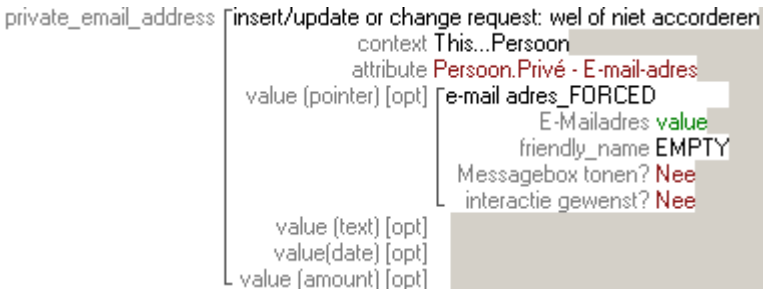
8.3 UITVOEREN TRANSITION FASE: IMPLEMENTATION AND TEST

Om zowel het actiescherm als de werking van de generic te testen maak ik in het systeem een testcase. Deze testcase voert de generic uit met handmatig opgegeven waarden voor de parameters. Bij het testen van het actiescherm en het uitvoeren van de testcase kwam ik een aantal problemen tegen:

XML waarden converteren
Doordat een attribuut in Pluriform van verschillende datatypes kan zijn en de opgegeven nieuwe waarde uit een wijzigingsvoorstel over het algemeen platte tekst is heb ik een oplossing moeten bedenken hoe uiteindelijk bij het doorvoeren van het voorstel de waarde van het juiste datatype is. Hieronder wordt een voorbeeld beschreven om het probleem te verduidelijken.

Een persoon wil via de website van een klant zijn nieuwe e-mailadres opgeven. In Pluriform is een e-mailadres die bij een persoon hoort van het type pointer. De waarde die de persoon doorgeeft en die uit het XML bericht updatePerson komt is platte tekst. De datatypen zijn dus niet overeenkomstig en dus zal de platte tekst geconverteerd moeten worden naar het juiste datatype van Pluriform.

Uit bovenstaand voorbeeld heb ik besloten om de generic, die de controle moet doen of een attribuut direct gewijzigd mag worden of dat er een wijzigingsvoorstel aangemaakt moet worden, aan te passen. Bij het inzetten van de generic wordt nu de context (de persoon), het attribuut (e-mailadres) en uiteraard de waarde (het nieuwe e-mailadres) opgegeven. De parameter waarde komt vier keer voor, voor elk datatype (pointer, text, date of amount) een keer. In het voorbeeld van hierboven zal de waarde geconverteerd worden naar een pointer met gebruik van bestaande generics (zie Figuur 28). Deze oplossing ondervangt het probleem dat er bij het doorvoeren van een wijzigingsvoorstel een datatype conflict optreedt.



Figuur 28 Platte tekst converteren naar het juiste datatype

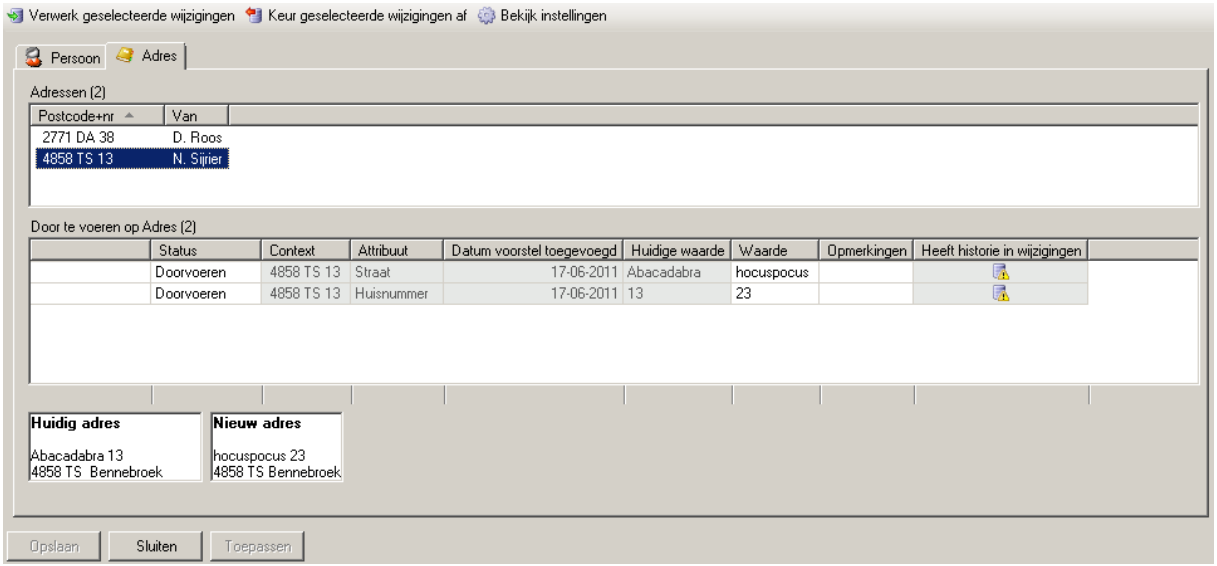
Transaction versus parallel transaction
Er ontstond een probleem bij het wijzigen van een waarde of status van een wijzigingsvoorstel in de editable grid. Elke keer als de gebruiker iets wilde wijzigen trad er een foutmelding op: Geen parallelle transactie gestart. Ik ben hiermee naar een collega gegaan maar die kon mij niet vertellen wat dit betekende omdat er geen ervaring was met een editable grid. Ik ben uiteindelijk nog eens door de instellingen van de dialog gegaan en kwam een veld Transaction tegen. Door deze aan te vinken was de melding verdwenen.
Uiteindelijk begrijp ik de foutmelding. Bij het opstarten van de dialog wordt er automatisch een transactie opgestart. Als er wijzigingen gedaan worden op de velden status of waarde van een wijzigingsvoorstel moet dus verplicht worden aangegeven dat die wijzigingen worden meegenomen in de huidige transactie en doorgevoerd worden na het klikken op *Toepassen* (transactie voltooien). Als je dit niet wilt als modelleur zal je dit moeten ondervangen door een parallelle transactie te starten voor elke wijziging die gedaan wordt door de gebruiker. Omdat de knoppen *verwerk geselecteerde wijzigingsvoorstellen* en *keur geselecteerde wijzigingen af* al in de naam aangeven dat de actie direct wordt uitgevoerd heb ik besloten om hiervoor een parallelle transactie te starten. Deze oplossingen ondervangen het probleem dat er geen parallelle transactie wordt gestart bij het wijzigen van een waarde of status van een wijzigingsvoorstel.

Samenhang tussen attributen

In Pluriform zijn attributen niet afhankelijk en gerelateerd aan elkaar, dit geeft enerzijds maximale vrijheid maar kan anderzijds ook problemen opleveren. Op het moment dat er bijvoorbeeld door een persoon een adres wordt toegevoegd of gewijzigd via de website van een klant kan het zijn dat de opgegeven waarden samen niet een bestaand adres vormen. Dit zal niet alleen bij een adres zo zijn maar ook bij andere contexten die nu nog niet opgenomen zijn in het systeem. Per context zal de oplossing verschillend zijn, maar om bovenstaand probleem van het adres af te vangen heb ik besloten om een validatie check bij het doorvoeren van een adres in te bouwen. Deze check vergelijkt de ingevoerde adresgegevens met adresXpress. Als adresXpress meldt dat de gegevens geen bestaand adres opleveren zal er een foutmelding teruggestuurd worden naar de website. Toch zal dit afhankelijk zijn van wat de klant van Matthat Software B.V. wil met de melding van adresXpress. De klant kan er namelijk ook voor kiezen om het niet bestaande adres toch door te voeren in de database.

Zoals te zien is in Figuur 8, heb ik een service operation getAddressXpress gemaakt. Door hiervan gebruik te maken op de website hoeft een persoon alleen een postcode en huisnummer in te voeren en zullen de overige adresgegevens worden opgehaald uit adresXpress. Deze oplossing ondervangt al op de website dat een persoon niet bestaande adresgegevens doorstuurt naar Pluriform.

Om aan te tonen dat het mogelijk is om bijvoorbeeld alle wijzigingsvoorstellen van een adres gegroepeerd te zien heb ik dat ontwikkelt met iets meer virtuele interactie waardoor de gebruiker het huidige adres en het nieuwe adres ziet. Het nieuwe adres wordt afgeleid uit de wijzigingsvoorstellen op het specifieke adres (zie Figuur 29).



Figuur 29 Groeperen op een specifiek object

8.3.1 EVALUATIE

Ondanks de problemen die ik tegenkwam tijdens het testen van de wijzigingsinterface kon ik deze allemaal oplossen en heb ik rekening kunnen houden met de functionele eisen. Ik heb geleerd om rekening te houden met koppeling, decompositie, data persistence en inkapseling, waarbij ik vooral let op toekomstige wijzigingen, testbaarheid en hergebruik. Daarnaast heb ik geleerd hoe Pluriform omgaat met transacties op de database, om de juiste dialoogstructuur toe te passen in de GUI volgens de richtlijnen van Matthat Software B.V. en om de juiste oplossingen te bedenken bij een technisch probleem.

8.3.2 VAN ITERATIE 2 NAAR ITERATIE 3

Nu de wijzigingsinterface is ontwikkelt en getest kan ik de service operations ontwerpen en ontwikkelen voor XML uitwisseling. Dit zal ik doen in een derde iteratie van het project waarbij ik gebruik zal maken van de module ‘XML framework’ en het SOAP communicatieprotocol.

9 ITERATIE 3# ONTWIKKELEN VAN SERVICE OPERATIONS VOOR XML UITWISSELING

Na het ontwikkelen van een wijzigingsinterface ga ik service operations ontwerpen en ontwikkelen die de XML berichten definiëren voor het uitwisselen van gegevens tussen Pluriform goede doelen en een website.

Het hoofdstuk is opgedeeld in drie fases van Unified Process, namelijk elaboration fase (9.1), construction fase (9.2) en de transition fase (9.3).

9.1 UITVOEREN ELABORATION FASE: ANALYSIS AND DESIGN

9.1.1 HET FUNCTIONELE ONTWERP VOOR DE SERVICE MESSAGES

De belangrijkste vraag voor mij in het klantonderzoek is de vraag: '*U geeft aan dat u interesse heeft in een rechtstreekse koppeling van Pluriform Goede Doelen met de website. Welke gegevens zouden beschikbaar moeten zijn op de website?'.* Deze vraag werd gesteld als klanten aangaven echt interesse te tonen voor een koppeling tussen Pluriform en hun website. Het doel van de vraag is om wensen op dit gebied naar boven te halen.
Van de 18 respondenten antwoorden 16 respondenten, verdeeld over 10 klanten, op bovenstaande vraag. Onderstaand is een overzicht van de overeenkomende antwoorden die gegeven zijn door deze respondenten (voor het volledig onderzoeksrapport wordt verwezen naar bijlage I).

Antwoord	**Komt aantal x voor**
Naam, adres en woonplaats (NAW) gegevens van donateur, persoon of vrijwilliger	11x verdeelt over 8 klanten
Gift- en toezeggingsgegevens	10x verdeelt over 7 klanten

De twee antwoorden werden door zoveel klanten gegeven dat het voor mij wel duidelijk werd in welke context ik moest gaan zoeken binnen Pluriform. Het eerste antwoord leidt tot een aantal klassen in Pluriform. Allereerst zijn donateurs en vrijwilligers in Pluriform allemaal een persoon. In de klasse persoon zit dan ook alle informatie voor NAW gegevens. Alleen het adres van een persoon komt uit de klasse adres. Deze redenering leidde tot de keuze van de klasse Persoon en de klasse Adres. De gegevens uit het tweede antwoord leidden tot de klassen Toezegging en Gift in Pluriform.
Ik heb besloten dat alles wat buiten de scope van de klassen Persoon, Gift, Adres en toezegging ligt, buiten de grenzen van de te ontwikkelen service operations valt. Zoals ik later gemerkt heb tijdens het opstellen van domein-klassediagrammen zal er toch informatie uit verschillende andere klassen nodig zijn om te voorzien in de eisen. Denk hierbij bijvoorbeeld aan een klasse Constanten waar geslacht en burgerlijke staat vandaan komen.

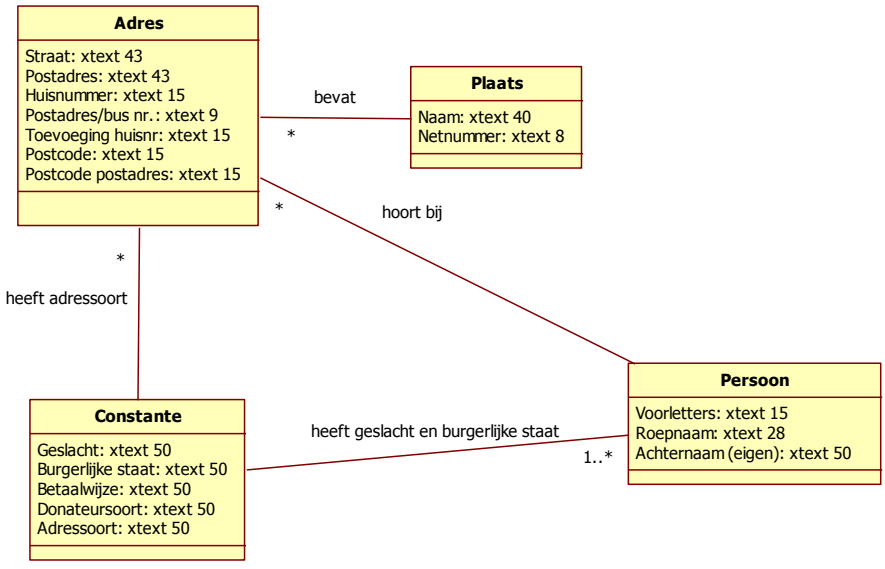
Bepalen van de interactie
Voor elke use-case heb ik eerst zo concreet mogelijk vastgesteld hoe de interactie tussen gebruiker en het systeem zal verlopen. Zie bijvoorbeeld de use-case beschrijving 'wijzig mijn adresgegevens' in Tabel 5.

Naam	Wijzig mijn adresgegevens
Samenvatting	De Service Operation updateAddress zorgt voor het wijzigen van adresgegevens van 1 adres uit Pluriform aan de hand van een relatienummer en te wijzigen adresgegevens met archivecode.
Aannamen	- Service is bereikbaar over HTTPS - Actor is geauthentiseerd door het systeem en ingelogd op de website - Relatienummer van actor is bekend in de sessie van de website - Actor heeft de mogelijkheid om een wijziging van adresgegevens door te geven via de website
Beschrijving	(1) Actor geeft aan zijn adresgegevens van 1 adres te willen wijzigen via de website (2) Actor selecteert een adres en wijzigt zijn adresgegevens via de website. Als

	Pluriform een licentie op adresXpress heeft treedt uitzondering [Haal adresgegevens op via adresXpress] op. (3) Website roept de operation updateAddress aan met de verwachte input parameters (4) Het systeem controleert via de klasse wijzigingsinstellingen voor iedere input parameter of de waarde direct gemuteerd mag worden in Pluriform, als dit niet het geval is treedt uitzondering [Input parameter moet worden geaccordeerd] op (5) Het systeem onthoudt per wijzigingsvoorstel het te wijzigen attribuut in groep 2 (6) Het systeem wijzigt per input parameter de waarde in het systeem (7) Het systeem onthoudt elk gemuteerd attribuut in groep 1 (8) Het systeem stuurt de een response message naar de website met een bericht dat de wijzigingen zijn uitgevoerd op groep 1 en dat er een voorstel is gemaakt op groep 2
Uitzonderingen	[Haal adresgegevens op via adresXpress] Sub-case 'Ophalen van adresgegevens uit adresXpress' wordt eerst afgehandeld door het systeem, daarna loopt use-case door. [Input parameter moet worden geaccordeerd] Sub-case 'maak wijzigingsvoorstel aan' wordt eerst afgehandeld door systeem daarna loop use-case door.
Resultaat	Website ontvangt een bevestiging vanuit Pluriform, actor heeft inzicht in de bevestiging

Tabel 5 Use-case beschrijving wijzig mijn adresgegevens

Ik heb gekozen voor het opstellen van een UML domein-klassediagram voor het vaststellen van de aanwezige data in Pluriform en om een koppeling te leggen tussen de requirements. Aan de hand van de requirements voor het systeem heb ik de geselecteerde klassen uitgediept in Pluriform. Door net als bij het ontwerpen van het POC de definitie van de bestaande klassen te bekijken heb ik logische selecties gemaakt uit de benodigde attributen die de juiste informatie moeten gaan leveren in het gebruik van de service operations. Het voordeel wat ik hier had was dat er hiervoor geen gedetailleerde requirements opgesteld zijn. Bij het opstellen van de domein-klassediagrammen, met gebruik van de tool StarUML, is het dan ook de bedoeling dat ik een basis aanbiedt die later aangevuld kan worden met specifieke en gedetailleerde wensen van klanten, maar dit zal niet binnen dit project vallen. Door gebruik te maken van de bestaande klassen kan ik concluderen dat het systeem zeker een standaardisatie oplevert en ingezet kan worden bij alle klanten. Alle klanten maken namelijk gebruik van deze klassen. Omdat het beschrijven van alle ontwerpen voor de service operations onoverzichtelijk wordt heb ik besloten om in dit verslag als voorbeeld de use-case 'wijzig mijn adresgegevens' te gebruiken. Voor het volledige functionele ontwerp voor de service operations wordt verwezen naar bijlage K. Figuur 30 geeft het resultaat van het domein-klassediagram voor betrokken klassen en attributen bij de use-case 'wijzig mijn adresgegevens'.



Figuur 30 domein-klassediagram adres

Na het opstellen van het domein-klassediagram kan ik de attributen selecteren die verwacht worden als velden in het binnenkomende XML bericht op service operation updateAddress.

- Straat
- Huisnummer
- Toevoeging huisnr
- Postcode
- Plaats
- Adressoort
- Archivecode
- Relatienummer

Het uitgaande XML bericht uit de service operation updateAddress zal 1 veld bevatten genaamd message. Afhankelijk van wat een klant in dit veld wil, kan er een statuscode of bericht in gezet worden.

Elke service operation die een wijziging in de database wil doorvoeren zal gebruik maken van de generic 'update or change request'. Om te voldoen aan de beveiligingseis zal elke operation gebruik maken van een SOAP header in het XML bericht. Hierin wordt een username en wachtwoord aangegeven waardoor het systeem de bezoeker die via de website een service operation initieert authentificeren.

9.1.2 HET TECHNISCHE ONTWERP VAN DE SERVICE OPERATIONS

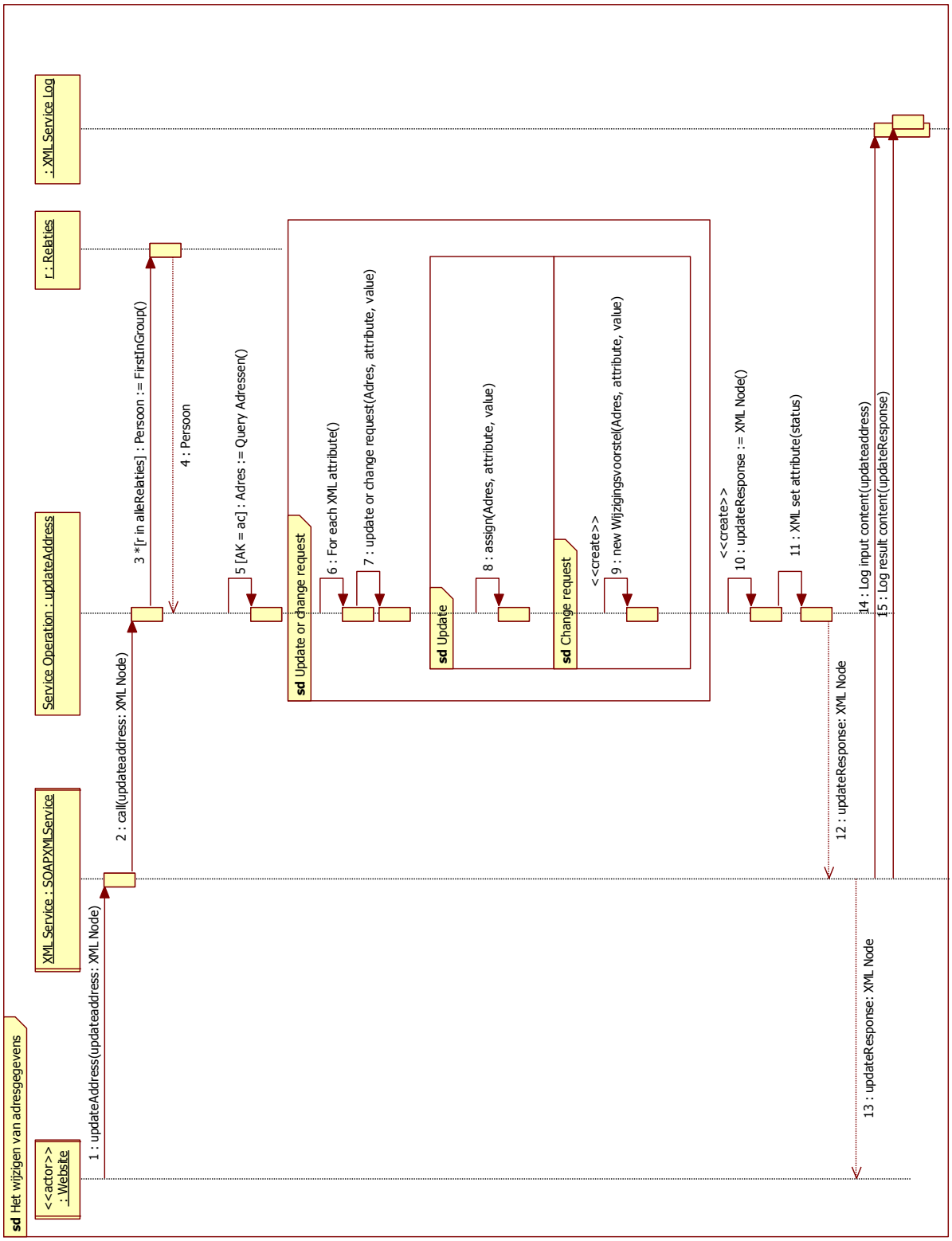
Aan de hand van het functioneel ontwerp kan ik het binnenkomende XML bericht op de service operation updateAddress definiëren:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:plur="pluriform:SOAPXMLService">
  <soap:Header>
    <plur:credentials>
      <plur:username>?</plur:username>
      <plur:password>?</plur:password>
    </plur:credentials>
  </soap:Header>
  <soap:Body>
    <plur:updateAddress archivecode="?" city="?" housenumber="?" housenumber_suffix="?" relationnumber="?" street="?" type="?"
zipcode="?"/>
  </soap:Body>
</soap:Envelope>
```

Het uitgaande XML bericht wordt na afhandeling door de service operation updateAddress als volgt door mij gedefinieerd:

```
<Envelope xmlns="http://www.w3.org/2003/05/soap-envelope">
  <Body>
    <updateAddressResponse message="?" xmlns="pluriform:SOAPXMLService"/>
  </Body>
</Envelope>
```

Nu ik precies weet welke velden ik verwacht in de binnenkomende en uitgaande XML berichten kan ik het gedrag en de structuur van de service operation ontwerpen. Om dit te ontwerpen heb ik gekozen voor het opstellen van een sequence diagram. Ik heb hiervoor gekozen omdat een sequence diagram inzicht geeft in de interactie tussen de objecten die nodig is om het gedrag van het systeem te verwezenlijken. Figuur… toont het diagram voor de use-case 'wijzig mijn adresgegevens. Het diagram toont een interactie met nadruk op het tijdsaspect. In het diagram worden de exacte boodschappen tussen de objecten aangegeven, inclusief eventuele parameters en is te zien dat er gebruik wordt gemaakt van componenten uit het XML framework.



Figuur 31 Sequence diagram voor use-case 'wijzig mijn adresgegevens'

9.2 UITVOEREN CONSTRUCTION FASE: DESIGN AND IMPLEMENTATION

9.2.1 PROBLEEM ONTWIKKELEN SERVICE OPERATIONS

Op het moment dat ik begon met de tweede iteratie voor het ontwikkelen van SOAP Service Operations werd bekend dat het bedrijf Pluriform Software de module XML framework, die nodig is om de Service Operations te ontwikkelen, had afgeschermd en afgesloten voor verdere ontwikkeling in de Pluriform ontwikkelversie van Matthat Software B.V. Reden hiertoe was het framework wat nog in bèta fase verkeerde. Dit werd ook duidelijk toen er een demo gegeven werd over het inrichten van het framework want autorisatie werkte nog niet naar behoren.
Het geheel leverde een vrij groot probleem op, aangezien het de werkzaamheden van het afstuderen volledig stil zou leggen. Het werd nog erger toen bekend werd dat Matthat Software B.V. pas eind april over zou stappen naar de meest recente versie van Pluriform software.

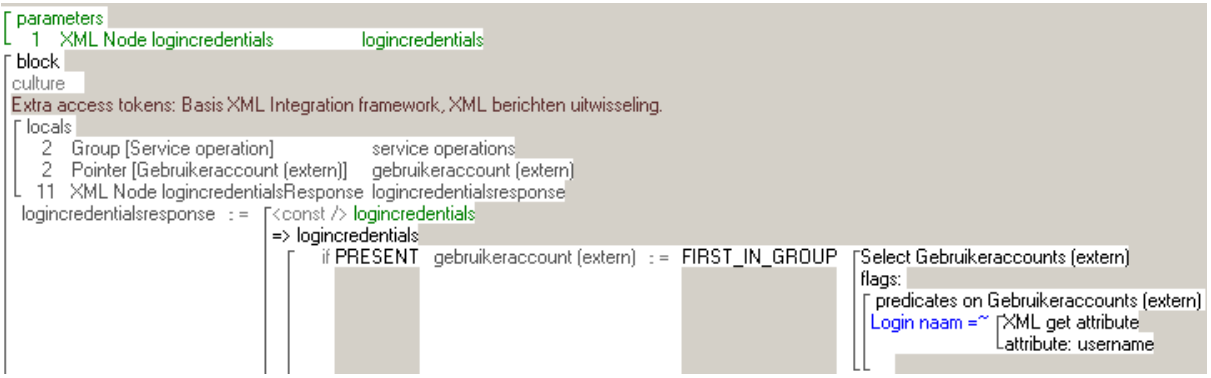
Hiertoe heb ik besloten om bij Pluriform Software te vragen naar de laatste ontwikkelversie van Pluriform voor het inrichten van een XML framework. Deze versie heb ik gekregen op voorwaarde dat het ontwikkelen van het framework in projectvorm met Pluriform Software zou gaan. Dit betekent concreet de problemen die je tijdens het ontwikkelen tegenkomt in het framework worden voorgelegd aan Pluriform Software. Deze zullen vervolgens, na akkoord voor de kosten, de problemen oplossen. Hierdoor zouden problemen en bugs direct verholpen kunnen worden. Dit heb ik besproken met Matthat Software B.V. en die gingen akkoord. Ik kon uiteindelijk na installatie van de nieuwe versie beginnen aan het ontwikkelen van het systeem.

Dit probleem moest uiteindelijk doorgevoerd worden in de opgestelde begroting voor het project. Het in projectvorm werken met Pluriform Software zou neerkomen op een week aan projecttijd wat natuurlijk door Matthat Software B.V. betaald moet worden.

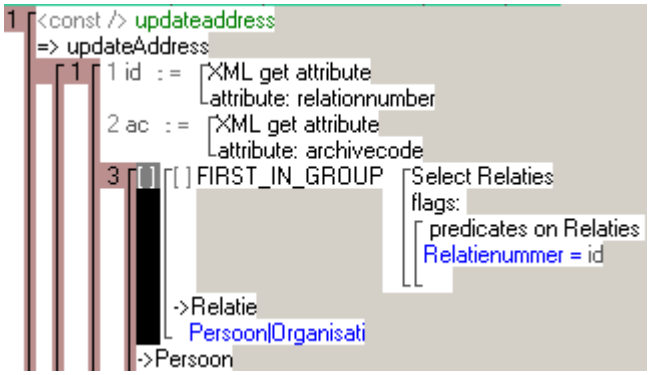
9.2.2 ONTWIKKELEN VAN DE SERVICE OPERATIONS

Voor het inzetten van SOAP XML berichten heb ik gebruik gemaakt van de klasse Service Operations. Een nieuwe Service Operation verwacht een aantal onderdelen. Hoe ik deze onderdelen implementeer is uitgelegd in paragraaf 7.2 tijdens het ontwikkelen van het POC. Ik zal in deze paragraaf wel verder ingaan op het handle request script van de service operation updateAddress

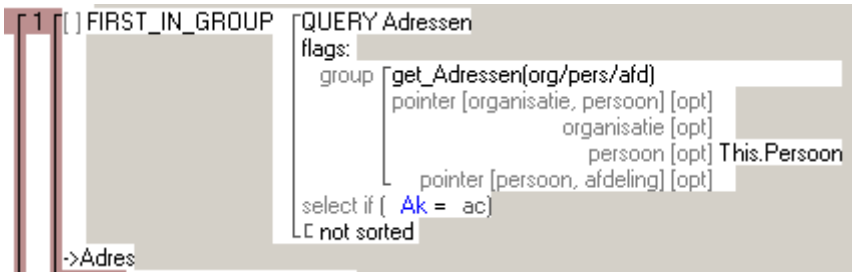
De service operation updateAddress
Het script moet ervoor zorgen dat de input message op een juiste manier wordt afgehandeld en de juiste output message met de juiste gegevens retourneert.
Allereerst worden de opgegeven username en password gecontroleerd in de klasse Gebruikersaccount (extern) om de bezoeker die via de website de service operation initieert te authentificeren.



Vervolgens haal ik uit het binnenkomende XML bericht het relatienummer op en de unieke identifier van een adres, het archivenumber, op. Daarna kan ik met het relatienummer de juiste persoon in Pluriform opzoeken.

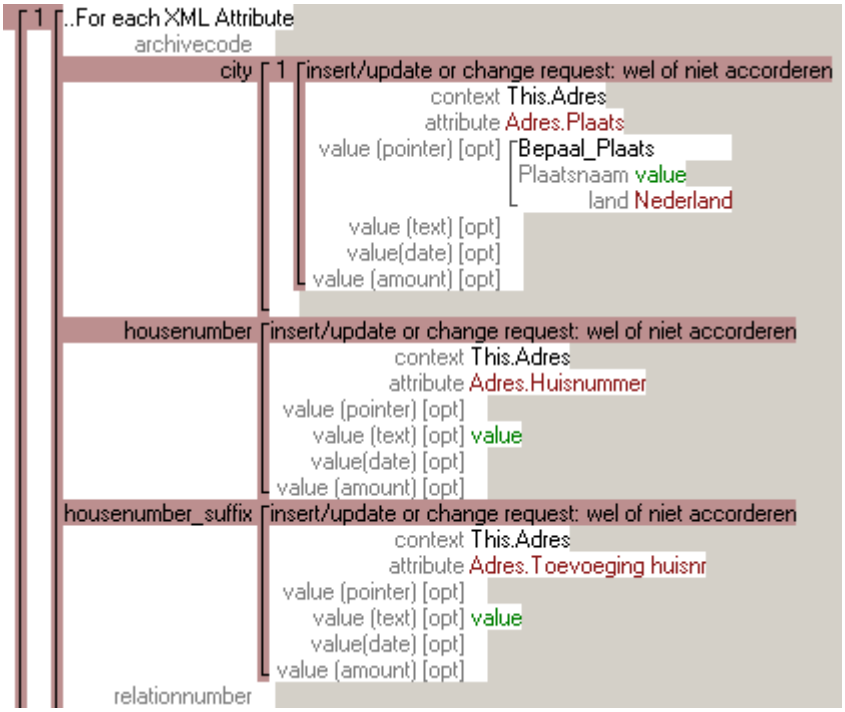


Nu ik de juiste persoon bij het relatienummer gevonden heb, kan ik het adres opzoeken van die persoon met het opgegeven archivenumber.



Omdat ik nu het juiste adres gevonden heb ga ik elk veld uit het XML bericht door de generic update or change request sturen. Zo zal per veld bekeken worden of het:

1. Een wijziging betreft
2. Zo ja, of hij direct gemuteerd mag worden in de database van Pluriform
3. Of er al eenzelfde wijzigingsvoorstel bestaat in de klasse wijzigingsvoorstellen



Als alle attributen behandeld zijn wordt er door de service operation een uitgaand XML bericht opgesteld.

```
2 updateresponse := ⌈XML Node
                    XML Node: updateAddressResponse
                      ⌈XML set attribute
                      attribute: message
                      ⌊value "address updated succesfully"
```

Als een persoon zich niet kan authentificeren en dus geen gebruikersaccount heeft in de klasse Gebruikersaccount (extern) van Pluriform zal er een melding terug gestuurd moeten worden die dit aangeeft. Daarom heb ik ervoor gekozen om een extra optioneel attribuut error op te nemen in het uitgaande XML bericht.

```
        else ⌈1 // wachtwoord niet correct
             2 ⌈</> logincredentialsresponse
               => logincredentialsResponse
                 ⌈XML set attribute
                 attribute: error
                 ⌊value "Password is incorrect!"
    2 logincredentialsresponse
else ⌈1 /* gebruikersnaam niet gevonden */
     2 ⌈</> logincredentialsresponse
       => logincredentialsResponse
         ⌈XML set attribute
         attribute: error
         ⌊value "Username is incorrect!"
     3 logincredentialsresponse
```

9.3 UITVOEREN TRANSITION FASE: IMPLEMENTATION AND TEST

Het belangrijkste om te testen in het systeem zijn de gegevensstromen en de verwerking daarvan. Door tijdens het ontwikkelen onderdelen te testen werd voor mij duidelijk of ik het ontwikkelproces goed volgde. Door gebruik te maken van de testtool soapUI kon ik het mogelijk maken om vanuit elke locatie het systeem aan te roepen. Met de tool kon ik in- en uitvoer XML berichten van de service operations goed testen. Hieronder is een procesmatig overzicht van het testen van één van de service operations, updateAddress.

In het functionele ontwerp staat aangegeven welke velden in het binnenkomende XML bericht aanwezig moeten zijn en hoe de waarde eruit moet zien. Zoals al eerder aangegeven levert SOAP een Web Service Description Language document (WSDL) waarin eigenlijk precies hetzelfde wordt aangegeven. Het voordeel van de testtool is dan ook dat een WSDL geïmporteerd kan worden en daardoor voor gedefinieerde XML berichten gegenereerd worden. Als ik de WSDL ga bekijken ziet het er voor de service operation updateAddress als volgt uit:

```
<xsd:element name="updateAddress">
  - <xsd:complexType>
        <xsd:sequence/>
        <xsd:attribute name="archivecode" type="xsd:integer"/>
        <xsd:attribute name="city" type="xsd:string"/>
        <xsd:attribute name="housenumber" type="xsd:string"/>
        <xsd:attribute name="housenumber_suffix" type="xsd:string"/>
        <xsd:attribute name="relationnumber" type="xsd:integer"/>
        <xsd:attribute name="street" type="xsd:string"/>
        <xsd:attribute name="type" type="xsd:string"/>
        <xsd:attribute name="zipcode" type="xsd:string"/>
    </xsd:complexType>
</xsd:element>
```

Bovenstaande figuur laat alle verwachte velden zien. Hierdoor weet SoapUI hoe hij de XML berichten moet aanmaken. Het invullen van waarden zal uiteindelijk gebeuren via de website van en klant en waarschijnlijk via een formulier met voor gedefinieerde velden. Met de tool kan ik dit vervangen. Ik heb besloten om twee testen te doorlopen voor deze service operation. Allereerst zal ik alle attributen van adres toevoegen aan de wijzigingsinstellingen. Dit betekent dat alle waarden die binnenkomen via updatAddress direct gemuteerd worden. Vervolgens zal ik alle attributen weer verwijderen om te testen of er voor het adres een wijzigingsvoorstel aangemaakt wordt.

Het binnenkomende XML bericht op de server geeft aan het huisnummer veld waarde ‘40’, wat betekent dat het huisnummer moet veranderen van 38 naar 40:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:plur="pluriform:SOAPXMLService">
  <soap:Header>
    <plur:credentials>
      <plur:username>---</plur:username>
      <plur:password>---</plur:password>
    </plur:credentials>
  </soap:Header>
  <soap:Body>
    <plur:updateAddress relationnumber="12483" street="Valkenburgerlaan" housenumber="40" housenumber_suffix=""
zipcode="2771 DA" city="Boskoop" archivecode="16865" type="Privé"/>
  </soap:Body>
</soap:Envelope>
```

Het uitgaande XML bericht naar soapUI is als volgt:

```
<Envelope xmlns="http://www.w3.org/2003/05/soap-envelope">
```

```
  <Body>
    <updateAddressResponse message="address updated succesfully" xmlns="pluriform:SOAPXMLService"/>
  </Body>
</Envelope>
```

Het uitgaande XML bericht geeft aan dat het adreswijziging succesvol is afgehandeld door het systeem. Door het adres te raadplegen weet ik of dit daadwerkelijk zo is:

Privé-adressen (1)

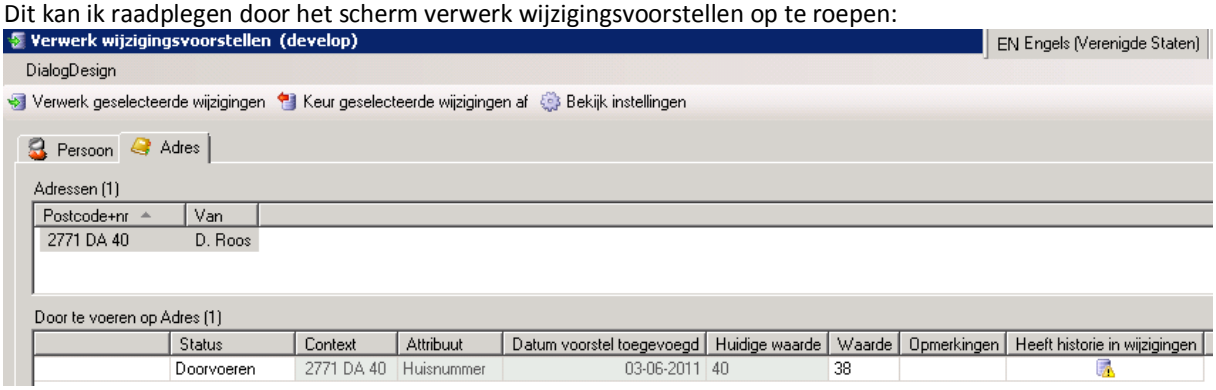
Pc+hnr	Straat	Huisnr	Postcode	Plaats	Telefoon	Contactgegevens		
2771 DA 40	Valkenburgerlaan	40	2771 DA	Boskoop				

Nu haal ik in het systeem uit de klasse wijzigingsinstellingen het attribuut huisnummer weg, waardoor er dus een wijzigingsvoorstel aangemaakt zou moeten worden door het systeem als ik het huisnummer weer naar 38 wil veranderen.

Het uitgaande XML bericht naar soapUI geeft aan dat er voor huisnummer een wijzigingsvoorstel is aangemaakt:

```
<Envelope xmlns="http://www.w3.org/2003/05/soap-envelope">
  <Body>
    <updateAddressResponse message="Change request created for fields: housenumber"
xmlns="pluriform:SOAPXMLService"/>
  </Body>
</Envelope>
```

Dit kan ik raadplegen door het scherm verwerk wijzigingsvoorstellen op te roepen:



Figuur 32 Een wijzigingsvoorstel op een adres

Zoals te zien in Figuur 32 is er inderdaad een wijzigingsvoorstel aangemaakt. Nu kan ik het voorstel selecteren en op verwerk drukken. Het attribuut wordt dan gemuteerd naar de nieuwe waarde.

Privé-adressen (1)

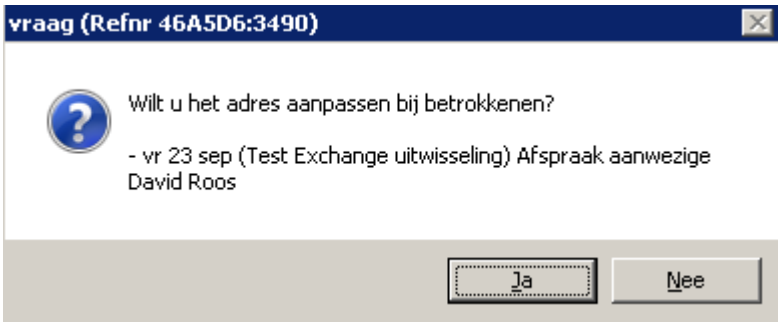
Pc+hnr	Straat	Huisnr	Postcode	Plaats	Telefoon	Contactgegevens		
2771 DA 38	Valkenburgerlaan	38	2771 DA	Boskoop				

Het Pluriform systeem zorgt voor automatische exception en error handling. Mocht er een error of exception ontstaan dat kan deze altijd terug gevonden worden in de logs van Pluriform. Zelfs alle binnenkomende en uitgaande XML berichten worden automatisch gelogd. Daarnaast is de module ‘XML framework’ zo ingericht dat het een SOAP error kan opvangen. Op het moment dat er een SOAP error ontstaat door een binnenkomend XML bericht zal Pluriform automatisch een gedetailleerd uitgaand XML bericht sturen naar de client.

Ondanks het veelvuldig testen tijdens de construction fase heb ik besloten om het framework als geheel nog eens op een aantal dingen te testen. Hierbij heb ik gekozen voor een dataflowtest als testspecificatietechniek. Dit is een test waarbij de verwerking door functies en de relaties tussen functies worden getest. De test wordt

vooral gebruikt bij het testen van de functionaliteit (de juistheid en volledigheid van de verwerking) in het kader van een black-box test (Martin Pol, 2002). Het testplan is opgenomen in bijlage M en de uitvoering in bijlage N. Na afhandeling van de test kon ik alles goedkeuren. Bij stap 2 en 14 had ik nog wel opmerkingen:

Stap 2 veroorzaakte een melding in Pluriform:



Dit heeft te maken met een trigger op de klasse Adres in Pluriform. Ik heb aangegeven aan mijn collega dat deze bij implementatie van het ontwikkelde systeem de trigger uitgezet moet worden.
Stap 14 zorgde er niet voor dat het wijzigingsvoorstel werd doorgevoerd door alleen een statuswijziging. Een mogelijke aanpassing zou zijn dat je de status niet handmatig kunt veranderen in 'Doorgevoerd'. Hiermee wordt voorkomen dat de gebruiker verwacht dat het wijzigingsvoorstel is doorgevoerd.

9.3.1 EVALUATIE

Tijdens deze derde iteratie heb ik besloten om alle ontwikkel werkzaamheden te doen in de testomgeving van Matthat Software B.V. omdat ik dan kon 'spelen', fouten mocht maken en andere bestaande componenten kon ombouwen. Achteraf had ik moeten ervoor moeten kiezen om alleen het POC hierin te ontwikkelen want ik moest uiteindelijk alle ontwikkelde service operations één voor één opnieuw maken in de live ontwikkelomgeving. Want alleen uit deze omgeving is het mogelijk om ontwikkelde software over te nemen in een Pluriform klantsysteem. Doordat ik hier niet goed over nagedacht heb kostte dit een dag aan werk.

Doordat tijdens het ontwikkelen van de service operations gebruik gemaakt is van bestaande klassen die ook beschikbaar zijn in de klantsystemen kan ik concluderen dat de service operations gestandaardiseerd zijn. Het gehele ontwikkelde systeem kan daardoor ingezet worden bij iedere klant. Uiteraard moet de klant dan de module 'XML framework' gaan gebruiken. Daarnaast levert het systeem een WSDL waardoor alle binnenkomende en uitgaande XML berichten gedwongen worden om generiek en volgens een standaard te worden opgezet.

Ik heb geleerd om op een efficiënte manier gebruik te maken van een bestaand framework (XML framework), keuzes te maken uit klassen om tot een standaardisatie te komen en een technisch ontwerp te gebruiken om bepaalde functionaliteit te verwezenlijken.

Overdragen van het systeem
Het systeem is uiteindelijk volledig ontwikkelt in de live ontwikkelomgeving van Pluriform. Alle documentatie voor onderhoud en implementatie kan worden aangeroepen vanuit de ingebouwde help-functie in Pluriform. Alle overige documenten zijn gekoppeld aan het afstudeerproject in Pluriform en zijn terug te vinden voor alle werknemers bij Matthat Software B.V.
De Deployment discipline uit UP is helaas niet gerealiseerd binnen het afstudeerproject. Op het moment van schrijven wordt er onderhandeld met meerdere klanten om het systeem te implementeren.

10 EVALUEREN

Dit hoofdstuk beschrijft het aantonen van het behalen van de beroepstaken zoals deze zijn opgegeven in het afstudeerplan (10.1), een evaluatie van de procesgang tijdens de afstudeerperiode (10.2) en een evaluatie van de opgeleverde (tussen)producten (10.3).

10.1 AANTONEN VAN HET BEHALEN VAN BEROEPSTAKEN

10.1.1 1.2 VOORBEREIDEN EN OPSTARTEN SOFTWARE ONTWIKKELTRAJECT

Beroepstaak 1.2 voorbereiden en opstarten software ontwikkeltraject vraagt om op hoofdlijnen een probleem te begrijpen, beschrijven en analyseren. Dit is op de meerdere onderdelen in het project behaald, te weten:

- Het opstellen en verdiepen van de afstudeeropdracht zoals beschreven in het tweede hoofdstuk ‘De afstudeeropdracht’.
  - Het beschrijven van de huidige situatie
  - Het beschrijven van de probleemstelling
  - Het beschrijven van de doelstelling
  - De concrete vertaling naar op te leveren producten
- Het inwerktraject die ik heb doorlopen
  - Het vaststellen en beschrijven van mijn rol en positie tijdens het afstuderen
  - De problemen en knelpunten waar ik tegen aan liep in het inwerktraject
  - Het verdiepen en inwerken in de terminologie en de ontwikkelomgeving
  - Het achterhalen van de componenten in de gebruikte module ‘XML framework’
- Het kiezen van een software ontwikkelmethode en een modelleertaal zoals beschreven in het derde hoofdstuk ‘Aanpak van de opdracht’
- Het opstellen van een planning
- Het opstellen van een begroting zoals beschreven in het vierde hoofdstuk ‘Uitvoeren inception fase: business modeling en requirements’
- Het achterhalen van de filosofie achter Pluriform, zoals beschreven in paragraaf 1.5.

Volgens het afstudeerplan wordt deze beroepstaak op niveau 3 uitgevoerd. Tijdens het afstudeerproject is deze beroepstaak op niveau 3 uitgevoerd vanwege het zelfstandig werken in een lastige context, door bovenstaande punten. Daarnaast is het analysen en begrijpen van Pluriform, wat ik beschreven heb in het eerste hoofdstuk ‘bedrijfsorganisatie’, een aspect wat zeker thuishoort bij het opstarten van een software ontwikkeltraject.

10.1.2 1.4 UITVOEREN ANALYSE DOOR DEFINITIE VAN REQUIREMENTS

Beroepstaak 1.4 uitvoeren analyse door definitie van requirements vraagt om het inventariseren, analyseren en beschrijven van de gewenste informatievoorziening en het analyseren, opstellen en beschrijven van de requirements. Dit is op onderdelen in het project gehaald, te weten:

- Het opzetten, inrichten en uitvoeren van het klantonderzoek naar de klantwensen en eisen en de huidige situatie met betrekking tot informatievoorziening zoals beschreven in paragraaf 4.2.
- Het vanuit onderzoek komen tot requirements zoals beschreven staat in het vijfde hoofdstuk ‘van onderzoek naar requirements’
- Het komen vanuit de requirements tot het functioneel ontwerp zoals deze is opgenomen in bijlage K en beschreven in de paragrafen 7.1, 8.1 en 9.1.

Niveau 3 is behaald vanwege het zelfstandig werken in een lastige context. Het gaat om één applicatie Pluriform die bestaat uit verschillende versies. Er zijn meerdere stakeholders, namelijk klanten, het bedrijf Matthat Software B.V. en website ontwikkelaars, hoewel van de laatste niet altijd sprake is. Door het inrichten en uitvoeren van het klantonderzoek kon ik komen tot de requirements. Hierbij heb ik rekening gehouden met

de kwaliteitsattributen uit ISO 9126. Na over en weer sturen van voorstellen naar de stakeholders hebben uiteindelijk alle stakeholders de requirements kunnen valideren en goedkeuren. Alle requirements zijn geprioriteerd aan de hand van de MoSCoW methode en komen uiteindelijk terug in het systeem, wat af te leiden is uit het functionele en technische ontwerp en de ontwikkelde producten.

10.1.3 2.1 OPSTELLEN GEGEVENSMODEL VOOR DATABASE

Beroepstaak 2.1 opstellen gegevensmodel voor database vraagt om het analyseren en beschrijven van gegevens over objecten in hun onderlinge samenhang. Dit is op onderdelen in het project gehaald, te weten:

- Het opstellen, door veelal reverse engineering, en analyseren van de domein-klassediagrammen voor het POC en de service operations aan de hand van de requirements zoals beschreven in de elaboration fase. Hierbij is een transformatie gemaakt van bestaande database naar een gegevensmodel
- Het ontwerpen en opstellen van het domein-klassediagram voor de wijzigingsinterface aan de hand van de requirements zoals beschreven in de elaboration fase
- Het gebruik maken van de tool starUML

Niveau 3 is behaald vanwege het zelfstandig werken in een lastige context. Zoals wordt aangetoond in de diagrammen uit het functionele ontwerp in bijlage K zijn er soms meerdere relaties tussen twee objecten. De modellen tonen een consistent beeld van het domein, waardoor ik het POC, de service operations en de wijzigingsinterface kon ontwikkelen. De modellen bevatten op meerdere plekken binaire en unaire relaties. Door het aanbrengen van multipliciteit heb ik structurele verbanden kunnen vastleggen waardoor er een compleet beeld van de bestaande database gegeven is.

10.1.4 3.1 ONTWERPEN SOFTWARE ARCHITECTUUR

Beroepstaak 3.1 ontwerpen software architectuur vraagt om het bepalen en beschrijven van de architectuur van de applicatie. Dit is op onderdelen in het project gehaald, te weten:

- Het opzetten van een architecture description document, zoals opgenomen in bijlage J en beschreven in het zesde hoofdstuk.
    - Het kiezen van een architectuurstijl en ontwerpstrategie
    - Het document is begrijpelijk beschreven voor alle stakeholders
    - De architectuur is beschreven vanuit de relevante viewpoints
    - Alle componenten en subsystemen met hun onderlinge relaties zijn beschreven
    - Er is rekening gehouden met de functionele eisen en relevante kwaliteitsattributen volgens ISO 9126
    - Er is rekening gehouden met verschillende scenario's die o.a. de belangen van de stakeholders weergeven.

Niveau 3 is behaald vanwege het zelfstandig werken in een lastige context. Ik heb het document zo ontworpen dat het geheel begrijpelijk is voor alle stakeholders. De architectuur heeft een standaardisatie opgeleverd die herbruikbaarheid in verschillende situaties en Pluriform applicaties garandeert van de verschillende functionaliteiten. De architectuur modellen heb ik ontworpen met gebruik van verschillende UML diagrammen.

10.1.5 3.2 ONTWERPEN SYSTEEMDEEL

Beroepstaak 3.2 ontwerpen systeemdeel vraagt om het beschrijven van systeemdelen, of componenten, hun structuur en het gedrag in detail, zodanig dat het bouwen mogelijk wordt. Dit is op onderdelen in het project gehaald, te weten:

- Het vanuit de requirements komen tot de use-cases en beschrijvingen van de verschillende service operations, het POC en de wijzigingsinterface zoals beschreven in het zesde hoofdstuk 'opstellen van een architectuur' en in de elaboration fase en opgenomen in het functioneel ontwerp in bijlage K.

- Het ontwerpen van de sequence diagrammen voor de service operations zoals beschreven in het negende hoofdstuk 'Iteratie 3# Ontwikkelen van service operations voor XML uitwisseling' en opgenomen in bijlage L.
- Het opstellen van de structuur van de binnenkomende en uitgaande XML berichten zoals beschreven in de elaboration fase van het POC en de service operations en opgenomen in het technische ontwerp in bijlage L.
- Het ontwerpen van de GUI van de wijzigingsinterface zoals beschreven in het achtste hoofdstuk 'Iteratie 2# ontwikkelen van een wijzigingsinterface' volgens de richtlijnen van Matthat Software B.V. en waarvan de layout overzichtelijk is.
- Het ontwerpen van de structuur van de generic 'update or change request' en de taak 'wijziging doorvoeren' zoals beschreven in het achtste hoofdstuk 'Iteratie 2# ontwikkelen van een wijzigingsinterface' en opgenomen in het technische ontwerp in bijlage L.
- Er is rekening gehouden met coupling, decompositie en inkapseling.
- Er is rekening gehouden met de requirements.

Niveau 3 is behaald vanwege het zelfstandig werken in een lastige context. Er is gebruik gemaakt van de ontwerpmethodiek UML, de tool StarUML en de ontwikkelomgeving van Pluriform. De GUI die ik ontworpen en gemodelleerd heb is geschikt voor toekomstige wijzigingen en toevoegingen en kan hergebruikt worden in alle Pluriform applicaties. Hetzelfde geld voor de service operations. Hiernaast heb ik rekening gehouden met verschillende aspecten waaronder data persistence en error and exception handling. Bij het ontwerpen heb ik tevens goed gelet op verschillende beveiligingseisen. De uitwerking hiervan vormde een login operation, SOAP header authenticatie en de mogelijkheid om beveiligde HTTPS verbindingen op te zetten. Hiernaast is het ontwerp in overeenstemming met de requirements. Het gedrag van de operations en de interface heb ik beschreven in de use-case beschrijvingen.

10.1.6 3.3 BOUWEN APPLICATIE

Beroepstaak 3.3 bouwen applicatie vraagt om verfijnen, bouwen en documenteren van systeemdelen. Deze systeemdelen worden samengesteld tot een werkende applicatie. Dit is op onderdelen in het project gehaald, te weten:

- Ontwikkelen van het POC volgens de requirements en het ontwerp zoals beschreven in het zevende hoofdstuk 'Iteratie 1# ontwikkelen van een proof of concept'.
- Ontwikkelen van de wijzigingsinterface volgens de requirements en het ontwerp zoals beschreven in het achtste hoofdstuk 'Iteratie 2# ontwikkelen van een wijzigingsinterface'.
- Ontwikkelen van de service operations volgens de requirements en het ontwerp zoals beschreven in het negende hoofdstuk 'Iteratie 3# Ontwikkelen van service operations voor XML uitwisseling'.
- Ontwikkelen van de generic en taak volgens de requirements en het ontwerp
- Het op de juiste manier gebruik maken van de bestaande module 'XML framework'.
- Het ontwikkelde systeem is getest en werkend opgeleverd zoals beschreven in de transition fase
- Het gebruik van de Pluriform ontwikkelomgeving zoals beschreven in het eerste hoofdstuk 'Bedrijfsorganisatie'.
- Slim hergebruik van methoden en generics uit Pluriform

Deze beroepstaak was vooraf niet opgenomen in het afstudeerplan omdat toen niet bekend was hoe groot de complexiteit van het ontwikkelproces in het afstudeerproject zou zijn. De beroepstaak is op niveau 3 behaald vanwege het zelfstandig ontwikkelen in een lastige context, met name een objectgeoriënteerde applicatie waarbij geavanceerde concepten van de gebruikte Pluriform programmeertaal aan de orde zijn gekomen. De service operations vormen een gestandaardiseerd set van XML berichten waarbij ik rekening gehouden heb met hergebruik en wijzigbaarheid en test mogelijkheden. Ik heb hergebruik gemaakt van bestaande methoden en generics uit de Pluriform programmeertaal en het systeem. Het gehele ontwikkelde systeem heb ik zelfstandig ontworpen en ontwikkelt waarbij ik nieuwe methoden gebruikt heb die passen binnen de programmeerstandaarden van Matthat Software B.V. Het systeem heb ik ontwikkelt in een unieke Pluriform ontwikkelomgeving en is niet alleen werkend afgeleverd aan Matthat Software B.V. maar voldoet ook aan de requirements en het functionele en technische ontwerp.

10.2 HET AFSTUDEERPROCES EVALUEREN

10.2.1 PLANNING

De planning was opgesteld aan het begin van de afstudeerperiode en net na de goedkeuring van het afstudeerplan. Ik heb in de planning de drie doorlopen iteraties nooit duidelijk naar voren laten komen maar de tijd die ik had ingepland hiervoor was voldoende. Ook heb ik bij het maken van de planning de UP fases achter elkaar ingepland, dus volgens de watervalmethode. Maar ik had juist voor UP gekozen om fases naast elkaar uit te kunnen voeren. De uitwerking van de planning is dan ook exact volgens de UP methode verlopen (zie ook paragraaf 4.1.1).
Om op planning te blijven moest ik bijvoorbeeld tijdens het klantonderzoek de respondenten blijven aansporen om te reageren op de vragenlijst.

10.2.2 UNIFIED PROCESS

Inception fase
In de eerste fase heb ik gewerkt aan het PVA. Dit document heb ik samen met de opdrachtgever beschreven bij Matthat Software B.V. Het PVA heeft me erg geholpen om het probleem en de opdracht beter te begrijpen. De 20 dagen die ik voor deze fase had ingepland heb ik nuttig gebruikt om ook met collega's in het bedrijf te communiceren en vragen te stellen over het project, te verdiepen in het probleem en onderzoek te verrichten naar de communicatieprotocollen en Pluriform als systeem en ontwikkelomgeving.
In de inception fase is het ook klantonderzoek uitgevoerd. Het klantonderzoek vond ik erg leuk om te doen omdat ik dan contact had met klanten van Matthat Software B.V. Het verloop van het klantonderzoek is uiteindelijk erg goed gegaan ondanks de tegenvallende respons in de eerste week. Voor de opzet en de uitvoering van het onderzoek heb ik collega's het onderzoek eerst laten invullen en toetsen. Dit leverde op verschillende plekken in de vragenlijst nog veranderingen op van vraag en onderwerp. De uitwerking van de vragenlijst door deze te versturen leverde uiteindelijk genoeg respons op en voldeed aan de eisen die ik stelde in het onderzoeksplan. De rapportage van het onderzoek heb ik uiteindelijk doorgegeven door het gehele bedrijf. Ik heb mijn conclusies en aanbevelingen hierin opgenomen om Matthat Software B.V. in te kunnen lichten en op de hoogte te houden over het onderzoeksonderwerp.
Ik concludeerde dat het onderwerp voor uitwisseling met de website erg leefde onder de respondenten, maar concludeerde ook dat de respondenten een wazig beeld hadden hierover. Binnen het klantonderzoek heb ik gesprekken gevoerd met klanten over het onderwerp om elkaar af te tasten en te peilen waar de interessegebieden lagen.

Als ik later het klantonderzoek opnieuw zou doen, zou ik zeker kiezen voor een langere doorlooptijd van het onderzoek. Klanten in de goede doelen branche hebben dit gewoon nodig en dit komt het onderzoek uiteindelijk alleen maar ten goede. De gesprekken met klanten mogen zeker niet achterwege blijven, omdat dit erg nuttig kan zijn om de wensen en eisen tot in detail boven tafel te krijgen.
Een ander leermoment voor mij is het deelnemen aan een gesprek met een klant. Ik had vaak nog moeite met uitleggen en vertellen wat er verteld moest worden. Omdat ik wist dat dit zo zou zijn heb ik besloten om samen met een collega deze gesprekken te voeren om zo van hem te kunnen leren.
Het verdiepen in de filosofie achter Pluriform was voor mij een belangrijk onderdeel om te begrijpen hoe en waarom het systeem was opgebouwd. Ik heb hiervoor een aantal gesprekken gevoerd met de directeur van Matthat Software B.V. waarin hij alles probeerde uit te leggen. Na elk gesprek heb ik alles op papier gezet en gevraagd om feedback en aanvullingen.

Elaboration fase
Door het doorlopen van de eerste iteratie waarin een POC werd ontwikkelt had ik veel kennis opgedaan die ik kon toepassen in de tweede en derde iteratie waarin de wijzigingsinterface en de service operations werden ontwikkelt. Als ik zonder een POC en dus zonder de eerste iteratie de wijzigingsinterface en de service operations was gaan ontwerpen had ik meer moeite gehad. Ik kon nu alles overzichtelijk en gestructureerd behandelen vanwege de voorkennis uit de eerste iteratie. Een voordeel van het ontwerp voor het POC was dat ik deze bijna volledig kon overnemen en gebruiken in de derde iteratie.

De tweede en derde iteratie in de elaboration fase verliep zonder tegenvallers. Het opstellen van de eisen en use-cases ging allemaal in samenwerking met klanten. Ik had het voordeel dat klanten niet gedetailleerd wisten wat men nodig had waardoor ik voorstellen kon aanleveren in de vorm van een architecture description. Door constant het voorstel te bespreken en te versterken kon ik zo duidelijke eisen en use-cases opstellen.

Het inzetten van UML tijdens de elaboration fase heeft mij erg geholpen om een goed ontwerp te maken. Ook naar collega's en klanten toe hebben de visuele ontwerpen geholpen om het ontwerp te begrijpen.

Construction fase
Dat de RPC technologie outdated verklaart werd aan het eind van de eerste iteratie had een voordeel en een nadeel. Het nadeel was dat er tijd verloren was gegaan bij het ontwikkelen van een POC met de RPC technologie. Een voordeel was dat de keuze tussen SOAP en RPC niet meer gemaakt hoefde te worden.

Op het moment dat ik begon met de derde iteratie voor het ontwikkelen van SOAP Service Operations werd bekend dat het bedrijf Pluriform Software de module XML framework, die nodig is om de Service Operations te ontwikkelen, had afgeschermd en afgesloten voor verdere ontwikkeling in de Pluriform ontwikkelversie van Matthat Software B.V. Het geheel leverde een vrij groot probleem op, aangezien het de werkzaamheden van het afstuderen volledig stil zou leggen. Het werd nog erger toen bekend werd dat Matthat Software B.V. pas eind april over zou stappen naar de meest recente versie van Pluriform software. Hiertoe heb ik besloten om bij Pluriform Software te vragen naar de laatste ontwikkelversie van Pluriform voor het inrichten van een XML framework.
Ik denk dat ik er goed aan gedaan heb om snel contact op te nemen met Pluriform Software anders zou ik of in tijdnood gekomen zijn of had ik het afstudeertraject moeten staken.

Ik zou het POC in het vervolg nog beter willen ontwerpen en ontwikkelen met het doel van het project in gedachte om zo de componenten uit de POC optimaal te kunnen hergebruiken in de derde iteratie.
Achteraf gezien had ik ontwikkelingen niet moeten maken in de testomgeving maar in de live ontwikkelomgeving van Matthat Software B.V. Het overzetten van de componenten heeft namelijk een dag gekost.

Het proces van ontwikkeling heb ik vrijwel zelfstandig uitgevoerd. Als ik vragen had kon ik terecht bij collega's die hier meestal dezelfde dag nog een antwoord op konden geven. Ik heb in vergelijking met stage vorig jaar nu erg gelet op de tijd die het ontwikkelen in beslag nam. Tijdens mijn stage was het zo dat ik vaak te lang bleef kauwen op bepaalde technische problemen. Met het afstuderen heb ik op zulke momenten geprobeerd om tijdig de hulp van collega's in te roepen om te zorgen dat ik niet in tijdnood zou komen.
Tijdens het ontwikkelen van het POC, de wijzigingsinterface en de service operations heb ik gemerkt dat de eisen niet erg gewijzigd zijn (op de eis van het inzetten van SOAP na) en dat geeft dus aan dat de eisen die opgesteld zijn in de elaboration fase een goede indicatie heeft gegeven van waar de componenten aan moeten voldoen.

De periode van de construction fase had ik ingepland op 20 dagen. Dit was inclusief documenteren. Ik had hier uiteindelijk een week langer voor nodig. Achteraf was het testen van de RPC service messages een aardige bottleneck vanwege het ontwikkelen van een .NET testtool hiervoor. Dit is ook weer een nadeel als ik eerder had geweten dat de RPC technologie outdated verklaart zou worden.

Transition fase
De allerlaatste fase in UP heb ik parallel met de construction fase behandeld. Door de soapUI testtool te gebruiken werd het testproces versoepeld.

Het overdragen van het XML framework is goed verlopen. Het framework is, klaar voor implementatie bij klanten, opgenomen in de live ontwikkelomgeving van Pluriform Goede Doelen. Hiernaast heb ik heb via e-mail aan alle collega's kenbaar gemaakt waar de documentatie te vinden is binnen Pluriform.

10.2.3 AFSPRAKEN EN OVERLEGMOMENTEN

Tijdens het afstudeertraject heb ik veel overlegmomenten gehad binnen het bedrijf. Vaak met de opdrachtgever, de manager en andere collega's. Ik heb deze momenten als zeer leerzaam en prettig ervaren. Ik heb daarin ook iedereen van mijn voortgang op de hoogte gehouden en hun input gevraagd indien dit nodig was. Aan het begin van het afstudeertraject had ik mijn grote twijfels over de begeleiding die verleent zou worden vanuit het bedrijf. Dit had te maken met de verdeelde bezetting op kantoor tijdens een werkweek. Maar ik heb geleerd dat door goed te communiceren en tijdig afspraken in te plannen collega's gemakkelijk in mijn behoefte naar begeleiding konden voorzien.

Een leermoment hierbij is dat als de juiste collega's aanwezig zijn hier goed gebruik van moet maken door de vragen die ik heb op te schrijven en te stellen.

Afspraken met klanten maken heeft mij doen beseffen dat hier moeilijk een planning voor te maken valt. Klanten hebben vaak weinig of geen tijd voor afspraken. Ik heb ook geleerd dat je niet altijd aankan op de dingen die klanten beloven. Om bijvoorbeeld de klant te nemen die mee zou gaan in het implementatietraject voor het systeem. In de eerste afspraak gaf men aan er binnen enkele weken er mee te willen starten, maar na een paar weken blijkt dat men het te druk heeft of het niet kan doen omdat het niet is opgenomen in de begroting van dit jaar.
Ik heb geleerd dat overleg met klanten ruim ingepland moet worden omdat de doorlooptijd weken kan duren.

10.2.4 MATTHAT SOFTWARE B.V. EN DE GOEDE DOELEN BRANCHE

Het afstuderen bij Matthat Software B.V. heb ik ervaren als erg leerzaam. Het bedrijf en de branche waarin het werkt spreekt mij dan ook erg aan. Binnen het bedrijf heerst een rustige sfeer en werken voornamelijk vriendelijke mensen die, wanneer mogelijk, tijd nemen om je te helpen. Doordat collega's een brede kennis hebben en vaak een functie vervullen die een mix vormt tussen consultant en programmeur kan vaak iedereen je helpen met een probleem.

De goede doelen branche spreekt mij heel erg aan. Enerzijds omdat je weet dat je enigszins kan bijdragen en ondersteunen aan goede doelen en anderzijds omdat je te maken hebt met klanten en vrijwilligers die allemaal iets willen betekenen voor het goede doel. Ik heb deze mensen ervaren als erg vriendelijk en meedenkend.

10.3 DE OPGELEVERDE PRODUCTEN EVALUEREN

10.3.1 ENQUETE EN ONDERZOEKSRAPPORT

De enquête die ik heb gehouden onder de klanten van Matthat Software B.V. heeft geleid tot antwoorden op de vragen die gesteld waren in het onderzoeksplan. Het doel om de wensen van klanten in kaart te brengen is hiermee behaald. Ik denk dat de belangstelling die de vragenlijst wekte onder de klanten als heel waardevol gezien kan worden. Ik zie het als een zijdelings effect dat de klanten op de hoogte zijn gesteld en wakker zijn geschud van de mogelijkheden die er zijn met betrekking tot uitwisseling van gegevens tussen website en Pluriform Goede Doelen. Het viel me wel tegen dat klanten niet iets gedetailleerder konden antwoorden op de vragen die gesteld werden. Dit was niet het doel van de vragenlijst, maar kon vervolg gesprekken wel wat sturen. Over het algemeen kan ik stellen dat de eisen en doelen die ik verlangde van het onderzoek zijn behaald.
Het onderzoekrapport waarin ik de resultaten presenteer van het onderzoek, conclusies trek en een aanbeveling doe heb ik gebruikt om de gesprekken met klanten te sturen. Het rapport heb ik verspreid onder de medewerkers van Matthat Software B.V. en dat leverde hier en daar vragen op over bijvoorbeeld welke klanten specifieke antwoorden hadden gegeven. Mede door de aanbeveling over het vormen van een visie over de rol van sociale media bij fondswervende organisaties heeft Matthat Software B.V. een bedrijf in de hand genomen om hier meer kennis over op te doen en een beter beeld te krijgen om zo klanten te informeren.

10.3.2 ARCHITECTURE DESCRIPTION

De architecture description bleek van grote waarde voor klanten. Het document heb ik gebruikt om voorstellen te doen over het systeem richting klanten. Ik wilde een document waarin klanten precies konden zien en begrijpen wat ze zouden krijgen in het systeem en hoe het ongeveer zou werken. Daarnaast moest het document mijzelf aan het nadenken zetten over alle aspecten die erin staan. Als ik het document beoordeel denk ik dat het er zeker aan voldoet. Ik heb ook gevraagd om feedback van klanten over het document. Men gaf aan door het document een helder beeld te hebben over het systeem. Ik heb altijd tegen klanten gezegd dat het een concept versie was en een levend document zal zijn. Dit is logisch aangezien in overleg met de klant er nog een heleboel aan toegevoegd zou kunnen worden.
Het document is inmiddels door een collega bij Matthat Software B.V. gebruikt om ook aan een klant voor te leggen.

10.3.3 POC, SERVICE OPERATIONS EN WIJZIGINGSINTERFACE

POC

Het doel van het POC was om kennis op te doen over de communicatieprotocollen waar ik in die fase nog uit moest kiezen. Met het POC heb ik aan Matthat Software B.V. laten zien waar het systeem toe in staat is en wat alle mogelijkheden zijn. Een goed en bijkomend voordeel was dat ik in de derde iteratie het POC kon hergebruiken en uiteindelijk kon opnemen in het systeem. De service message die ik ontwikkeld heb met de RPC technologie heeft uiteindelijk geen toegevoegde waarde gehad en was totaal onbruikbaar. Ik ben blij dat ik eerst een POC ontwikkeld heb in de eerste iteratie omdat het mij erg heeft geholpen om het uiteindelijke systeem te ontwikkelen.

Service operations en wijzigingsinterface

De service operations en de wijzigingsinterface heb ik helemaal kunnen opleveren. Ik heb alles gestelde eisen en wensen kunnen verwerken in het product. Het belangrijkste doel was om een gestandaardiseerde interface aan klanten te kunnen aanbieden. Met de componenten die ik heb ontwikkeld is dit gelukt. Het gehele systeem kan ingezet worden bij iedere klant, puur omdat de componenten voor alle klanten overeenkomstig zijn. Maar ik ben ervan overtuigd dat voor elke klant een stukje maatwerk er gewoon bij hoort omdat elke klant dingen wel net iets anders of extra zou willen.
In een vervolg op het project zou de wijzigingsinterface aangevuld kunnen worden met triggers en checks die de binnenkomende gegevens al controleren op verschillende regels om zo veel werk uit handen te nemen van de gebruiker van de wijzigingsinterface. Mocht het in de toekomst blijken dat meerdere klanten

overeenkomende wensen hebben die nog niet in het ontwikkelde systeem zitten dan kunnen hiervoor nieuwe gestandaardiseerde service operations worden ontwikkelt.
Helaas is een eerste implementatie bij een klant nog niet gerealiseerd. Matthat Software B.V. is nog in overleg met een aantal klanten hierover. Ik ben van mening dat als er parallel aan het afstuderen ook een implementatie traject bij een klant geweest zou zijn, de service operations en interface hier en daar nog aangepast zijn. Dit komt omdat je dan één op één met de klant kan praten over de implementatie.

Conclusie
Aan de hand van feedback van collega's en klanten van Matthat Software B.V. mag ik concluderen dat de documenten over het algemeen inhoudelijk prettig leesbaar zijn, kwalitatief goed zijn en duidelijkheid schept.

LITERATUURLIJST

BOEKEN

Kruchten, Philippe. 2004. *The Rational Unified Process, an introduction.* 3e druk. 2004.
Rozanski, Nick & Woods, Eoin. 2005. *Software Systems Architecture.* sl : Pearson Education, 2005.
Verhoeven, Nel. 2010. *Wat is onderzoek? Praktijkboek methoden en technieken voor het hoger onderwijs.* 3e druk. Den Haag : Boom onderwijs, 2010.
Warmer, Jos & Kleppe, Anneke. 2006. *Praktisch UML.* 2006. Vol. 3e editie.
Martin Pol, Ruud Teunissen en Erik van Veenendaal. 2002. *Testen volgens TMap.* 2e druk. sl : Uitgeverij Tutein Nolthenius, 2002.
Nicole de Swart. *Handboek Requirements*

WEBSITES

NetQ. 2011. Online enquete maken. *NetQ.* [Online] Februari 2011. http://netq.nl/.
RUP op maat. [Online] http://www.rupopmaat.nl/.
soapUI. [Online] http://www.soapui.org/.
XML-RPC.NET. [Online] http://www.xml-rpc.net/.
W3C. technology standards. [Online] http://www.w3.org/TR/tr-technology-stds.

BEGRIPPENLIJST

Actor	Iets of iemand die een handeling verricht en daarmee een proces beïnvloedt
adresXpress	Een betaalde webservice van Cendris voor het ophalen van adressen uit een gigantische bibliotheek
Branchemodel	Een verzameling modellen die samen de processen in de betreffende branche ondersteunen
Dynamic-Link Library (DLL)	een bibliotheek met functies, die door meerdere applicaties gebruikt kunnen worden
Enterprise Resource Planning (ERP)	Een softwareprogramma ter ondersteuning van alle processen binnen een bedrijf
Generic	Methode zonder gespecificeerd datatype
Graphical User Interface (GUI)	Een interface die gebruikers in staat stelt om interactie te hebben met een systeem door middel van grafische elementen in plaats van alleen tekst commando's
Internet Information Services (IIS)	Een verzameling serverdiensten voor het internet bedoeld voor Windows machines
Niet-Gouvernementele Organisatie (NGO)	NGO's zijn niet-gouvernementele organisaties, onafhankelijk van de overheid en richten zich op een maatschappelijk belang. Over het algemeen gaat het om organisaties die werken aan het bevorderen van milieubescherming, gezondheid, ontwikkelingswerk of het bevorderen van de mensenrechten
Pointer	Een verwijzing naar een ander object in een systeem
Proof of Concept (POC)	Basisimplementatie om aan te tonen dat de voorgestelde oplossing in de praktijk te gebruiken is
Remote Procedure Call (RPC)	Een technologie die een computerprogramma op één bepaalde computer toestaat om code uit te voeren op een andere machine zonder dat de programmeur de code expliciet hiervoor geschreven heeft.
Response	Het XML bericht die een client van een server ontvangt nadat hij eerst een request gedaan heeft
Request	Een XML bericht van een client naar een server waarin een specifieke vraag om gegevens gesteld wordt
Simple Object Access Protocol (SOAP)	Een computerprotocol dat wordt gebruikt voor communicatie tussen verschillende componenten. Werd later ook Service Oriënted Architecture Protocol genoemd.
Survey onderzoek	Een methode voor het verzamelen van informatie over items in een populatie
Unified Modeling Language (UML)	Een modelmatige taal om objectgeoriënteerde analyses en ontwerpen voor een informatiesysteem te kunnen maken
Unified Process (UP)	Een iteratief softwareontwikkelingsproces
Web Service	Een interface van een applicatiecomponent die toegankelijk is via standaard webprotocollen en waarbij wordt gecommuniceerd via XML zonder menselijke tussenkomst
Web Service Description Language (WSDL)	Een XML-taal waarmee de interfaces van webservices kunnen worden beschreven
eXtension Markup Language (XML)	Een standaard voor de syntaxis van formele markuptalen waarmee men gestructureerde gegevens kan weergeven in de vorm van platte tekst

BIJLAGEN

De bijlagen zijn opgenomen in het tweede document genaamd Afstudeerverslag: de bijlagen.