Automatische koppeling formulieren aan webservices

Naam: Twan van Buuren
Studentnummer: 1604889
Opleiding: Opleiding informatica
Opdrachtgever: Verne
Eerste examinator: Linda Terlouw
Tweede examinator: Rory Sie

1 Voorwoord

Verne beheert een online Software as a Service (SaaS) platform voor verzekeringsadministratie. Zij stelt zich tot doel de volledige levenscyclus van verzekeringen zo veel mogelijk te automatiseren. Bijna een jaar geleden heeft Verne de FormsEngine geïntroduceerd. Deze omgeving wordt door partner MyCubes ontwikkeld.

In de FormsEngine is het mogelijk om formulieren te laten definiëren door mensen met de domeinkennis van verzekeringen. Deze kunnen vervolgens in verschillende websites geïntegreerd worden. Daarbij kan een integrerende partij de layout en styling volledig bepalen zonder aan de inhoud van de vragen in een formulier te komen.

Om de gegevens uit zo'n formulier in het verzekeringsplatform van Verne te krijgen moet er nog een koppeling gemaakt worden aan de webservices van Verne. Bij het koppelen van zo'n formulier aan de webservices worden echter nog veel handmatige werkzaamheden door een programmeur verricht. Naast de grote foutgevoeligheid van deze werkzaamheden, kostten ze ook veel tijd.

Bij het inrichten van mijn afstudeerproject, wist ik direct dat ik een praktijkopdracht wilde uitvoeren. Het verbeteren van de workflow bij het inrichten van formulieren voor nieuwe klanten door te kijken hoe de koppeling geautomatiseerd kan worden ingericht was hiervoor zeer geschikt.

Dit was enerzijds inhoudelijk een interessante opdracht, onder andere omdat het verschillende nieuwe technologieën combineert en een flexibele oplossing vraagt. Anderzijds vind ik het een belangrijk onderdeel van software ontwikkeling om handmatig werk zo veel mogelijk te automatiseren.

In dit project heb ik eerst gekeken naar de mogelijkheden die bestaande tools bieden. Hieruit bleek duidelijk de noodzaak om zelf een applicatie te ontwikkelen. Vooral omdat er geen pakket beschikbaar was dat de gewenste functionaliteit bood. In een iteratief proces heb ik een tool ontwikkeld die een spectaculaire tijdswinst oplevert en de programmeurs herhaalwerk uit handen neemt.

2 Samenvatting

Verne beheert en ontwikkelt een online verzekeringsadministratieplatform. Zij stelt zich tot doel de volledige levenscyclus van verzekeringen zo veel mogelijk te automatiseren. Daarbij hebben ze te maken met verschillende klant-omgevingen van verzekeraars waarin producten aangevraagd en gewijzigd moeten worden.

Dit kan via de SOAP services die Verne beschikbaar stelt. Echter is gebleken dat integratie van deze services vaak niet soepel verloopt, omdat integratiepartijen vaak niet de verzekeringskennis hebben om web-formulieren op te stellen die de juiste gegevens uitvragen. Om dit te ondervangen biedt Verne web-formulieren aan via de FormsEngine. In deze omgeving kunnen web-formulieren worden aangemaakt in een CMS door mensen met de domeinkennis. Deze formulieren worden gepubliceerd via een Publisher component. Deze maakt het formulier beschikbaar om te integreren in een website naar keuze. Daarbij kan de layout en styling van het formulier volledig aangepast en beheerd worden door de partij die de website beheert, zonder dat die de gestelde vragen kunnen of moeten aan passen.

De gegevens die uit deze formulieren worden verstuurd zijn echter in een ander formaat (JSON) dan de webservices van het Verne-platform (SOAP). De DataIntegrator lost dit probleem op door de gegevens te vertalen volgens een ingevoerde mapping. Deze mapping wordt door een ontwikkelaar bij Verne gemaakt en ingevoerd. Dit mapping proces is een bottleneck bij het invoeren van nieuwe formulieren. In dit project wordt onderzocht hoe dit geautomatiseerd kan worden.

Uit een inventarisatie van de beschikbare mapping tools is gebleken dat er geen software beschikbaar is die de belangrijkste functionaliteit bevat: ***Automatisch de mapping genereren voor nieuwe bron en doel datastructuren.*** Daarom is gekozen om zo'n tool zelf te ontwikkelen, de DataMapper.

Tabel 1 (pagina 4) bevat de daarvoor te bouwen functionaliteiten, waarbij ook is aangegeven of ze binnen de scope (tijd en middelen) van dit project vallen. Er is gekozen om de applicatie te ontwikkelen als plugin die zelfstandig functioneert, zodat deze op zichzelf ontwikkeld en gebruikt kan worden maar ook kan worden opgenomen in een van de applicaties in de FormsEngine. De huidige architectuur van de FormsEngine is er op gericht dit soort uitbreidingen te faciliteren.

De eerste versie van de DataMapper die ontwikkeld is heeft bewezen dat de belangrijkste functionaliteiten mogelijk zijn. Echter was het beheren en debuggen van de zogenaamde rules die de gegenereerde mapping bepalen niet zoals gewenst. Om dit te ondervangen is een tweede versie ontwikkeld waarin de structuur is verbeterd door het toepassen van een combinatie van de composite en observer patterns.

Uit metingen blijkt dat bij gebruik van deze tool, afhankelijk van de (nieuwe) complexiteit van de koppelingen, **op jaarbasis 157 tot 325 uur bespaard wordt op het maken van koppelingen**. Deze besparing kan nog verder uitgebreid worden door de gebouwde applicatie verder te ontwikkelen.

Het advies is daarom om de tool op te nemen in de processen bij Verne. Daarbij dient de tool ook verder ontwikkeld te worden voor een nog grotere optimalisatie van de werkzaamheden. Daarbij kunnen in eerste instantie de functionaliteiten uit Tabel 1 (pagina 4) die buiten scope van dit project

vallen als uitgangspunt dienen. Ten slotte is onderzoek aan te bevelen hoe de tool breder kan worden ingezet, denk daarbij bijvoorbeeld aan het (deels) genereren van testscripts.

Tabel 1 Featurelijst van te bouwen DataMapper geschaald volgens MoSCoW methode

Functie categorie	Functionaliteit	Beschrijving	MoSCoW ranking	In scope afstuderen
Schema ondersteuning	Ondersteunen van Soap Webservices	Parsen van WSDL	MUST	Ja
	Ondersteunen van formulier structuur	Parsen van JSON (forms structuur)	MUST	Ja
	Ondersteunen REST Services	o.a. Parsen van JSON schema (http://json-schema.org/)	SHOULD	Nee
Aanmaken van data mapping rules	Framework voor het schrijven van data mapping rules	De gebruiker heeft de mogelijkheid regels in te voeren en te bewaren	MUST	Ja
	Schrijven van rules in Groovy taal	De ingevoerde regels kunnen als Groovy (script) worden uitgevoerd	MUST	Ja
	Ondersteuning andere programeertalen	De ingevoerde regels kunnen als een andere programmeer/script taal worden uitgevoerd	COULD	Nee
	Ondersteunen van zowel request als response berichten	De vertaling moet twee kanten op gegenereerd worden: het versturen en ook het antwoord van de server moet weer terugvertaald worden naar het formulier.	MUST	Ja
	Ondersteunen van herbruikbare rule sets voor verschillende formulieren	De regels moeten hergebruikt kunnen worden	SHOULD	Ja
	Ondersteunen van versie beheer van mapping rules	Verschillende versies van een mapping rule kunnen worden aangemaakt en bewaard	SHOULD	Nee
Rule engine	DataMapping rules toe passen en de templates genereren	De rules worden in de betreffende programmeertaal uitgevoerd om een resultaat te genereren	MUST	Ja
	Ondersteunen van zowel request als response berichten	De vertaling moet twee kanten op gegenereerd worden: het versturen en ook het antwoord van de server moet weer terugvertaald worden naar het formulier.	MUST	Ja
	Ondersteunen van versie beheer van gegenereerde templates	Verschillende versies van een mapping rule kunnen worden aangemaakt en bewaard	SHOULD	Nee
Integratie	Plugin op de FormsEngine Publisher	De DataMapper kan aan de Publisher worden toegevoegd	MUST	Ja
	Data integratie met DataIntegrator	Direct uploaden van template naar de juiste app	MUST	Nee

	Override van gegenereerde templates voor specifieke formulieren	De gegenereerde templates moeten handmatig kunnen worden aangepast	SHOULD	Nee
	Ondersteuning voor omgevingsvariabelen van DataIntegrator	De in DataIntegrator bestaande variabelen kunnen worden gebruikt in de rules	COULD	Nee
GUI	Visualisatie van de data mappings	de data mapping heeft een visuele representatie die de mapping verduidelijkt	WON'T	Nee
	Data mappings maken middels grafische mappings	De mapping rules kunnen worden aangemaakt middels een visuele representatie die de werking verduidelijkt.	WON'T	Nee

Inhoud

3 Inleiding

IT omgevingen worden steeds complexer, met steeds meer uiteen lopende componenten. Tegelijk verwacht de eindgebruiker een vergaande integratie van verschillende componenten en omgevingen. Een verzekeringsnemer wil bijvoorbeeld niet in de ene omgeving inloggen om zijn rekeningen in te zien, in een andere omgeving om zijn persoonlijke gegevens of dekkingen bij te werken en weer in een andere om een schade te melden. Dit betekent dat alle verschillende componenten en hun databronnen moeten kunnen communiceren. Dit gebeurt vaak via (web-)services.

Verne ontwikkelt een online verzekeringsadministratieplatform. Verne stelt zich tot doel de volledige levenscyclus van verzekeringen zo veel mogelijk te automatiseren. Daarbij hebben ze te maken met verschillende klant-omgevingen van verzekeraars die producten, die onder andere binnen het Verne platform worden geadministreerd, moeten kunnen verwerken.

Dit kan via de SOAP services die Verne beschikbaar stelt. Daarnaast bied Verne standaard web-formulieren aan, waarin gegevens kunnen worden bijgewerkt. Het handmatig koppelen van al deze verschillende omgevingen en services neemt steeds meer tijd in beslag. Hiervoor wordt de DataIntegrator gebruikt. Men ondervindt dat het configureren/mappen van dit component te veel tijd in beslag neemt en foutgevoelig is.

Het doel van dit onderzoek is een methode te vinden om het configureren/ mappen van dit component minder tijd te laten kosten. Een aspect daarvan zou ook kunnen zijn om de foutgevoeligheid terug te dringen, want fouten moeten weer worden opgelost en dit kost tijd.

Dit document beschrijft verder eerst het theoretisch kader. Het hoofdstuk “Probleemstelling opdrachtgever Verne” behandelt wat voor bedrijf Verne is, wat hun vraagstelling is en hoe de projectorganisatie is geregeld. Vervolgens wordt uiteengezet hoe dit project als afstudeeropdracht wordt aangepakt, beginnend met de vraagstelling, gevolgd door de methodieken en technieken. De resultaten zijn opgedeeld in twee secties, de eerste beschrijft de ontwikkelde software, de tweede de resultaten van de uitgevoerde metingen. De beantwoording van de deelvragen wordt in de conclusie samengevat en gebruikt om een antwoord op de hoofdvraag te geven. Op basis van deze uitkomst worden tot slot aanbevelingen gedaan hoe Verne kan profiteren van de uitkomsten van dit onderzoek.

4 Theoretisch kader

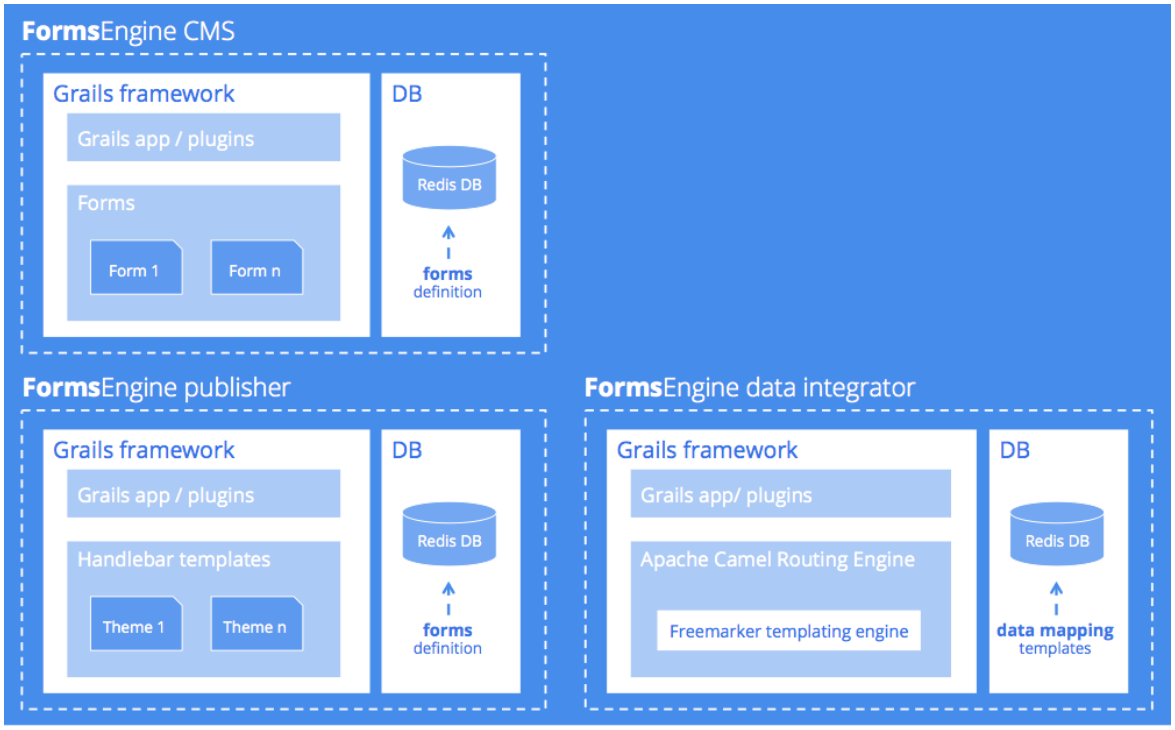
In deze sectie zullen de technieken en theorieën die ten grondslag liggen aan de uitvoering van het project worden uiteengezet.

4.1 Ontwikkelframework en technologieën

4.1.1 Huidige applicatielandschap & frameworks

Bij dit project wordt een component ontwikkeld binnen het bestaande applicatielandschap van de FormsEngine. De bestaande architectuur is dus het uitgangspunt voor het ontwikkelen van de beoogde architectuur en applicatie binnen dit project. Dit hoofdstuk beschrijft het huidige applicatielandschap en de technologieën voor dit project.

Figuur 1 geeft een overzicht van de componenten en de gebruikte technologieën. In de volgende sub-hoofdstukken zullen de gebruikte technieken verder worden toegelicht.



Figuur 1 Schematische weergave van huidige gebruikte applicatieframeworks

4.1.2 Grails

De applicatie zal als plugin in het Grails framework worden ontwikkeld.

Grails is een op Spring gebaseerd webapplicatieframework voor het Java platform. Het is gericht op het verbeteren van de productiviteit van ontwikkelaars door gebruik te maken van conventies en 'sensible defaults'. (grails.org, 2015)

Binnen het Grails framework wordt hoofdzakelijk in Groovy geprogrammeerd. Het is daarnaast mogelijk met Java te programmeren of Java componenten te gebruiken. (grails.org, 2015)

4.1.3 Groovy

Groovy is de belangrijkste programmeertaal in het Grails framework.

“Groovy is a powerful, optionally typed and dynamic language, with static-typing and static compilation capabilities, for the Java platform aimed at multiplying developers’ productivity thanks to a concise, familiar and easy to learn syntax. It integrates smoothly with any Java program, and immediately delivers to your application powerful features, including scripting capabilities, Domain-Specific Language authoring, runtime and compile-time meta-programming and functional programming.” (The Groovy programming language, 2015)

Hoewel Groovy op Java gebaseerd is en Groovy classes op het niveau van de JVM compatible zijn met Java classes (The Groovy programming language, 2015), is een belangrijk verschil dat Groovy niet statisch getyped is. Zo worden de uitgevoerde methodes runtime gekozen gebaseerd op de types van de argumenten en niet tijdens de compilatie, zoals bij Java.

4.1.4 Redis

Binnen de FormsEngine wordt Redis gebruikt als database. Redis is een in-memory key-value store, met persistentie op schijf. Dat wil zeggen dat gegevens uit Redis via sleutelwoorden kunnen worden opgevraagd. Ze worden dan opgehaald uit het werkgeheugen van de computer. Dit geheugen is snel maar wordt geleegd bij uitschakelen of opnieuw opstarten. Om te voorkomen dat gegevens verloren gaan als de computer wordt uitgeschakeld, maakt Redis regelmatig en bij uitschakelen backups op schijf. Deze opslag is langzamer, maar de gegevens blijven bewaard bij uitschakelen of opnieuw opstarten.

Het belangrijkste verschil met de gebruikelijke SQL/relationele databases is dat er geen ingebouwde voorzieningen zijn om relaties vast te leggen. Daar tegenover staat een snelle performance en ingebouwde datastructuren.

4.1.5 Freemarker template engine

De te genereren transformatie templates zijn Freemarker templates. Freemarker is een "template engine": een generieke tool om tekst te genereren, van html tot broncode of zoals in ons geval XML of JSON.

Freemarker is een Java component, bedoeld om door ontwikkelaars in hun applicatie te gebruiken (freemarker, 2015). De DataIntegrator gebruikt dit component om het resultaat van de vertaling van de service calls te kunnen definiëren. Daarbij is de binnenkomende call als objectstructuur beschikbaar en kan aan de hand daarvan het resultaat worden bepaald.

4.2 Inventarisatie van bestaande data mapping tools

Bij het zoeken naar een oplossing is gekeken naar wat voor tools er bestaan die (een deel van) de gewenste functionaliteit bieden.

Daarbij is gekeken naar de volgende opties:

- Altova
- Stylus studio
- Jamper
- Excel

Deze lijst is tot stand gekomen na een eerste inventarisatie van de mogelijkheden via zoekmachines en gesprekken met betrokkenen.

Excel is een beetje een vreemde eend in de bijt. Andere tools zijn bedoeld speciaal voor datatransformaties, Excel is een spreadsheet programma. Goede integratie met XML maakt dat Excel ook gebruikt kan worden om gegevens uit een Xml-bestand te verwerken en om te zetten.

Voor al deze tools blijkt dat ze gericht zijn op het voor de gebruiker vereenvoudigen van het definiëren van een transformatie, waarbij de bron- en doelstructuur vast liggen. Hoewel ze dit op zeer gebruiksvriendelijke wijze doen, is dit niet de functionaliteit die Verne nodig heeft, aangezien de wens juist is om voor nieuwe bron- en doel-datastructuren automatisch de transformatie te genereren (van Buuren, 2015).

Er zijn echter geen tools beschikbaar die deze functionaliteit bieden.

Wel kunnen sommige features van deze tools ook voor de tool die Verne wil gebruiken van toepassing zijn, daar ze wel in hetzelfde domein werken. In de volgende tabel zijn de features die relevant kunnen zijn opgenomen.

Tabel 2 Feature overzicht van beschikbare data mapping tools

Feature	Altova	Stylus studio	Jamper	Excel
visuele weergave en aanmaken mapping	Ja	Ja	Ja	Nee
operaties en vergelijkingen zonder code	Ja	Nee	Nee	Nee
zelf complexe transformatie in code definiëren	Ja	Ja	Ja	Ja
Nieuwe transformatie genereren	Nee	Nee	Nee	Nee

5 Probleemstelling opdrachtgever Verne

5.1 Verne als organisatie

Verne ontwikkelt en levert een backoffice (SaaS) platform voor verzekeringsmaatschappijen. Dit platform is volledig op webservices gebaseerd en biedt eindklanten via een portaal toegang tot een “mijn” omgeving. Backoffice medewerkers kunnen via hetzelfde webportaal, of middels een uitgebreidere .NET smart cliënt, de klantportfolio’s beheren.

Via webservices kunnen verzekeraars en externe ontwikkelteams functionaliteit van het platform in hun eigen omgeving integreren. Zo kan men bijvoorbeeld bij het boeken van een reis d.m.v. één extra vinkje een reisverzekering aanvragen via een externe site, of een eigen “mijn omgeving” integreren binnen een bestaand portaal.

Verne heeft als doelstelling om de verzekeringsbranche zo ver mogelijk te automatiseren. Een onderdeel van het platform doet elke dag 160.000 aanvragen in 26 landen per dag, inclusief de financiële verwerking hiervan in alle verschillende landen. Onlangs heeft Verne haar aandachtsgebied binnen de financiële diensten verbreed door de samenwerking met het Crowdfunding platform Symbid.

Verne is een kleine organisatie met 10 medewerkers. Het bedrijf kent een platte organisatiestructuur die past bij het kleine aantal werknemers. De betrokkenen bij dit project zijn allen goed aanspreekbaar via e-mail, Skype en telefoon en tijdens kantoor uren over het algemeen ook in persoon. Degene die het meeste met de oplossing zal moeten werken deelt het kantoor met de uitvoerder van dit project (Twan van Buuren). De directe betrokkenen en hun rollen zullen worden uiteengezet in het hoofdstuk “Projectorganisatie & begeleiding binnen Verne”.

5.2 Probleemstelling Verne

5.2.1 Inleiding

Onlangs heeft Verne een nieuw product geïntroduceerd samen met een externe ontwikkelpartij (MyCubes). Het product genaamd: “FormsEngine”, heeft als doel de integratie van externe partijen met het Verne platform te versimpelen waardoor activiteiten als premieberekeningen, aanvragen en mutaties door externe partijen sterk vereenvoudigd worden.

FormsEngine als product is tot stand gekomen uit de probleemstelling dat externe partijen moeite hebben met het koppelen aan de externe webservices (API) van Verne. Er is vaak een gebrek aan specifieke domeinkennis van verzekeringen bij externe ontwikkelpartijen waardoor de doorlooptijd en kosten hoog zijn bij een implementatie. Tevens is de “leercurve” van de Verne API dermate hoog dat Verne veel support moet leveren aan partijen die aansluiten.

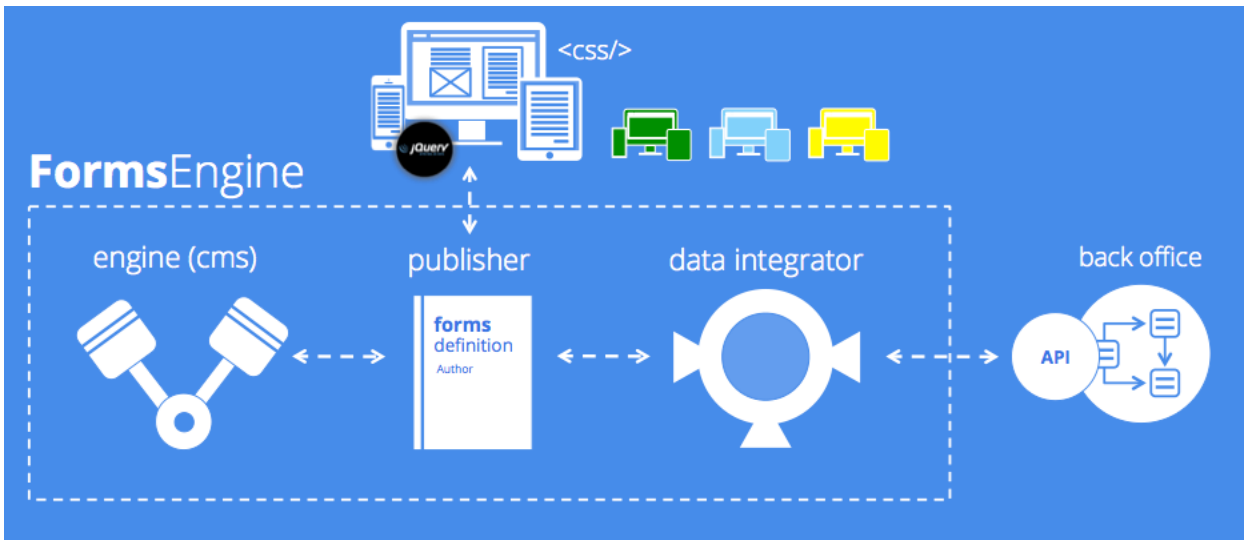
Met de introductie van ‘Responsive design’, A/B testen, Google Analytics en de wens van elke verzekeraar om een eigen “styling” te kunnen aanhouden volstaat een i-Frame oplossing niet. Maatwerkontwikkeling door externe partijen zoals hiervoor aangegeven levert een te lange doorontwikkeling op en vergt te veel ondersteuning en uitleg door Verne. De FormsEngine wordt ingezet om dit op te lossen.

5.2.2 FormsEngine componenten & toelichting

De FormsEngine wordt ingezet om het Verne verzekeringsplatform met websites en web-portalen van derden te integreren. De FormsEngine bestaat uit de volgende componenten:

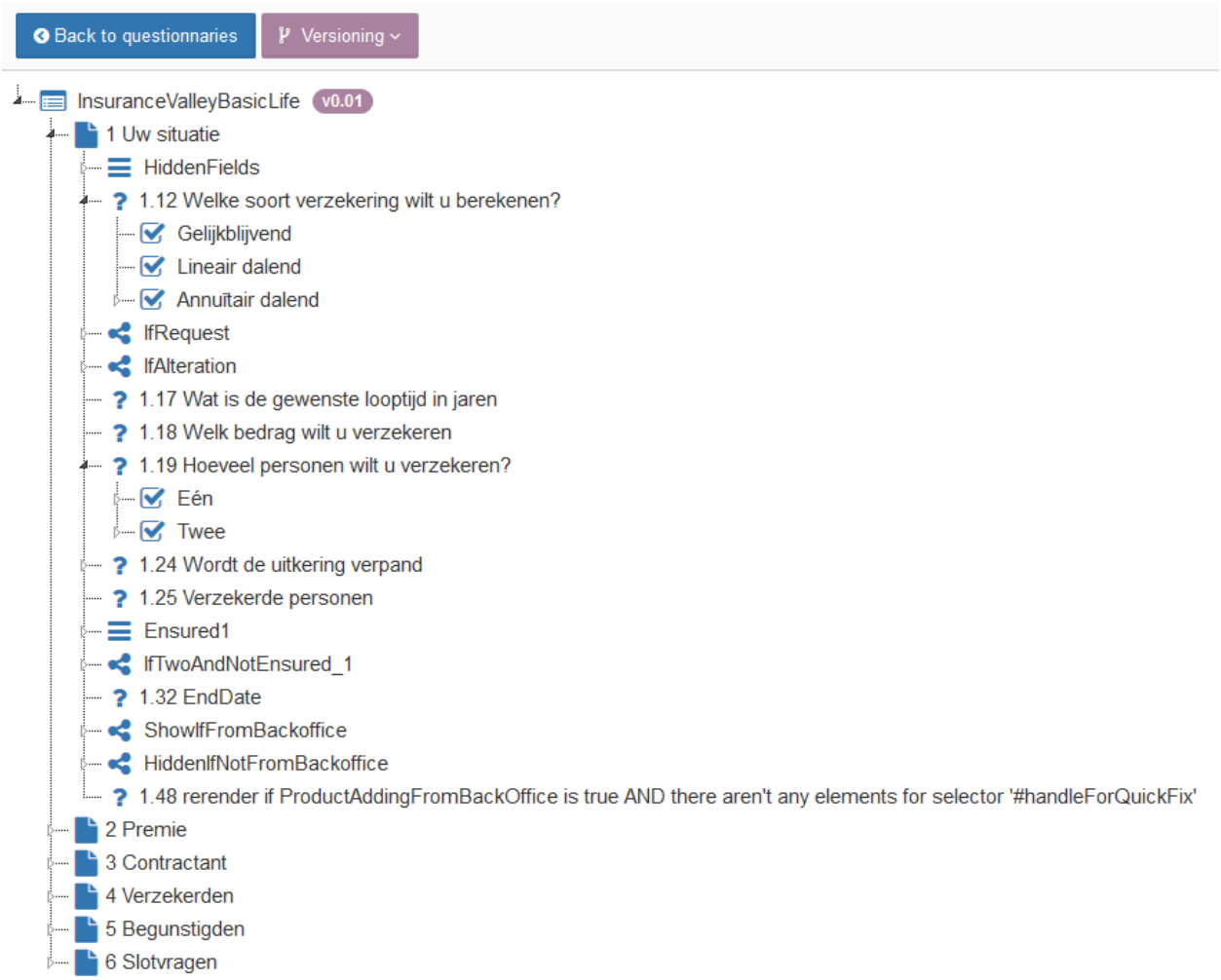
1) Engine (CMS)
2) Publisher
3) DataIntegrator

In Figuur 2 Schematische weergave van bestaande FormsEngine componenten zijn de componenten schematisch weergegeven.



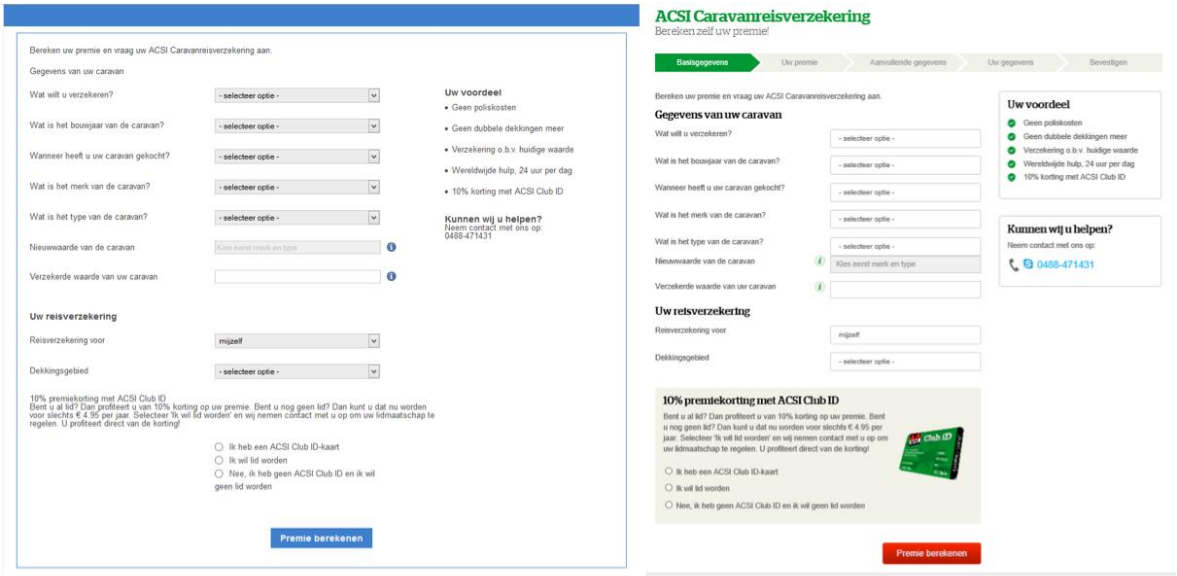
Figuur 2 Schematische weergave van bestaande FormsEngine componenten

De Engine is een web based CMS voor het opstellen en beheren van formulieren. Hierin stel je hiërarchisch de vraag-antwoord combinaties op. Dit wordt ook wel het opstellen van de “form definitie” genoemd. De form definitie bevat de volgorde van vragen, antwoorden, validaties, standaard antwoorden, type vragen (radiobuttons, checkboxes etc.). In Figuur 3 Screenshot van formulier structuur/definitie in de FormsEngine CMSis te zien hoe dit er uit ziet. Deze formulierdefinitie wordt in JSON-formaat opgeslagen. Dit is een zowel voor computers als mensen leesbaar formaat om gegevens te bewaren of over te dragen en wordt vaak gezien als een alternatief voor XML. Wanneer men klaar is met het opstellen van de formulierdefinitie in de Engine kan het formulier “gepubliceerd” worden naar een Publisher.



Figuur 3 Screenshot van formulier structuur/definitie in de FormsEngine CMS

Een Publisher stelt een formulier beschikbaar voor gebruik en levert daarbij het format en javascript om het formulier naar HTML om te zetten in de browser van de gebruiker. De Publisher(s) host(en) de formulier definities die vervolgens door een externe integratiepartij gebruikt worden. Een integratiepartij moet voor het gebruiken van het formulier een simpel script opnemen op haar site met een verwijzing naar de Publisher URL en het FormsEngine javascript. Dit javascript, welke op de Publisher staat, vertaalt de formulierdefinitie naar HTML middels het gebruik van zogenaamde "Handlebar templates" (het voornoemde format om het formulier naar html om te zetten), die elke integratiepartij naar wens kan aanpassen. Deze templates bepalen welke HTML elementen voor de formulier elementen worden getoond. Zo kan hetzelfde formulier er op twee manieren uitzien, zoals te zien in onderstaande screenshots van hetzelfde formulier met een andere template set.



Figuur 4 Screenshots van zelfde formulier met twee verschillende (styling) thema's

Doordat het javascript van de Publisher de vertaling naar HTML (incl. classes, element-id's etc.) maakt, kan een externe ontwikkelaar dus zelf de styling bepalen van het formulier terwijl de mensen met domeinkennis zelf makkelijk het formulier gecentraliseerd kunnen beheren, aanpassen en publiceren zonder tussenkomst van een externe ontwikkelaar of ontwikkelpartij als de eerste integratie met een formulier voltooid is. Dit scheelt enorm in doorlooptijd, support en kosten.

Bij het versturen van het formulier komen de vraag en antwoordcombinaties in JSON-formaat terug naar de Publisher (ook het versturen van het formulier wordt middels het javascript afgevangen). Aangezien dit JSON-formaat niet door de Verne API of andere API's begrepen wordt, is er een vertaling nodig naar andere API's. Deze vertaalslag gebeurt in de DataIntegrator.

De DataIntegrator vertaalt in het geval van Verne het JSON formaat naar SOAP webservices. Hiervoor wordt gebruik gemaakt van (Freemarker) templates.

5.2.3 Communicatie tussen FormsEngine componenten

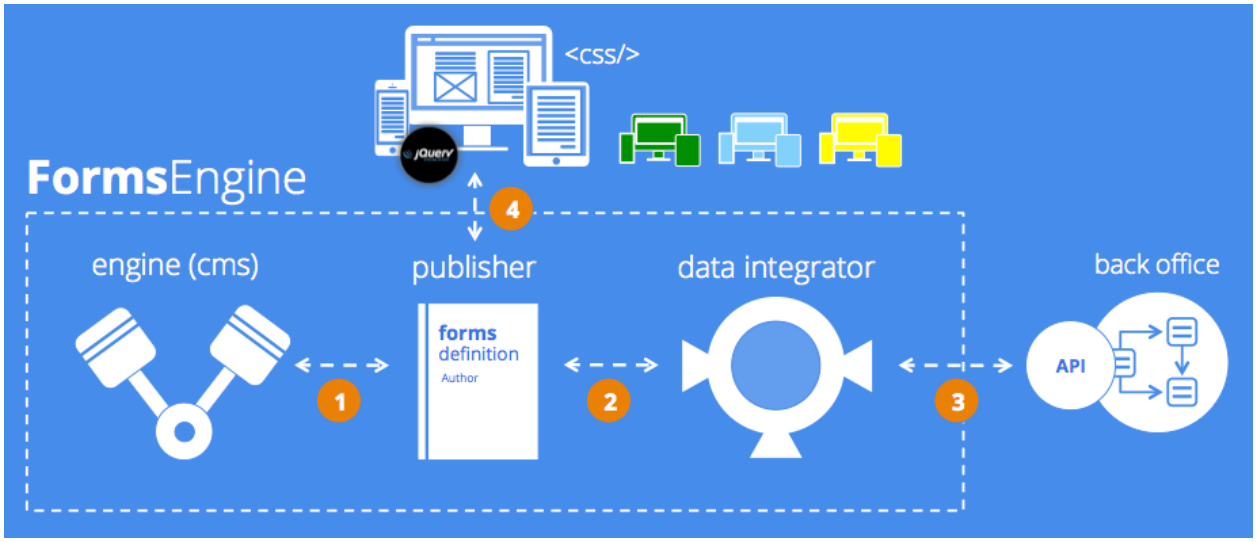
In dit hoofdstuk zal de communicatie tussen de verschillende componenten van de FormsEngine worden uiteengezet aan de hand van Figuur 5. In de bijlagen is dit uitgewerkt in UML sequence diagrams.

De communicatie tussen de componenten wordt in de afbeelding weergegeven met de genummerde pijlen in Figuur 5:

1. Op het moment dat (een versie van) een formulier af is kan de gebruiker deze publiceren op een geselecteerde Publisher. Bij het publiceren wordt de definitie van het formulier op de Publisher geplaatst. Deze maakt het formulier als HTML beschikbaar.
2. De invoer op het formulier wordt vanuit de Publisher doorgestuurd naar de DataIntegrator. De response uit de DataIntegrator wordt weer terug gestuurd naar de Publisher.
3. In DataIntegrator worden de gegevens omgezet naar een webservice call. Deze call wordt uitgevoerd en de response wordt ook weer via Freemarker templates terug vertaald naar het formaat voor de Publisher. Dit wordt weer naar de Publisher teruggestuurd (2) en in het

formulier dat de eindgebruiker ziet verwerkt.

4. Een eindgebruiker bekijkt een formulier dat in een site is opgenomen. De webpagina haalt de formulierdefinitie en opbouw op van de Publisher en stuurt de ingevulde resultaten terug. De response van stappen 2 en 3 wordt aan het formulier terug gegeven.



Figuur 5 Schematische weergave van communicatie tussen FormsEngine componenten

5.2.4 Probleem

De vertaalslag in de DataIntegrator naar de Verne API vergt op dit moment nog altijd werk van ontwikkelaars om de juiste velden te “mappen”. Aangezien er veel formulieren zijn, komt het maken van mappings vaak voor, zeker daar waar er binnen de FormsEngine gebruik gemaakt wordt van “bibliotheek”-elementen (standaard vragenset die je kunt hergebruiken binnen formulieren). Het maken van data mappings is dus op dit moment voor Verne een bottleneck in het maken van volledig werkende formulieren (met integraties naar de Verne services). De werkzaamheden daarbij bestaan voor een groot deel uit herhaling en zijn tevens foutgevoelig.

5.3 Doelstelling opdrachtgever

Door het grotendeels automatiseren van data mappings wil Verne de kwaliteit verhogen van opgeleverde formulieren en voornamelijk de tijd om tot een volledig werkend en geïntegreerd formulier te komen, sterk verkorten.

Binnen de scope van de afstudeeropdracht is voor de opdrachtgever de doelstelling dat de basis wordt gelegd wat betreft architectuur en ontwikkeling om data mappings te kunnen automatiseren. Belangrijk hierin is wel dat de ontwikkelde software intellectueel eigendom is van Verne gezien de software onder licentie aan partners en derde partijen wordt gedistribueerd.

5.4 Projectorganisatie & begeleiding binnen Verne

Bij de verschillende projecten wordt bij Verne veelal gewerkt volgens de scrum methodiek. Bij deze methodiek wordt op iteratieve wijze aan een product gewerkt. Binnen deze methodiek zijn verschillende rollen gedefinieerd. Daarvan is de “product owner” eindverantwoordelijke voor (het bepalen van) de functionaliteiten.

Hary Selman is “product owner” van FormsEngine binnen Verne en verantwoordelijk voor de begeleiding en aansturing van zowel het afstudeerproject als de generieke doorontwikkeling van het platform.

Hary Selman is direct verantwoordelijk voor Twan van Buuren maar wordt additioneel technisch begeleid door lead ontwikkelaar van het platform: John Li.

Hary is afgestudeerd als Ir. Technische Bestuurskunde aan de TU Delft en heeft onder andere gewerkt als Technology Architect bij Accenture. Hij is de grondlegger achter het concept van de FormsEngine en kan gezien worden als de “product owner”. Hary vervult binnen Verne zowel de rol van Solution Architect als Technisch Projectleider. Met Hary zal naast de dagelijkse scrum ook dagelijks contact zijn over de specifieke voortgang van het project.

John Li is Lead Developer & Architect (en eigenaar) bij MyCubes. MyCubes voert een deel van de door-ontwikkeling van het Verne platform uit. Hij is tevens de lead developer achter de FormsEngine en is daarnaast veelal werkzaam als integratie architect bij overheid & zorg organisaties (o.a. Logius). Een dag in de twee weken werkt de student op locatie bij John werken om optimaal te kunnen afstemmen en code reviews te kunnen doen. Verder is hij voor dit project continu aanspreekbaar via e-mail, telefoon en Skype.

- Hary Selman begeleidt op:
    1. Aanpak, scope, proces en deliverables
    2. Overall Solution Architectuur
        1. Technisch Ontwerp / Architectuur
        2. Functionele bijdrage voor gebruiker / organisatie
        3. Beheerbaarheid / kosten van de oplossing
    3. Afhankelijkheden met externe partijen in ontwikkeling

- John Li begeleidt op:
    1. Inhoudelijke Technische ontwerp & architectuur
    2. Applicatie ontwikkeling & code reviews
    3. Integratie van opdracht binnen huidige applicatielandschap

Tabel 3 Contactgegevens afstudeerbegeleiders

Contactgegevens begeleiding		
	Hary Selman	John Li
e-mail	hary@verne.nu	john@mycubes.nl
telefoon	+31 (0) 6 2000 6102	+31 (0) 6 2151 3273
Skype	hary.selman	

6 Onderzoeksvragen

Verne levert een platform waarin verzekeringen geadministreerd worden. Daarbij wordt gebruik gemaakt van de FormsEngine om web-formulieren te maken voor aanvragen en wijzigingen die door derden op hun website kunnen worden opgenomen. In dit proces ligt er een bottleneck zowel qua doorlooptijd als foutgevoeligheid bij de mapping templates die bepalen hoe de data uit de formulieren in de webservices worden ingevuld. Om de kosten en fouten te beperken wil Verne deze mappings voor een zo groot mogelijk deel geautomatiseerd genereren.

In dit hoofdstuk wordt uitgewerkt hoe het zoeken naar de oplossing vorm krijgt als afstudeerproject.

Om het probleem van Verne op te lossen zal de hieronder volgende hoofdvraag beantwoord moeten worden; het antwoord op die vraag zal worden samengesteld uit de antwoorden op de deelvragen.

6.1 Hoofdvraag

Op welke wijze kan Verne de bestaande koppeling van verschillende formulieren aan webservices automatiseren om tijd en kosten te besparen?

6.2 Deelvragen

1. Hoeveel tijd kan Verne besparen door de mappings van de koppeling te laten genereren?
    a. Hoeveel tijd wordt er nu besteed aan de koppeling?
    b. Hoeveel tijd wordt er besteed aan de koppeling bij gebruik van de tool?
2. Welke functionaliteiten zijn er bij andere “data-mapping” tools die ook terug verwacht mogen/moeten worden bij de uiteindelijke oplossing?
3. Welke gegevens zijn nodig om data mappings te kunnen automatiseren?
4. Hoe past het automatiseren van de data mappings binnen de huidige architectuur van FormsEngine, Publisher & de DataIntegrator?
5. Welke wijzigingen zijn er binnen het applicatielandschap van de FormsEngine nodig (FormsEngine/Publisher/DataIntegrator)?
6. Hoe wordt een mapping in de software bepaald?

7 Methodologie

Bij het beantwoorden van de (deel-)vragen kunnen verschillende typen werkzaamheden worden onderscheiden. Deze opdrachttypes vragen allen om een eigen aanpak. De verschillende type opdrachten met hun aanpak worden hieronder uiteengezet. Tabel 4 Deelvragen en deliverables per methodiek geeft weer aan welke deelvragen en af te leveren resultaten de methodieken zijn gekoppeld.

Tabel 4 Deelvragen en deliverables per methodiek

Type opdracht/methodieken	Deelvragen	Deliverables
7.1.1 Effectmeting: Tijd die nodig is om een koppeling te maken.	1	1, 5
7.1.2 Korte inventarisatie bestaande data-mapping tools	2	2, 5
7.1.3 Opstellen architectuur van DataMapper binnen bestaande FormsEngine landschap	4, 5	3, 5
7.1.4 Ontwikkeling van de basis van de tool passend binnen de tijdslijnen van het afstudeerproject	2, 3, 6	2, 4, 5

7.1 Type opdracht en methodieken

7.1.1 Effectmeting: Tijd die nodig is om een koppeling te maken.

Verne heeft de wens uitgesproken om tijd te besparen op het creëren van de templates die de vertaling tussen formulier en webservice bepalen. Om te onderzoeken of de oplossingsrichting dit doel bereikt in een mate die evenredig is aan de investering, moet gemeten worden hoe groot de besparing is.

Daartoe zal eerst gekeken worden wat het aantal uren is dat zonder de nieuwe tool aan het koppelen van een formulier aan de webservices wordt besteed. Vervolgens wordt ook de tijd gemeten die wordt besteed aan de koppeling met gebruik van de tool.

Door de twee deelresultaten met elkaar te vergelijken kan de vraag beantwoord worden hoeveel tijd wordt bespaard.

7.1.1.1 Procedure

De volgende stappen zullen doorlopen worden:

a) Door medewerkers wordt geregistreerd hoeveel tijd ze aan welk onderdeel van welk project besteden. Dit is al een standaard proces binnen Verne. De gegevens in de ontwikkelingsmanagementsoftware moet alleen gefilterd worden op relevante werkzaamheden.
b) Door deze data te filteren kan per project gezien worden hoeveel tijd aan de data-koppeling is besteed. (deelvraag 1a en 1b) Verbeteringen in doorlooptijd zouden behalve door de tool ook door betere kennis van de Freemarker templating taal en de FormsEngine kunnen komen. Om dit effect zo veel mogelijk uit te sluiten in de meting is gekozen alleen naar de koppeling van de meest recente formulieren te kijken. Bij deze formulieren is de wijze van gebruik van de FormsEngine stabiel, evenals de kennis van de Freemarker template taal, terwijl hier bij eerdere formulieren nog extra tijd aan het leren kennen van deze tools werd

besteed.

c) Door de bestede uren per project zonder en met gebruik van de te ontwikkelen tool te vergelijken wordt duidelijk hoeveel tijd bespaard wordt door gebruik van de tool.

7.1.1.2 Categorieën

Bij het analyseren van de gemaakte koppelingen en de tijd die besteed werd aan het maken hiervan werd snel duidelijk dat er grote verschillen zaten in de aard van de koppeling en de uren die besteed werden. Daarom is gekozen om de te maken koppelingen volgens twee parameters in tweeën te splitsen. Er is gekozen voor een opdeling in simpele/complexe koppelingen en koppelingen met nieuwe en reeds bestaande methodieken/logica in de vertaling template.

Deze laatste categorie speelt alleen een rol bij de complexe koppelingen, hieronder verstaan we de koppelingen die transformaties op de gegevens moeten uitvoeren, anders dan ze één op één van de ene structuur naar de andere omzetten. Voor zulke simpele formulieren is geen sprake van 'nieuwe logica'. Zo ontstaan de categorieën in Tabel 5 Categorie indeling van de koppelingen.

Tabel 5 Categorie indeling van de koppelingen

Categorie koppeling	Complexiteit	Bestaand / nieuw
simpel	De koppeling omvat alleen het één op één overzetten van gegevens met dezelfde naam van de ene structuur naar de andere.	De logica is al eerder op dezelfde manier toegepast
complex bestaand	De koppeling bevat aanvullende logica, zoals gegevens die met verschillende namen moeten worden overgezet, samengevoegd of een loop of aanvullende condities nodig hebben.	De logica is al eerder op dezelfde manier toegepast
complex nieuw	De koppeling bevat aanvullende logica, zoals gegevens die met verschillende namen moeten worden overgezet, samengevoegd of een loop of aanvullende condities nodig hebben.	De logica is niet eerder op dezelfde manier toegepast

7.1.1.3 Dataverzameling

7.1.1.3.1 Bestede tijd zonder de tool

De registratie van zowel geplande taken als gewerkte uren is bij Verne een standaard proces. Vanuit deze data is de bestede tijd aan het inrichten van de koppeling van een formulier bepaald. Daarbij ontbreken nog de uren die besteed zijn aan het corrigeren van fouten achteraf. De opdrachtgever heeft gezegd dat hieraan nog eens 50% tot 100%, van de tijd besteed aan het inrichten, wordt besteed. Voor de meting is daarom met een opslag van 50% gewerkt.

Om de metingen zo zuiver mogelijk te kunnen vergelijken is het belangrijk andere factoren die de tijdsduur beïnvloeden zo veel mogelijk uit te sluiten. De belangrijkste factoren die daarbij een rol zouden kunnen spelen zijn de leercurve van Freemarker en de FormsEngine en de doorontwikkeling van de FormsEngine. Om deze invloeden zo veel mogelijk te beperken is gekozen alleen werk van dit jaar mee te rekenen(tot april 2015).

Voor ieder formulier wordt de bestede tijd onder een van de categorieën verzameld.

7.1.1.3.2 Bestede tijd met de tool

Het meten van de bestede tijd aan het maken van een koppeling met de tool zal niet kunnen door uren uit de uren-registratie te analyseren. Dit omdat de planning voor het meten van het resultaat niet afgestemd kan worden met het produceren van nieuwe koppelingen, omdat dit laatste niet voorspelbaar is. In plaats daarvan zal een aantal koppelingen dat is gerealiseerd in de projecten waarvan de uren zijn gemeten worden nagebouwd met gebruik van de tool. De bestede tijd hierbij wordt opgenomen en per categorie geregistreerd.

Bij de meting van de tijdsbesteding zonder de tool is rekening gehouden met een opslag in verband met tijdsbesteding aan support/ debuggen achteraf. Deze factor kunnen we nog niet geheel uitsluiten maar omdat het een geautomatiseerd proces betreft, kan worden aangenomen dat de tijd hieraan besteed wel kleiner wordt, gezien fouten maar op één plek gecorrigeerd moeten worden. Daarom is er rekening gehouden met een opslag van 25%.

7.1.1.3.1 Participanten

De werkzaamheden in de gemeten uren zijn door drie verschillende mensen uitgevoerd. Aan de meeste items is door twee of drie personen gewerkt. Het is niet mogelijk uit te splitsen wie aan welke deel heeft gewerkt. Er wordt in de metingen geen onderscheid per persoon gemaakt.

De achtergrond van de participanten is zeer divers. De voornaamste genoten opleidingen lopen uiteen van MBO, tot HBO informatica tot WO technische-bedrijfskunde.

7.1.1.4 Analyse

Door te vergelijken hoeveel tijd een koppeling uit een van de drie categorieën met en zonder gebruik van de tool in beslag neemt ontstaat een beeld van de tijdswinst die de tool oplevert. Bij de beperkte hoeveelheid data van de metingen kan niet worden vastgesteld dat gegevens per categorie normaal verdeeld zijn. Er is toch gekozen om te werken met een gemiddelde. Gezien de beperkte hoeveelheid data en het beperkt uiteenlopen daarvan, geeft dit toch een duidelijker beeld dan andere methodes, zoals box plotten of een vergelijking met hulp van een Mann-Whitney U test. Het gemiddelde eindresultaat over de verschillende categorieën is gewogen naar aantal formulieren per categorie.

Omdat de verwachte tijdswinst per categorie verschilt, is het belangrijk onderscheid te maken tussen de 3 verschillende categorieën. Naar verwachting zal de tijdswinst voor een bestaande systematiek/ logica nagenoeg honderd procent zijn. De winst voor nieuwe logica is naar verwachting in aantal bijna gelijk, maar naar verhouding veel kleiner. Hierbij zijn naast de herhaalde, weg geautomatiseerde werkzaamheden, ook de werkzaamheden van het definiëren van de nieuwe systematiek tijd rovend.

7.1.2 Korte inventarisatie bestaande data-mapping tools

Dit deel wordt uitgevoerd als een literatuuronderzoek of deskresearch en zal antwoord geven op deelvraag 2.

7.1.2.1 Procedure

Daarbij worden de volgende stappen doorlopen:

a) Inventarisatie van de te onderzoeken tools door interviews en gebruik van zoekmachines
b) Officiële documentatie van de relevante producten beschouwen

c) Documenteren hoe deze producten zich verhouden tot de te ontwikkelen tool

7.1.2.2 Dataverzameling

Uit interviews en resultaten van zoekmachines met de termen ‘data mapping’, ‘koppelen services’, ‘koppelen xml json’, ‘automatiseren service koppeling’ en hun Engelstalige equivalenten, wordt een lijst met applicaties opgesteld.

Van deze applicaties wordt de documentatie bestudeerd om de ondersteunde functionaliteiten te verzamelen.

7.1.2.3 Analyse

De verzamelde features worden met de benodigde features opgenomen in een tabel. Hierin wordt per tool aangegeven of deze een features wel of niet bevat.

7.1.3 Opstellen architectuur van DataMapper binnen bestaande FormsEngine landschap

De volgende fases zullen doorlopen worden bij het opstellen van de architectuur.

a) Bestaande documentatie over het platform en de daarin gebruikte pakketten doornemen
b) Overleggen met huidige product owner, architect & lead developer van het platform
c) UML model invoeren van huidige situatie en beoogde einddoel
d) Modellen verifiëren bij betrokkenen en waar nodig aanpassen. Daarbij worden de eisen gesteld zoals genoemd in ad. 3) van 7.2.1 Kwaliteitseisen deliverables.

Dit proces zal wederkerig afhankelijk zijn van 7.1.4 Ontwikkeling van de basis van de tool passend binnen de tijdslijnen van het afstudeerproject worden uitgevoerd. De ontwikkelde applicatie zal immers in de beoogde architectuurmoeten passen, maar de vereisten aan de architectuur zullen deels volgen uit inzichten verkregen bij de ontwikkeling van de applicatie.

7.1.4 Ontwikkeling van de basis van de tool passend binnen de tijdslijnen van het afstudeerproject

Hierbij worden de volgende stappen doorlopen:

a) Verzamelen functionaliteiten, features en componenten op basis van Korte inventarisatie bestaande data-mapping tools, gesprekken met betrokkenen. Hierbij zullen deelvraag 2 en 3 beantwoord worden
b) Aan het begin van het project wordt de backlog gevuld met alle producten en onderdelen die binnen de scope van het project horen.
c) Per sprint wordt gekeken welke items worden opgepakt. En deze worden uitgevoerd. Na elke sprint volgt een demo. Hierbij zal ook deelvraag 6 beantwoord worden.
d) Er wordt rekening gehouden met de planning van de deliverables binnen het project en de tijdslijnen van het afstudeerproject.

Op het eerste gezicht kunnen deze stappen lijken op een watervalproces. Toch is er eerder sprake van een iteratieve werkwijze, volgens scrum methodiek. Dit omdat hoewel de functionaliteiten vooral aan het begin worden verzameld, per sprint opnieuw wordt geanalyseerd in hoeverre het product helpt de doelstellingen van Verne te bereiken en wat de volgende (aanpassing van een) functionaliteit is om de doelstelling zo dicht mogelijk te naderen. Dat betekent in de praktijk dat bij iedere sprint opnieuw al deze stappen zullen worden doorlopen. Daarbij zal bij de eerste sprints het zwaartepunt meer bij a en b liggen en richting het einde van het project meer bij stap c.

7.1.4.1 Ontwikkelframework

Uit de kick-off meeting, aan de start van dit project bij MyCubes in Amsterdam (Buuren, 2015), is gebleken dat het voor de gebruiker het handigst is als de oplossing als plugin op de Publisher kan worden toegepast. Zo hoeft een gebruiker niet naar een nieuwe omgeving te navigeren en daar in te loggen. Ook is de Publisher een logisch punt in het bedrijfsproces, omdat hier de verwijzing naar de DataIntegrator wordt ingeregeld.

Om dit te faciliteren is het handig om in hetzelfde ontwikkelframework te werken, zeker omdat deze is ingericht voor het toevoegen van functionaliteiten middels plugins in een applicatie. De Publisher is net als de overige componenten van de FormsEngine een Grails applicatie. Dit houdt concreet in dat de oplossing zal worden ontwikkeld als Grails plugin.

Daarbij zal Redis als database gebruikt worden. Hier is voor gekozen omdat naar verwachting geen uitgebreide relationele structuren vastgelegd worden en zo op de bestaande architectuur wordt aangesloten met minimale aanpassingen (er hoeft geen andere database geïnstalleerd en beheerd te worden).

7.2 Deliverables

Er wordt gezocht naar een oplossing die binnen de periode van het afstudeerproject geïmplementeerd kan worden.

Er wordt verwacht dat aan het einde van het project de volgende zaken zijn opgeleverd:

1. Resultaatmeting: een vergelijking van de doorlooptijden voor en na de implementatie van de tool.
2. Onderzoeksresultaten van bondig onderzoek naar bestaande data-mapping tools
3. Architectuur DataMapper binnen het huidige applicatielandschap
4. Implementatie van basisversie van de DataMapper
5. Scriptie: Deze zal punten één, twee en drie en verantwoording /uitleg van vier bundelen

7.2.1 Kwaliteitseisen deliverables

De kwaliteitseisen die worden gesteld aan de deliverables vanuit de opdrachtgevers zijn:

Ad. 1) De resultaatmeting moet een analyse van de urenregistratie data doen, waaruit blijkt wat het voordeel is dat de tool oplevert.

Ad. 2) Overzicht van de features wordt in een matrix vergelijking opgenomen met daarin tevens volgens MoSCoW methode een waardoordeel vanuit Verne over de relevantie van de features.

Ad. 3) Een voor alle betrokkenen duidelijke architectuurplaat volgens de UML standaard met heldere omschrijving van de interfaces en functies die gekoppeld zijn aan de FormsEngine landschap. Hierbij gelden deze voorwaarden:

   a. De modellen zijn opgesteld volgens de UML standaard
   b. De modellen komen overeen met de werkelijke situatie.
   c. De modellen zijn compleet: ze bevatten alle componenten die op het betreffende niveau een rol spelen.

Ad. 4) Bij de applicatieontwikkeling worden de volgende eisen gesteld:

a. De opdrachtgever verwacht gestructureerde en over zichtelijke code volgens principes zoals DRY (Don't Repeat Yourself).
b. Tevens verwacht de opdrachtgever comments volgens geldige normen in de code die de functies en logica beschrijven.
c. Oplossingen voor problemen dienen gezocht te worden in gangbare design patterns.
d. Er moeten geautomatiseerde tests bij de applicatie zijn die laten zien of de programmatuur nog werkt na (code) aanpassingen. Uiteraard moeten deze ook met positief resultaat doorlopen worden.
e. In het project wordt een lijst met functionaliteiten opgesteld waarin is aangegeven welke functionaliteiten beoogd zijn, wat de prioriteit is (volgens MoSCoW) en welke binnen de scope van het project vallen. (Tabel 6 Featurelijst van te bouwen DataMapper geschaald volgens MoSCoW methode) Hiervan moeten in elk geval de must haves die binnen de scope van dit project vallen worden gerealiseerd in de prototype applicatie.

Ad. 5) Scriptie voldoet aan de eisen zoals gesteld in het beoordelingsformulier uit de afstudeerleidraad (Hogeschool Utrecht Faculteit Natuur en Techniek Institute for ICT, 2014).

De opdrachtgever verwacht tevens een proactieve houding waar het gaat om het verkrijgen van feedback op deelresultaten. De projectorganisatie is daarop ook ingericht.

7.2.2 Risico's

In onderstaande tabel worden de voornaamste risico's voor dit project beschreven.

Beschrijving risico	**Impact**	**Kans**	**Mitigerende activiteiten**
Functionaliteiten die moeten worden ontwikkeld aan de FormsEngine of Publisher applicatie om functionaliteiten binnen de datamapper mogelijk te maken.	Middel	Laag/ Middel	Dit kan ondervangen worden door handmatige in-/uitvoer van data, zoals bij de scope van het project aangegeven is. Tevens wordt het project als een aparte module / plugin beschreven waardoor afhankelijkheid van externe ontwikkeling geminimaliseerd wordt.
Kortstondige ziekte	Laag	Laag	Indien het een lichte ziekte betreft zal dit geen grote vertraging betekenen omdat alle betrokkenen de middelen hebben om vanuit huis contact te onderhouden en alle werkzaamheden te kunnen uitvoeren.
Langdurige, ernstige ziekte	Hoog	Zeer laag	Bij een ernstige, langdurige ziekte, zal een collega als vervanging dienen. Van de betrokkenen bij het project zijn de vervangende collega's al benoemd.
Het werken volgens scrum methodiek introduceert grote flexibiliteit in het inrichten van het project, nadeel hier van is dat dit ook onzekerheid inhoud.	Middel	Middel	Om te waarborgen dat het project wel binnen het kader van het afstudeerproject blijft, wordt de backlog aan het begin van het project gevuld. Vervolgens wordt per sprint gekeken welke items op dat moment aan bod komen.

Miscommunicatie	Middel	Middel	Er is dagelijks overleg tussen de bedrijfsbegeleider en de uitvoerder om te overleggen wat de stand van zaken is en welke betrokkenen geïnformeerd of geraadpleegd moeten worden. Tevens zorgt een demo aan het einde van elke sprint ervoor dat geen langdurige periode kan zijn waarop er verschillen van inzicht kunnen zijn.

7.3 Afbakening project

Het verzamelen en definiëren van de exacte functionaliteiten die de oplossing bevat gebeurt bij het beantwoorden van deelvragen twee, drie, vier en vijf. Het is daarom niet mogelijk op dit punt al per functionaliteit aan te geven welke wel of niet binnen de scope vallen. Dit zal bij het beantwoorden van de betreffende deelvragen aangegeven worden middels de MoSCoW methode.

Bij de beoordeling van de MoSCoW scores wordt rekening gehouden met de volgende factoren:

- Het afstudeerproject valt binnen een groter project met een bestaand applicatielandschap.
    - De ontwikkeling van de DataMapper (na finaliseren van de architectuur deliverables) is beperkt tot een standalone module / plugin.
    - Huidige applicatie architectuur ondersteunt de ontwikkeling van standalone modules / plugins.
- Van alle input en output informatiestromen van de DataMapper moet worden aangenomen dat deze tijdelijk handmatig moeten worden gekoppeld tussen systemen. Dit om het project te doen passen binnen de tijdslijnen van het afstudeerproject. Er zal dan ook geen sprake zijn van data integratie tussen de FormsEngine componenten en de DataMapper. Indien de voortgang voorspoediger loopt dan gepland, zou dit (gedeeltelijk) binnen de scope gehaald kunnen worden.

8 Resultaat software

8.1 Verwachte functionaliteit van de DataMapper

8.1.1 Inventarisatie van bestaande data mapping tools

Er is, in 4.2 Inventarisatie van bestaande data mapping tools (pagina 10), gekeken naar welke data-mapping tools beschikbaar zijn en of die de wensen van Verne mogelijk maken. Daaruit is gebleken dat er geen software beschikbaar is die voor nieuwe bron en doel datastructuren automatisch de transformatie genereert. Daarom zal de software ontwikkeld moeten worden.

8.1.2 Vereisten

Uit overleggen met de betrokkenen blijkt duidelijk dat de doelstelling is om de transformatie zo veel mogelijk automatisch te laten genereren met zo min mogelijk additionele configuratie. Daarbij zijn de volgende 4 functies van belang.

- De tool kan zelf omgaan met nieuwe structuren.
- Regels en conventies die zijn vastgelegd, moeten steeds voor een volgende transformatie kunnen worden hergebruikt.
- De regels en conventies/ logica van de tool moet aanpasbaar/ aanvulbaar zijn zonder de software opnieuw te moeten deployen.
- De tool moet zelf de benodigde en beschikbare informatie ophalen.

Om dit mogelijk te maken zijn in de volgende paragraaf alle functies opgenomen. Daarbij is ook aangegeven hoe belangrijk deze voor Verne zijn en of ze binnen de scope van dit project vallen.

8.1.3 Opsomming features met wenselijkheid

Naar aanleiding van de analyse van bestaande tools en doelstellingen van Verne hebben de betrokkenen bij deze afstudeeropdracht een lijst met functionaliteiten opgesteld, geordend naar categorie. Daarbij is middels de MoSCoW methode aangegeven in hoeverre deze functionaliteit voor Verne van belang is om de tool succesvol in te zetten. Daarnaast is gezamenlijk bekeken welke van de gewenste functionaliteiten binnen de scope van het afstudeerproject haalbaar zijn.

Dit resulteert in onderstaand overzicht van functionaliteiten. Binnen dit project zullen alleen de items die binnen de scope vallen worden opgenomen, op volgorde van belang volgens de ranking. De Must- en eventueel de should- en could-haves die niet binnen de scope vallen kunnen na afloop in een nieuw project worden toegevoegd aan de applicatie.

Tabel 6 Featurelijst van te bouwen DataMapper geschaald volgens MoSCoW methode

Functie categorie	Functionaliteit	Beschrijving	MoSCoW ranking	In scope afstuderen
Schema ondersteuning	Ondersteunen van Soap Webservices	Parsen van WSDL	MUST	Ja
	Ondersteunen van formulier structuur	Parsen van JSON (forms structuur)	MUST	Ja
	Ondersteunen REST Services	o.a. Parsen van JSON schema (http://json-schema.org/)	SHOULD	Nee
Aanmaken van data mapping rules	Framework voor het schrijven van data mapping rules	De gebruiker heeft de mogelijkheid regels in te voeren en te bewaren	MUST	Ja

	Schrijven van rules in Groovy taal	De ingevoerde regels kunnen als Groovy (script) worden uitgevoerd	MUST	Ja
	Ondersteuning andere programeertalen	De ingevoerde regels kunnen als een andere programmeer/script taal worden uitgevoerd	COULD	Nee
	Ondersteunen van zowel request als response berichten	De vertaling moet twee kanten op gegenereerd worden: het versturen en ook het antwoord van de server moet weer terugvertaald worden naar het formulier.	MUST	Ja
	Ondersteunen van herbruikbare rule sets voor verschillende formulieren	De regels moeten hergebruikt kunnen worden	SHOULD	Ja
	Ondersteunen van versie beheer van mapping rules	Verschillende versies van en mapping rule kunnen worden aangemaakt en bewaard	SHOULD	Nee
Rule engine	DataMapping rules toe passen en de templates genereren	De rules worden in de betreffende programmeertaal uitgevoerd om een resultaat te genereren	MUST	Ja
	Ondersteunen van zowel request als response berichten	De vertaling moet twee kanten op gegenereerd worden: het versturen en ook het antwoord van de server moet weer terugvertaald worden naar het formulier.	MUST	Ja
	Ondersteunen van versie beheer van gegenereerde templates	Verschillende versies van en mapping rule kunnen worden aangemaakt en bewaard	SHOULD	Nee
Integratie	Plugin op de FormsEngine Publisher	De DataMapper kan aan de Publisher worden toegevoegd	MUST	Ja
	Data integratie met DataIntegrator	Direct uploaden van template naar de juiste app	MUST	Nee
	Override van gegenereerde templates voor specifieke formulieren	De gegenereerde templates moeten handmatig kunnen worden aangepast	SHOULD	Nee
	Ondersteuning voor omgevingsvariabelen van DataIntegrator	De in DataIntegrator bestaande variabelen kunnen worden gebruikt in de rules	COULD	Nee
GUI	Visualisatie van de data mappings	de data mapping heeft een visuele representatie die de mapping verduidelijkt	WON'T	Nee
	Data mappings maken middels grafische mappings	De mapping rules kunnen worden aangemaakt middels een visuele representatie die de werking verduidelijkt.	WON'T	Nee

8.2 Gegevens waarmee het genereren van data mappings werkt

8.2.1 Ondersteuning van type mappings

Bij het nagaan van de mappings die de tool moet kunnen maken is met name de richting van de transformatie relevant. Daarbij zijn er twee typen:

- **Request:** Van Form naar API
- **Response:** Van API naar Form

8.2.2 Benodigde gegevens

De volgende gegevens zijn nodig om de juiste data mappings te bepalen:

- Structuur/velden output
- Gegevens die gebruikt worden in template:
    - Variabelen die beschikbaar zijn bij het toepassen van een template (de zgn. header properties)
    - Structuur/velden van de brongegevens

De structuur en velden van de gegenereerde template worden bepaald door de service waar ze heen worden gestuurd. Dit is afhankelijk van of het een request of response betreft de WSDL of form-definitie (zie respectievelijk punt 1 en 2 in Figuur 6 Schematische weergave beoogde architectuur op pagina 30).

Welke gegevens ingevuld kunnen worden in de velden, wordt bepaald door de form-definitie of het response bericht uit de WSDL (respectievelijk voor de request en de response). Daarnaast zijn er nog gegevens, header properties, die door de DataIntegrator beschikbaar worden gesteld aan de template. Bijvoorbeeld inloggegevens die voor een service vereist zijn, deze worden niet in een formulier opgenomen, maar zijn binnen de DataIntegrator instelbaar.

8.3 Benodigde wijzigingen binnen applicatielandschap FormsEngine

In deze paragraaf zal een beeld worden gecreëerd van de wijzigingen die moeten worden doorgevoerd in het applicatielandschap van de FormsEngine om de gewenste features binnen de scope van het afstuderen af te ronden.

Uit het voorgaande hoofdstuk blijkt dat de volgende zaken nodig zijn.

- Toevoegen mapper component
- Structuur externe service: ligt daar, in geval soap service al beschikbaar
- Structuur form: bestaat al
- Header properties: DataIntegrator API
- Integratie gegenereerde templates: DataIntegrator API

Een aantal van deze zaken is al beschikbaar en sommige zaken zullen moeten worden toegevoegd. De volgende componenten dienen aan het applicatielandschap te worden toegevoegd:

- Mapper component
- DataIntegrator moet ontsloten worden voor het mapper component.

8.4 Toekomstige architectuur aansluitend bij oplossing

8.4.1 Inleiding

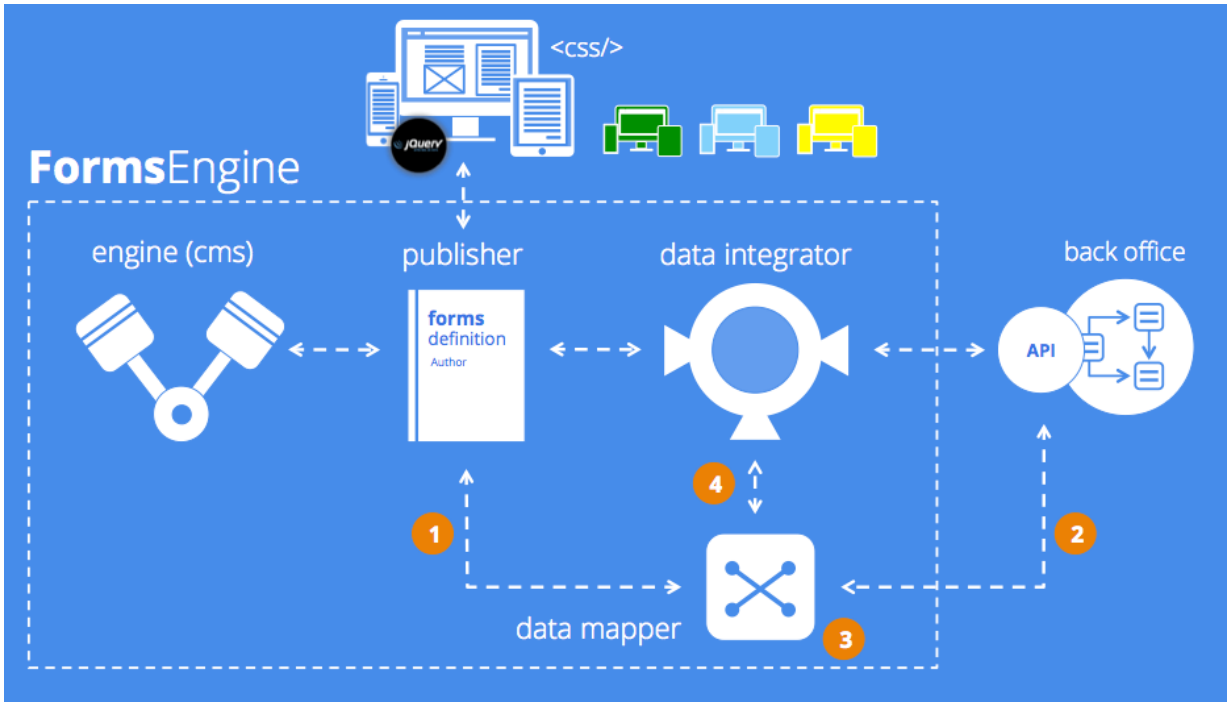
Uit onderzoek naar bestaande tools en vergelijking met de wensen en eisen van Verne volgt dat er een nieuwe applicatie ontwikkeld moet worden die de gewenste functionaliteiten bundelt. In de kick-off meeting is besproken in welke vorm en omgeving deze applicatie beschikbaar zou moeten zijn.

Om uiteindelijk tot een volledige integratie van de DataIntegrator-mapper te komen is een overzicht nodig van alle onderdelen die in de architectuur moeten worden opgenomen. Hoewel het implementeren van de integratie niet binnen de scope van dit project valt, valt het wel binnen de opdracht om een overzicht te geven van de beoogde architectuur. Deze wordt gegeven en vervolgens wordt aangegeven welke onderdelen binnen dit project wel en niet worden geïmplementeerd.

8.4.2 Uiteindelijke architectuur

In de volgende afbeelding zijn de toevoegingen aan de architectuur opgenomen. In de bijlages is dit in UML weergegeven. Op deze plek is gekozen voor een versimpelde weergave. De toevoegingen aan de hand van de nummers uit Figuur 6 Schematische weergave beoogde architectuur zijn:

1. De data mapper haalt de structuur van het formulier op van de Publisher. Dit is een nieuwe interactie op de bestaande interface op de Publisher. Buiten de data mapper zelf vergt dit geen nieuwe componenten.
2. De data mapper haalt de structuur van de webservice op. Dit is een nieuwe interactie op de bestaande externe service. Buiten de data mapper zelf vergt dit geen nieuwe componenten.
3. De data mapper zelf is een nieuw component. Deze kan als plugin aan een van de bestaande applicaties worden toegevoegd.
4. De data mapper zal uiteindelijk data moeten koppelen met de DataIntegrator. Hiervoor is een nieuwe Interface op de DataIntegrator vereist.



Figuur 6 Schematische weergave beoogde architectuur

8.4.2.1 *Afbakening van de realisatie binnen dit project*

De opdracht vraagt om een compleet overzicht van de functionaliteiten die de tool nodig heeft om optimaal inzetbaar te zijn, en een architectuur die dit mogelijk maakt. Echter is de ontwikkeling van de programmatuur binnen dit project beperkt tot een zelfstandige applicatie.

Dit houdt in dat de koppeling met de nog te realiseren webservices en de realisatie van deze buiten de scope van dit project vallen. Hier is voor gekozen omdat realisatie van de betreffende services niet binnen de tijdslijn van dit project past. Daarbij is dit ook niet nodig om wel gebruik te kunnen maken en voordeel te hebben van de tool.

Daarnaast zijn op dit moment alleen koppelingen naar SOAP services relevant voor de externe services. Andere externe services zoals REST services zullen voor dit project buiten beschouwing gelaten worden.

Voor de architectuur die gerealiseerd wordt binnen dit project betekent dit dat de zogenaamde TemplateService op DataIntegrator komt te vervallen(4. In Figuur 6 Schematische weergave beoogde architectuur).

Verder is de user interface bij dit project van ondergeschikt belang. Er wordt wel verwacht dat deze bestaat, om de applicatie te kunnen gebruiken, echter worden hier geen kwalitatieve of esthetische eisen aan gesteld.

8.5 Realisatie prototype DataMapper

8.5.1 inleiding

Onderdeel van dit project is de realisatie van een eerste versie van de DataMapper, de tool die Verne moet helpen het koppelen van formulieren aan de webservice te verbeteren. In dit deel wordt

uiteengezet wat de belangrijkste overwegingen waren bij het maken van deze applicatie en in welke beslissingen die hebben geresulteerd voor de applicatie.

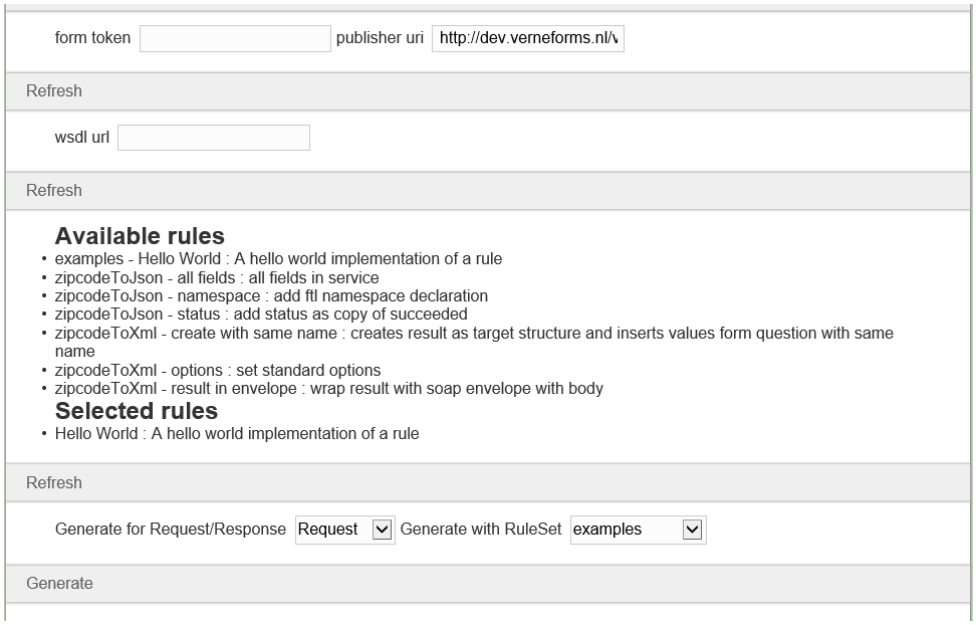
8.5.2 Eerste iteratie ontwikkeling

Om de haalbaarheid aan te tonen en gewend te raken aan de techniek, is heel vroeg in het traject begonnen aan de implementatie van een basis versie van de tool. Als basis voor het configureren van de werking is gekozen om te werken met regels, die allemaal hun eigen stukje functionele logica toepassen.

Deze worden geïmplementeerd als een verzameling scripts, die serieel worden uitgevoerd.

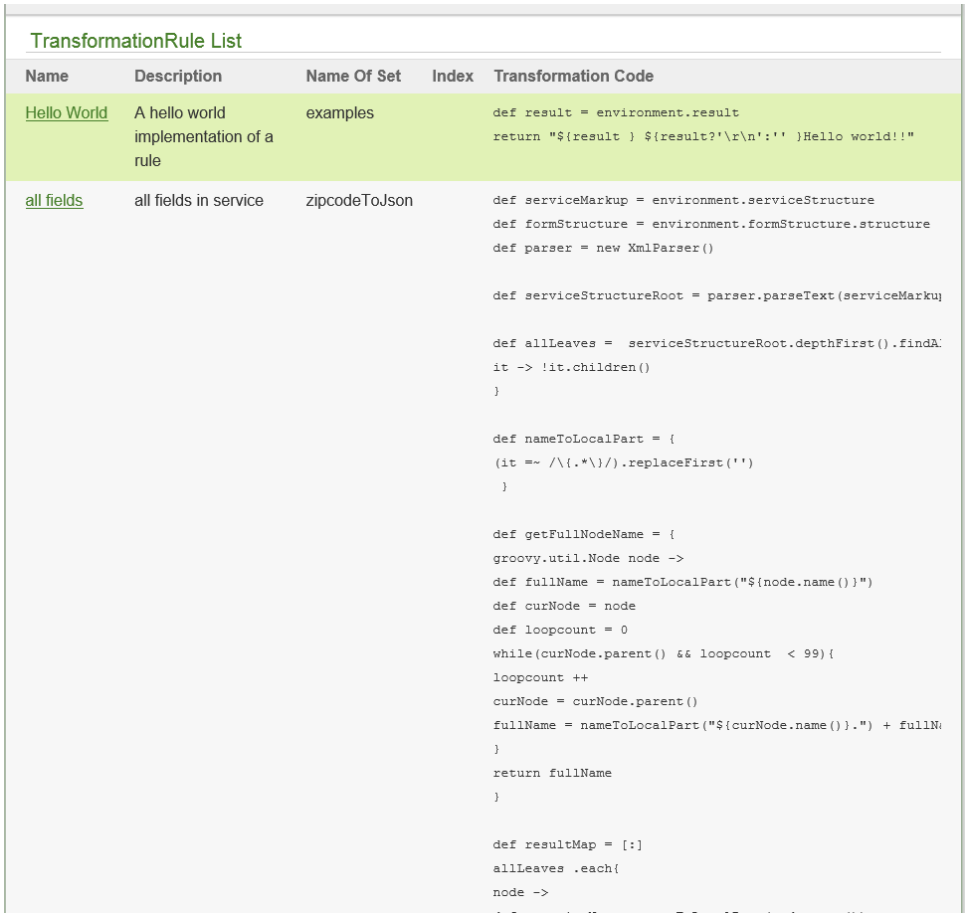
8.5.2.1 Resultaat eerste iteratie

Dit prototype werkt en maakt alle gewenste constructies in de output mogelijk. Hieronder zijn screenshots opgenomen die de werking van deze versie laten zien. Bij deze versie moest de gebruiker zelf kiezen welke 'set' van regels wordt gebruikt en worden alle regels in de gekozen set onafhankelijk van elkaar na elkaar uitgevoerd. (op volgorde van een door de gebruiker te kiezen index)



Figuur 7 Selectie ruleSet voor genereren

Gevolg hiervan was dat in de rules met alle wijzigingen/aangebrachte structuren of output van de voorgaande regels rekening gehouden moet worden en opnieuw bepaald moet worden op welke plekken teksten ingevoerd of aangepast moeten worden. Door deze factoren werden de regels erg lang en bevatten veel code die nodig was voor de werking, maar niet weerspiegelde wat de rule functioneel moet doen. In figuur 8 is een voorbeeld van zo'n regel te zien waarin veel code staat die voorgaande resultaten bekijkt, eventueel parset en afhankelijk van wat het is dan pas zijn eigen output produceert.



TransformationRule List

Name	Description	Name Of Set	Index	Transformation Code
Hello World	A hello world implementation of a rule	examples		def result = environment.result return "${result } ${result?'\r\n':'' }Hello world!!"

all fields	all fields in service	zipcodeToJson		def serviceMarkup = environment.serviceStructure
				def formStructure = environment.formStructure.structure
				def parser = new XmlParser()
				
				def serviceStructureRoot = parser.parseText(serviceMarku
				
				def allLeaves = serviceStructureRoot.depthFirst().findA
				it -> !it.children()
				}
				
				def nameToLocalPart = {
				(it =~ /\{.*\}/).replaceFirst('')
				}
				
				def getFullNodeName = {
				groovy.util.Node node ->
				def fullName = nameToLocalPart("${node.name()}")
				def curNode = node
				def loopcount = 0
				while(curNode.parent() && loopcount < 99){
				loopcount ++
				curNode = curNode.parent()
				fullName = nameToLocalPart("${curNode.name()}.") + fullN
				}
				return fullName
				}
				
				def resultMap = [:]
				allLeaves .each{
				node ->

Figuur 8 Regels met lange, niet functionele code

8.5.2.2 Gevolgtrekkingen uit eerste iteratie

Om dit te ondervangen is gekozen de werking van de regels verder te ontwerpen, het liefst volgens patterns die gangbaar zijn om soortgelijke problemen op te lossen. Er is uitgewerkt hoe de rules zouden uitpakken bij implementatie volgens verschillende ontwerpen. In de bijlage “hypothetische rules bij design” is te zien hoe dit er voor verschillende varianten uit zou zien, waarbij nog geen rekening is gehouden met gebruik van de template engine(zie laatste paragraaf van dit hoofdstuk).

De volgende drie opties zijn daarbij overwogen:

1. Composite en observer pattern:
   Rules opgebouwd als een structuur volgens het composite pattern, met een eigenschap die toepasbaarheid bepaalt, via een implementatie volgens het observer pattern.
2. Strategy pattern:
   Structuur wordt bepaald door statische code; user bepaalt de invulling van de vooraf gedefinieerde delen.
3. Freeform templating:
   Alleen beginpunt staat vast, vanaf daar kan de gebruiker zelf in zijn rules bepalen wat de vervolg of deel stap is.

Design pattern	1) Composite en observer	2) Strategy pattern	3) Freeform templating
Voordelen	• goed testbaar • gebruiker heeft veel vrijheid • Zelfde manier van omgaan met simpele en complexe rules	• eenvoudig voor gebruiker • goed testbaar	• gebruiker heeft alle vrijheid • ook om fouten te maken
Nadelen	• complexere doorloop van de regels	• bij wijziging in logica uitrol nodig • daardoor weinig flexibel	• weinig complexe statische code • beheer rules complex • lastig te testen

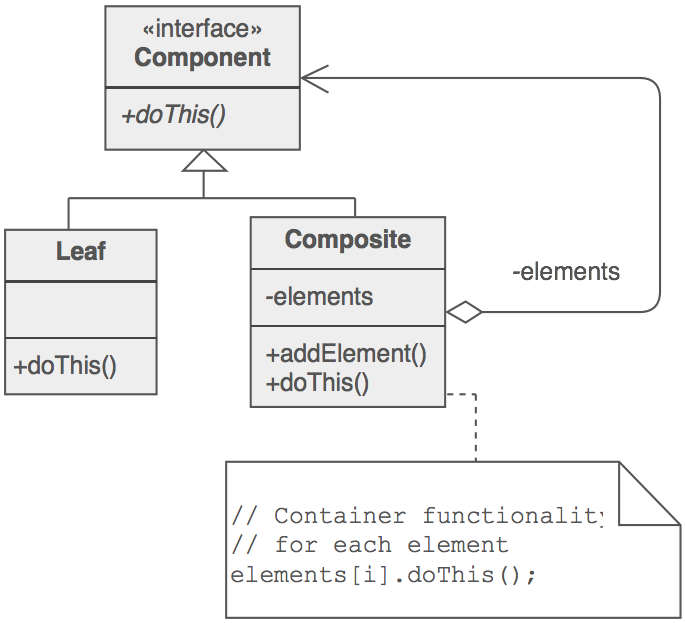
De eerste optie resulteert in complexere doorloop van de regels, maar omdat de werking in statische code ligt is deze goed testbaar.

Bij de tweede optie loop je al heel snel tegen beperkingen aan wanneer wijzigingen in de logica aangebracht moeten worden. De statische code moet in zo'n geval aangepast/aangevuld en uitgerold worden.

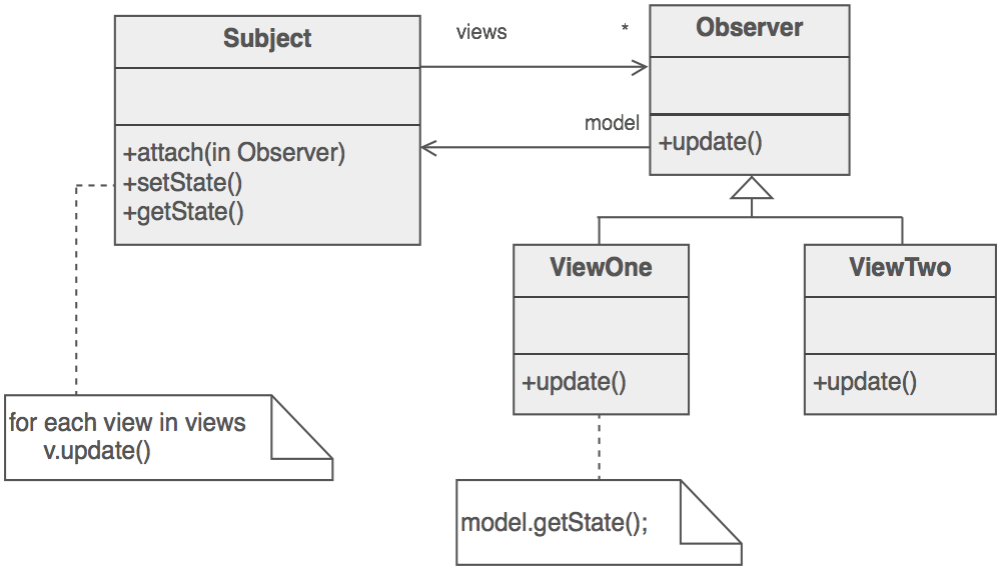
Bij de derde optie geef je de gebruiker alle vrijheid en verantwoordelijkheid. Hierdoor kan het beheer van de regels erg complex worden, tevens wordt het lastiger te testen.

De keuze is, na vergelijking van de hypothetische rules en implicaties van de samenstelling, gevallen op een combinatie van composite en observer patterns en gebruik van de standaard Groovy template engine.

Door rules volgens het composite pattern te implementeren kunnen complexe en simpele bewerkingen op eenzelfde manier worden aangeroepen en complexe koppelingen kunnen worden opgebouwd uit verschillende simpelere. Door dit te combineren met een eigenschap op de rules die bepaalt of zij voor een bepaalde context (plaats in de doel structuur) toepasbaar zijn, ontstaat een systeem waarbij de rules aan alle eisen van flexibiliteit en complexe output kunnen voldoen. Terwijl de rules zelf simpel kunnen zijn en één duidelijk gedefinieerde transformatie beschrijven. (Design Patterns, 2015) (Design Patterns | Object Oriented Design, 2015). In figuur 7 en 8 zijn uitwerkingen van de patronen in UML weergegeven. In de implementatie die is toegepast is het composite pattern in de meest gebruikelijke vorm toegepast. Bij de implementatie van het observer pattern, om regels bij een veranderende context te laten checken of zij van toepassing zijn, is gekozen voor een gedeeltelijke implementatie. Hier is voor gekozen omdat de registratie onnodige overhead gaf, omdat het in onze case altijd de children (in Figuur 9 elements genoemd) uit de composite structuur betreft.



Figuur 9 Composite design pattern



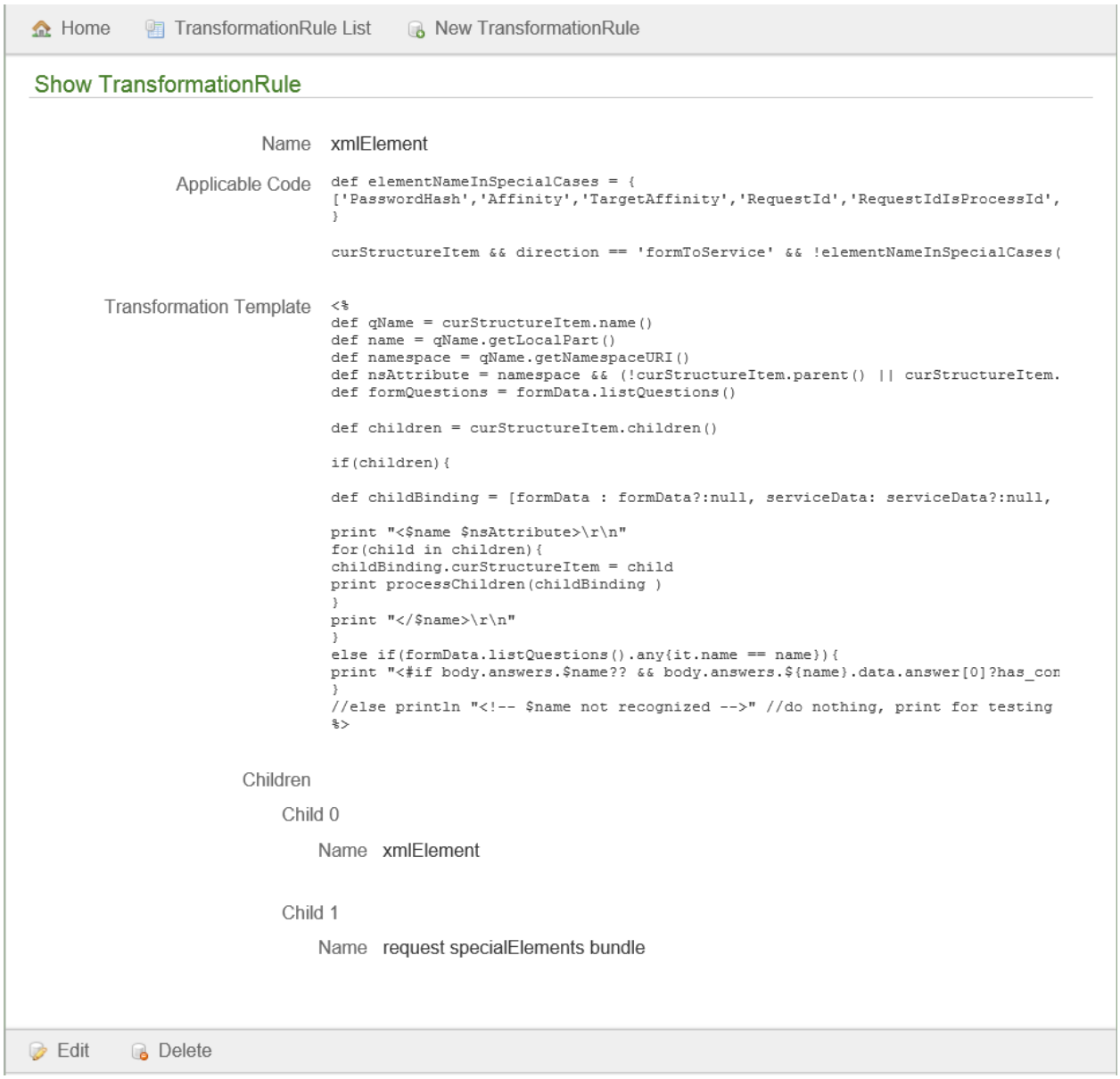
Figuur 10 Observer design pattern

Het gebruik van de Groovy tekst-template engine maakt het mogelijk de hoeveelheid niet expressieve code verder terug te brengen. Zo hoeft voor de output niet meer gewerkt te worden met een StringWriter in een variabele om de uitvoer terug te geven. (The Groovy programming language, 2015)

8.5.3 Tweede iteratie software ontwikkeling

In deze iteratie worden de gevolgtrekkingen van de voorgaande iteratie verwerkt, om zo tot een beter inzetbare tool te komen. Dat wil zeggen dat de rules zijn herschreven volgens het composite pattern. Ook is de verantwoordelijkheid om te bepalen of een rule van toepassing is bij de rule zelf

belegd, zoals ook in het observer pattern. In Figuur 11 Voorbeeld van een rule in de op patterns gebaseerde structuur is een voorbeeld hiervan opgenomen.



Figuur 11 Voorbeeld van een rule in de op patterns gebaseerde structuur

9 Resultaat effectmeting

9.1 Tijdsbesteding voor het genereren van data mappings

9.1.1 Bestede tijd zonder de tool

In onderstaande tabel zijn de resultaten weergegeven naar de tijdsbesteding aan het maken van een koppeling in de oude situatie. Hierin is rekening gehouden met het debuggen en correcties achteraf, na afsluiten van de oorspronkelijke taken, middels een opslag van 50%. In de bijlage, 13.4.1 Zonder de Tool, staan de uitgebreidere meet data.

Tabel 7 Bestede tijd koppeling formulieren zonder tool

Categorie koppeling	Aantal koppelingen	Gemiddelde duur inregelen koppeling
simpel	4.5	10.33
complex bestaand	3	14.50
complex nieuw	2	24.00
Gewogen gemiddelde		14.53

Uit dit overzicht blijkt dat er inderdaad een significante hoeveelheid tijd wordt besteed aan het maken van de koppeling voor ieder formulier. Dit maakt de wens dit proces te versnellen zeer relevant, aan gezien de verwachting is dat de aantallen te koppelen formulieren in de nabije toekomst nog zullen stijgen. In de volgende paragrafen zal worden gekeken in hoeverre de ontwikkelde tool daar aan bijdraagt.

9.1.2 Bestede tijd met de tool

In de volgende tabel staan de resultaten van de tijdmeting over het maken van koppelingen met gebruik van de DataMapper tool. Hierbij is rekening gehouden met support en correcties achteraf middels een opslag van 25%. De uitgebreidere meet data staan in de bijlages, 13.4.2 Met de Tool.

Tabel 8 Aan koppelingen bestede tijd bij gebruik van de tool

Categorie koppeling	Aantal koppelingen	Gemiddelde duur inregelen koppeling
simpel	2	1.20
complex bestaand	6	1.37
complex nieuw	2	13.75
Gewogen gemiddelde		3.81

9.1.3 Analyse

In Tabel 9 Vergelijking tijdsbesteding met en zonder tool worden de meetresultaten vergeleken. Uit de metingen blijkt dat er een tijdswinst is van zo'n 9/10^{e} bij de koppelingen volgens bestaande/ eerder toegepaste systematiek. Voor koppelingen volgens nieuwe systematiek geldt een tijdswinst van bijna de helft. Dit komt overeen met de verwachtingen, dat overal een significante winst geboekt kon worden, maar op de herhaal-werkzaamheden nog veel meer dan op het maken van templates met (gedeeltelijk) nieuwe logica.

Uitgaande van 92 besteedde uren in één kwartaal (volgens de meting zonder tool), betekend dit resultaat op jaarbasis een besparing van 157 tot 325 uur, volgens het gewogen gemiddelde 271 uur.

Daarbij moeten in de huidige versie nog handmatige acties worden uitgevoerd om de gegenereerde templates in de DataIntegrator te plaatsen. Door verdere ontwikkeling van de tooling, kan op dit vlak nog meer winst behaald worden. Ook is de verwachting dat op lange termijn de winst zal toenemen omdat er steeds minder koppelingen uit de 'nieuwe' categorie zullen voorkomen.

Tabel 9 Vergelijking tijdsbesteding met en zonder tool

Categorie koppeling	**Zonder de tool**	**Met de tool**	**Verschil**
simpel	10.33	1.20	88%
complex bestaand	14.50	1.37	91%
complex nieuw	24.00	13.75	43%
gewogen gemiddelde	14.53	3.81	74%

10 Conclusie

Verne zet de FormsEngine in om specifieke formulieren voor verzekeringen te creëren die in websites opgenomen kunnen worden. De gegevens die verstuurd worden bij het verzenden van deze formulieren worden binnen de FormsEngine in de DataIntegrator gekoppeld aan SOAP services van Verne. Daarbij worden de gegevens via een mapping vertaald, zodat de service van het formulier (Rest service met Json data formaat) kan communiceren met de SOAP services.

Het maken van deze mappings wordt nu handmatig uitgevoerd. In het merendeel van de gevallen zijn dit herhaal-oefeningen. Dit kost erg veel tijd en door het steeds opnieuw handmatig handelen ontstaan steeds fouten.

Om een oplossing te vinden voor dit probleem is de volgende hoofdvraag voor dit project geformuleerd:

> *Op welke wijze kan Verne de bestaande koppeling van verschillende formulieren aan webservices automatiseren om tijd en kosten te besparen?*

Uit de deelvragen volgt de volgende conclusie.

De belangrijkste functionaliteit die Verne nodig heeft is het geautomatiseerd genereren van nieuwe data koppelingen voor verschillende formulieren. Er zijn geen tools beschikbaar die deze functionaliteit bieden. Wel is er bij verschillende beschikbare tools gekeken naar functionaliteiten die ook in de oplossing voor Verne toegevoegde waarden hebben. Verne moet daarom zelf een tool ontwikkelen die de volgende functionaliteit biedt:

- Ondersteunen van SOAP services
- Ondersteunen van formulier structuur
- Framework voor het schrijven van data mapping rules
- Ondersteunen van zowel request als response berichten (de koppeling moet dus twee kanten op werken)
- Inzetbaar als standalone applicatie en als plugin op applicaties uit de bestaande FormsEngine
- Geautomatiseerd genereren van nieuwe data koppelingen

De gegevens waar de tool rekening mee moet houden zijn:

- De tool maakt onderscheid tussen de request en response koppeling
- De structuur/velden die als resultaat worden verwacht
- Gegevens die gebruikt worden in template:
    - Variabelen die beschikbaar zijn bij het toepassen van een template (de zogenaamde header properties)
    - Structuur/velden van de bron gegevens

Dit betekent dat de volgende elementen aan het applicatielandschap van de FormsEngine moeten worden toegevoegd:

- De DataMapper component zelf
- DataIntegrator moet via en API ontsloten worden voor het DataMapper component.

Binnen dit project is een prototype van deze mapper-tool ontwikkeld. Deze werkt op basis van regels

die de gegevensstructuren als input nemen en via een template systeem een koppeling genereren. Om complexe structuren en bewerkingen mogelijk te maken, zonder het gebruik van de regels ingewikkeld en verschillend te laten worden, is gekozen de regels samen te stellen volgens het Composite pattern.

Uit metingen blijkt dat dit prototype al een grote besparing aan bestede uren oplevert. De besparing varieert afhankelijk van het type koppeling dat gegenereerd wordt tussen de 43% en 91% van de oorspronkelijk bestede tijd. Concreet betekent dit op jaarbasis **een besparing van 157 tot 325 uur, volgens het gewogen gemiddelde 271 uur.** Deze besparing zal nog groter worden door de applicatie door te ontwikkelen tot een volledige applicatie die met de DataIntegrator integreert via een API beschreven in de resultaten met betrekking tot het ontwerp van de applicatie.

11 Aanbevelingen

De tool die in dit project is ontwikkeld biedt grote voordelen boven de huidige werkwijze bij het maken van de koppelingen. De belangrijkste hiervan is de tijdsbesparing en de beperking van de herhaal-handelingen.

De tijdswinst die geboekt wordt is zeker de moeite waard. Echter kan deze nog vergroot worden als de DataMapper tool verder wordt ontwikkeld. De voornaamste aanbeveling is daarom ook om de tool verder te integreren in het bestaande FormEngine landschap door de API's met de DataIntegrator en Publisher verder uit te breiden. De focus is daarbij om de uitrol (deployment) van de data mappings verder te automatiseren. Dit reduceert het aantal manuele handelingen en tijdsbesteding welke eerder vanwege planning en risico afbakening buiten de scope van het afstudeerproject zijn gehouden.

Om de doorontwikkeling van DataMapper te borgen, is te adviseren om de tool binnen de bestaande toolset van Verne op te nemen en toe te wijzen aan de bestaande product owner van het FormsEngine landschap. Praktisch komt het er op neer dat de DataMapper binnen het ontwikkel- en support traject van Verne wordt opgenomen en daarmee de verdere doorontwikkeling en verbeteringen worden gegarandeerd ten einde de doelstelling van Verne te behalen.

Ten slotte is onderzoek aan te bevelen hoe de tool breder kan worden ingezet om broncode, instellingen of testscripts te genereren. Zo maakt Verne bijvoorbeeld sinds kort geautomatiseerde tests voor formulieren. Deze testscripts baseren zich op de id's (kunnen gezien worden als uniek identificeerbare namen) van elementen welke nu handmatig worden ingevoerd maar zouden wellicht ook automatisch uit de formulier structuur gegenereerd kunnen worden. De standalone en plugin architectuur van de DataMapper, waartoe eerder in het afstudeerproject is besloten, ondersteunt een dergelijke doorontwikkeling.

12 Bronnenlijst

altova. (sd). Opgehaald van altova: http://www.altova.com

Beckwith, B. (2013). *Programming Grails (Best practices for experienced Grails developers).* Sebastopol: oreilly.

Berg, K. P. (2007, 11 12). *client-side-wsdl-processing-with-groovy-and-gant.html*. Opgehaald van http://www.javaworld.com: http://www.javaworld.com/article/2077790/open-source-tools/client-side-wsdl-processing-with-groovy-and-gant.html

Buuren, T. v. (2015, 1 30). Verslag project kickoff automatische koppeling formulieren aan webservices 30-01-2015 bij MyCubes. Schiphol-Rijk, Noord Holland, Nederland.

Design Patterns | Object Oriented Design. (2015, 04). Opgehaald van oodesign.com/: http://www.oodesign.com/

Design Patterns. (2015, 04). Opgehaald van sourcemaking.com: https://sourcemaking.com/design_patterns

easywsdl. (2015, 2 11). Opgehaald van easywsdl.ow2.org/: http://easywsdl.ow2.org/index.html

freemarker. (2015). *FreeMarker Java Template Engine*. Opgehaald van http://freemarker.org/

FreeMarker. (sd). *FreeMarker Java Template Engine - Overview*. Opgehaald van freemarker.org/: http://freemarker.org/

grails.org. (2015, 02 01). *grails.org/learn*. Opgehaald van grails.org/learn: https://grails.org/learn

Hogeschool Utrecht Faculteit Natuur en Techniek Institute for ICT. (2014). *AFSTUDEERLEIDRAAD BUSINESS IT & MANAGEMENT SOFTWARE ENGINEERING INFORMATION ENGINEERING SYSTEM AND NETWORK ENGINEERING TECHNISCHE INFORMATICA* . Utrecht: Hogeschool Utrecht.

jackson-module-jsonSchema. (2014, 2 8). Opgehaald van https://github.com/FasterXML/jackson-module-jsonSchema: https://github.com/FasterXML/jackson-module-jsonSchema

JacksonTreeModel. (2015, 02 10). Opgehaald van http://wiki.fasterxml.com: http://wiki.fasterxml.com/JacksonTreeModel

jax-ws RuntimeWSDLParser.html. (sd). Opgehaald van https://jax-ws.java.net: https://jax-ws.java.net/nonav/2.2.7/javadocs/com/sun/xml/ws/wsdl/parser/RuntimeWSDLParser.html

parse-wsdl-java-api.htm. (2015, 2 12). Opgehaald van membrane-soa.org: http://www.membrane-soa.org/parse-wsdl-java-api.htm

Pavlovic, F. (sd). *jamper*. Opgehaald van sourceforge.net: http://jamper.sourceforge.net/

redis.io. (2015, 02 01). *redis.io/documentation*. Opgehaald van redis.io/documentation: http://redis.io/documentation

The Groovy programming language. (2015). Opgehaald van groovy-lang.org: http://www.groovy-lang.org/

van Buuren, T. (2015, 02 03). Verslag functionaliteit bespreking automatische koppeling formulieren aan webservices 03-02-2015. Culemborg, Gelderland, Nederland.

van Buuren, T. (2015, 1 30). Verslag project kickoff automatische koppeling formulieren aan webservices 30-01-2015 bij MyCubes. Schiphol-Rijk, Noord Holland, Nederland.

woden. (2015, 2 1). Opgehaald van ws.apache.org: http://ws.apache.org/woden/

www.stylusstudio.com. (sd). *stylusstudio xml_mapper*. Opgehaald van stylusstudio.com: www.stylusstudio.com/xml_mapper.html

13 Bijlages

13.1 Plan van aanpak

Plan van Aanpak automatische koppeling formulieren aan webservices

Naam: Twan van Buuren
Studentnummer: 1604889
Opleiding: Opleiding informatica
Opdrachtgever: Verne
Eerste examinator: Linda Terlouw
Tweede examinator: Rory Sie

Inhoud

Inleiding

De auteur, Twan van Buuren, volgt de opleiding informatica, richting software Engineering, aan de HU(Hogeschool Utrecht). Daarnaast werkt hij parttime voor Verne, dat administratie software voor verzekeringen als Software As A Service (SaaS) oplossing levert.

Bij het invullen van zijn afstudeerproject, wist de auteur gelijk dat hij een praktijkopdracht wilde uitvoeren. Daarbij zijn er een aantal verschillende opties overwogen. Een aantal kwamen van Verne af en een aantal kwamen van de auteur zelf.

Het verbeteren van de workflow bij het inrichten van formulieren voor nieuwe klanten stond hoog in beide lijstjes met onderwerpen. Hierbij was al duidelijk dat de bottleneck vooral zat bij het vertalen van de gegevens zoals die uit een door klanten ingevuld formulier komen naar de webservices van de verzekeringsadministratie.

Voor de auteur was dit enerzijds inhoudelijk een interessante opdracht, onder andere omdat het verschillende technologieën combineert, een flexibele oplossing vraagt en de auteur het weg automatiseren van werk een belangrijk element van software ontwikkeling vindt. Anderzijds was het ook leuk aan een probleem te werken wat men uit de praktijk kent.

Dit document omschrijft verder eerst uitgebreider wat voor bedrijf Verne is, wat hun probleemstelling is en hoe de projectorganisatie is geregeld. Vervolgens wordt uiteengezet hoe dit project als afstudeeropdracht wordt aangepakt, beginnend met de vraagstelling, gevolgd door de methodieken en technieken, verwachtte resultaten, afbakening van het project planning en een voorlopige literatuurlijst.

Opdracht vanuit de opdrachtgever

Verne als organisatie

Verne ontwikkelt en levert een backoffice (SaaS) platform voor verzekeringsmaatschappijen. Dit platform is volledig op webservices gebaseerd en biedt eindklanten via een portaal toegang tot een “mijn” omgeving. Backoffice medewerkers kunnen via hetzelfde webportaal de klantportfolio’s beheren of middels een uitgebreidere .NET smart cliënt.

Via webservices kunnen verzekeraars en externe ontwikkelteams functionaliteit van het platform in hun eigen omgeving integreren. Zo kan men bijvoorbeeld bij het boeken van een reis d.m.v. één extra vinkje een reisverzekering aanvragen via een externe site, of een eigen “mijn omgeving” integreren binnen een bestaand portaal.

Verne heeft als doelstelling om de verzekeringsbranche zo ver mogelijk te automatiseren. Een onderdeel van het platform doet elke dag 160.000 aanvragen in 26 landen per dag, inclusief de financiële verwerking hiervan in alle verschillende landen. Onlangs heeft Verne haar aandachtsgebied binnen de financiële diensten verbreed door de samenwerking met het Crowdfunding platform Symbid.

Verne is een kleine organisatie met 10 medewerkers. Het bedrijf kent een platte organisatiestructuur die past bij het kleine aantal werknemers. De betrokkenen zijn allen goed aanspreekbaar. Degene die het meeste met de oplossing zal moeten werken deelt het kantoor met de uitvoerder van dit project (Twan van Buuren). De directe betrokkenen en hun rollen zullen worden uiteengezet in het hoofdstuk “Projectorganisatie & begeleiding binnen Verne”.

Probleemstelling Verne

Context

Onlangs heeft Verne een nieuw product geïntroduceerd samen met een externe ontwikkelpartij (MyCubes). Het product genaamd: “FormsEngine” heeft als doel de integratie van externe partijen met het Verne platform te versimpelen waardoor activiteiten als het premieberekeningen, aanvragen en mutaties door externe partijen sterk vereenvoudigd wordt.

FormsEngine als product is tot stand gekomen uit de probleemstelling dat externe partijen moeite hebben met het koppelen aan de externe webservices (API) van Verne. Er is vaak een gebrek aan verzekeringsspecifieke domeinkennis bij externe ontwikkelpartijen waardoor de doorlooptijd en kosten hoog zijn bij een implementatie. Tevens is de “leercurve” van de Verne API dermate hoog dat Verne veel support moet leveren aan partijen die aansluiten.

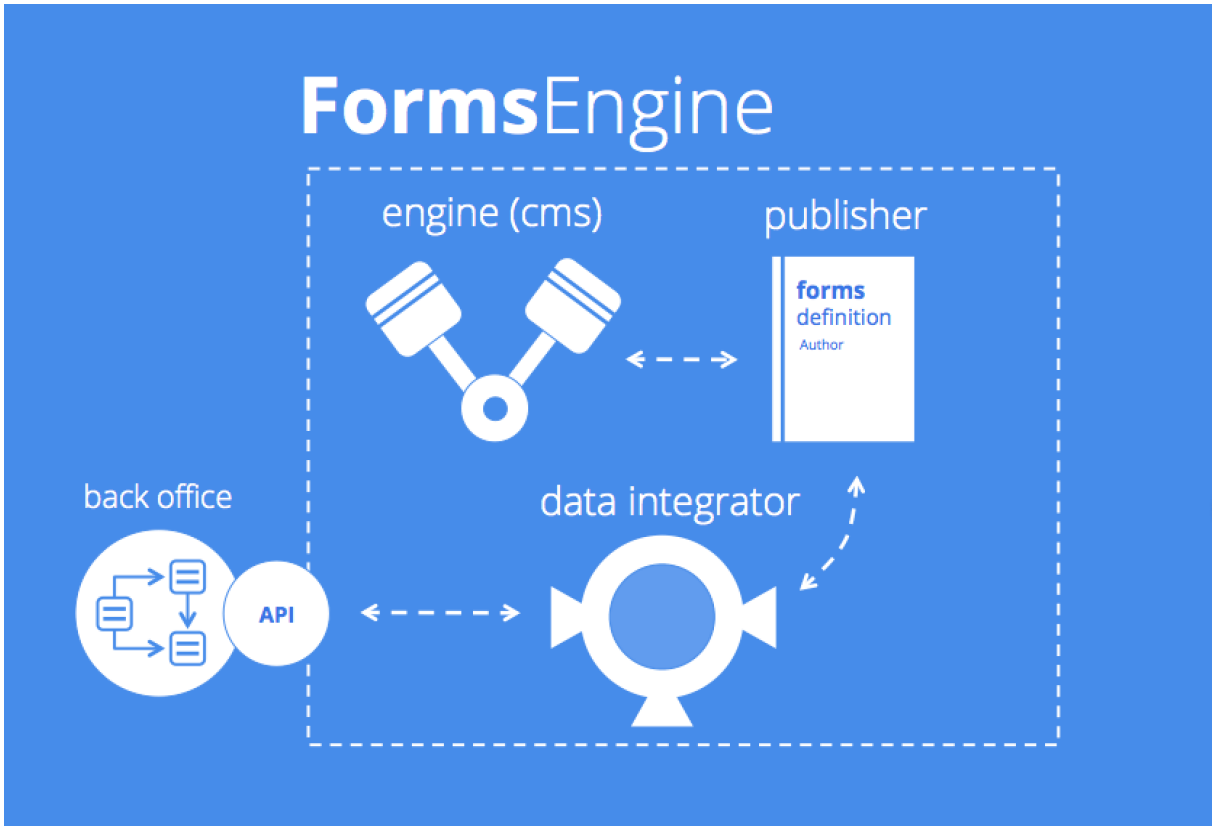
Met de introductie van Responsive design, A/B testen, Google Analytics en de wens van elke verzekeraar om een eigen “styling” te kunnen aanhouden volstaat een i-Frame oplossing niet. Maatwerkontwikkeling door externe partijen zoals hiervoor aangegeven levert een te lange doorontwikkeling op en vergt te veel ondersteuning en uitleg door Verne.

De FormsEngine wordt ingezet om dit op te lossen en bestaat uit de volgende 3 componenten:

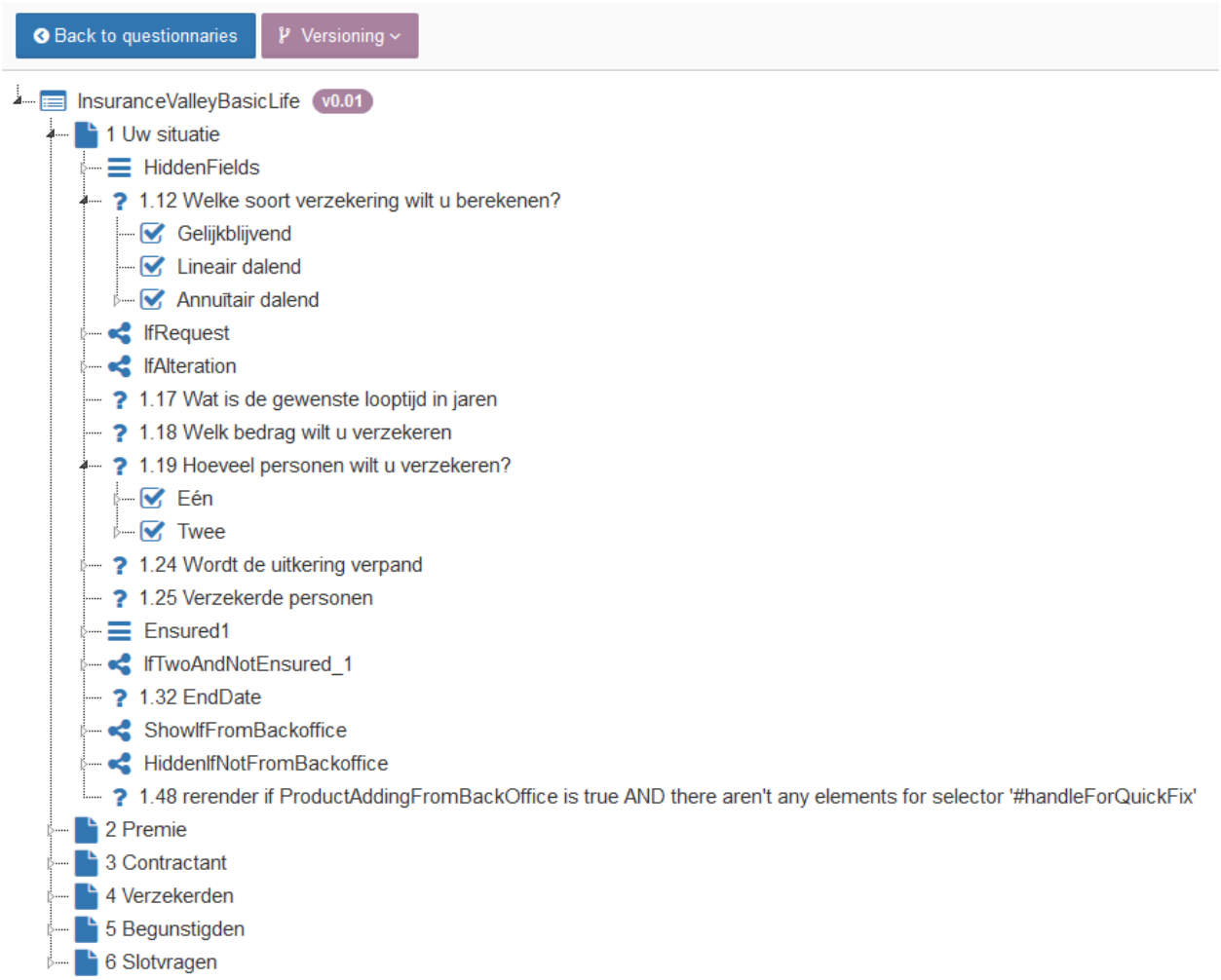
4) Engine (CMS)
5) Publisher

6) Data Integrator (“broker”)

In de onderstaande afbeelding zijn de componenten schematische weergegeven, daaronder wordt dit toegelicht.



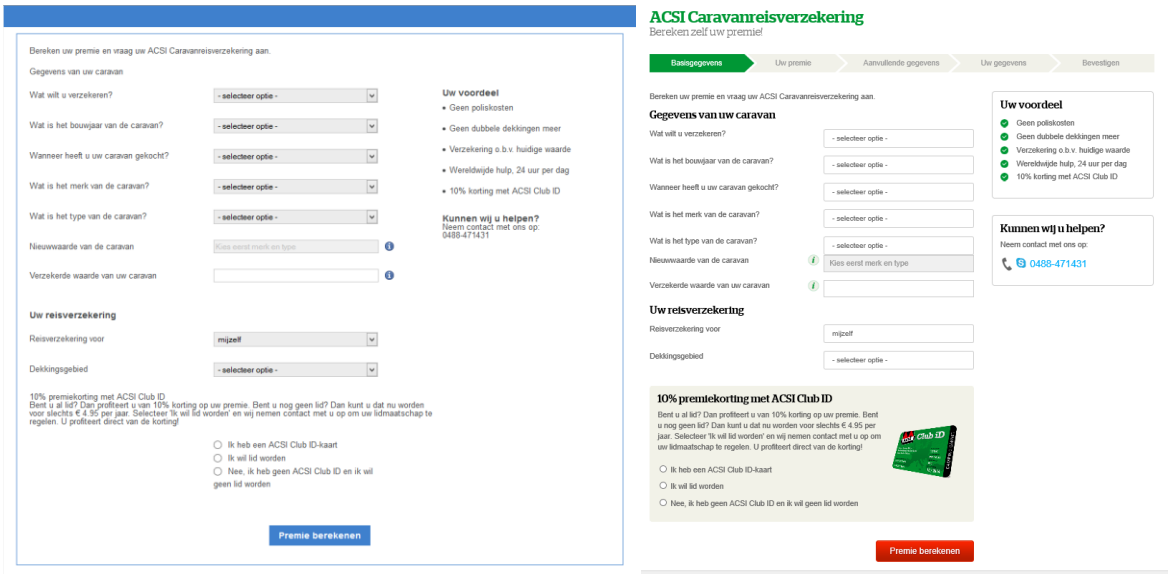
De Engine is een webbased CMS voor het opstellen en beheren van formulieren. Hierin stel je hiërarchisch de vraag-antwoord combinaties op. Dit wordt ook wel het opstellen van de “formdefinitie” genoemd. De formdefinitie bevat de volgorde van vragen, antwoorden, validaties, standaard antwoorden, type vragen (radiobuttons, checkboxes etc.). In onderstaande afbeelding is te zien hoe dit er uit ziet. Deze formulierdefinitie wordt in JSON-formaat opgeslagen. Dit is een zowel voor computers als mensen leesbaar formaat om gegevens te bewaren of over te dragen en wordt vaak gezien als een alternatief voor xml.



Wanneer men klaar is met het opstellen van de formulierdefinitie in de Engine kan het formulier “gepubliceerd” worden naar een publisher.

Een Publisher stelt een formulier beschikbaar voor gebruik en levert daarbij het format en javascript om het formulier naar html om te zetten in de browser van de gebruiker.

De Publisher(s) host(en) de formulierdefinities die vervolgens door een externe integratiepartij gebruikt worden. Een integratiepartij moet voor het gebruiken van het formulier een simpel script opnemen op haar site met een verwijzing naar de publisher URL en het FormsEngine javascript. Dit javascript, welke op de publisher staat, vertaalt de formulierdefinitie in naar HTML middels het gebruik van zogenaamde "handlebar templates"(het voornoemde format om het formulier naar html om te zetten). die elke integratiepartij naar wens kan aanpassen. Deze templates bepalen welke html elementen voor de formulier elementen worden getoond. Zo kan het zelfde formulier er op twee manieren uitzien, zoals te zien in onderstaande screenshots van hetzelfde formulier met een andere template set.



Doordat het javascript van de Publisher de vertaling naar HTML (incl. classes, element-id's etc.) maakt, kan een externe ontwikkelaar dus zelf de styling bepalen van het formulier terwijl de mensen met domeinkennis zelf makkelijk het formulier gecentraliseerd kunnen beheren, aanpassen en publiceren zonder tussenkomst van een externe ontwikkelaar of ontwikkelpartij als de eerste integratie met een formulier voltooid is. Dit scheelt enorm in doorlooptijd, support en kosten.

Bij het versturen van het formulier komen de vraag en antwoordcombinaties in JSON-formaat terug naar de Publisher (ook het versturen van het formulier wordt middels het javascript afgevangen). Aangezien dit JSON-formaat niet door de Verne API of andere API's begrepen wordt, is er een vertaling nodig naar andere API's. Deze vertaalslag gebeurt in de DataIntegrator, ook wel "broker" genoemd.

De broker vertaalt in het geval van Verne de JSON formaat naar SOAP webservices. Hiervoor wordt gebruik gemaakt van (Freemarker) templates. Freemarker is een "template engine", een generieke tool om text te genereren, van html tot broncode of zoals in ons geval xml of JSON.

Probleem

De vertaalslag in de broker naar de Verne API vergt op dit moment nog altijd werk van ontwikkelaars om de juiste velden te "mappen". Gezien er veel formulieren zijn, komen veel mapping vaak voor, zeker daar waar er binnen de FormsEngine gebruik gemaakt wordt van "bibliotheek"-elementen (standaard vragenset die je kunt hergebruiken binnen formulieren). Het maken van datamappings is dus op dit moment voor Verne een bottleneck in het maken van volledig werkende formulieren (met integraties naar de Verne services). De werkzaamheden daarbij bestaan voor een groot deel uit herhaling en zijn tevens fout gevoelig.

Doelstelling opdrachtgever

Door het grotendeels automatiseren van datamappings wil Verne de kwaliteit verhogen van formulier opleveringen en voornamelijk de tijd om tot een volledig werkende en geïntegreerd formulier te komen, sterk verkorten.

Binnen de scope van de afstudeeropdracht is voor de opdrachtgever de doelstelling dat de basis wordt gelegd wat betreft architectuur en ontwikkeling om datamappings te kunnen automatiseren. Belangrijk hierin is wel dat de ontwikkelde software intellectueel eigendom is van Verne gezien de software onder licentie aan partners en derde partijen wordt gedistribueerd.

Projectorganisatie & begeleiding binnen Verne

Bij de verschillende projecten wordt bij Verne veelal gewerkt volgens de scrum methodiek. Bij deze agile methodiek wordt op iteratieve wijze aan een product gewerkt. Binnen deze methodiek zijn verschillende rollen gedefinieerd. Daarvan is de “product owner” eindverantwoordelijke voor (het bepalen van) de functionaliteiten.

Hary Selman is “product owner” van FormsEngine binnen Verne en verantwoordelijk voor de begeleiding en aansturing van zowel het afstudeerproject als de generieke doorontwikkeling van het platform.

Hary Selman zal direct verantwoordelijk zijn voor Twan van Buren maar wordt additioneel technisch begeleid door lead ontwikkelaar van het platform: John Li.

Hary is afgestudeerd als Ir. Technische Bestuurskunde aan de TU Delft en heeft o.a. Gewerkt als Technology Architect bij Accenture. Hij is de grondlegger achter het concept van de FormsEngine en kan gezien worden als de “product owner”. Hary vervult binnen Verne zowel de rol van Solution Architect als Technisch Projectleider. Met Hary zal naast de dagelijkse scrum ook dagelijks contact zijn over de specifieke voortgang van het project.

John Li is Lead Developer & Architect (en eigenaar) bij MyCubes. MyCubes voert een deel van de door-ontwikkeling van het Verne platform uit. Hij is tevens de lead developer achter de FormsEngine en is daarnaast veelal werkzaam als integratie architect bij overheid & zorg organisaties (o.a. Logius). Een dag in de twee weken zal de student op locatie bij John werken om optimaal te kunnen afstemmen en code reviews te kunnen doen. Verder Is hij continu aanspreekbaar via e-mail, telefoon en Skype.

- Hary Selman begeleidt op:
    1. Aanpak, scope, proces en deliverables
    2. Overall Solution Architectuur
        1. Technisch Ontwerp / Architectuur
        2. Functionele bijdrage voor gebruiker / organisatie
        3. Beheerbaarheid / kosten van de oplossing
    3. Afhankelijkheden met externe partijen in ontwikkeling

- John Li begeleidt op:
    1. Inhoudelijke Technische ontwerp & Architectuur
    2. Applicatie Ontwikkeling & code reviews
    3. Integratie van opdracht binnen huidige applicatielandschap

Contactgegevens begeleiding		
	Hary Selman	John Li
e-mail	hary@verne.nu	john@mycubes.nl
telefoon	+31 (0) 6 2000 6102	+31 (0) 6 2151 3273
Skype	hary.selman	

Project

Verne levert een platform waarin verzekeringen geadministreerd worden. Daarbij word gebruik gemaakt van de FormsEngine om web formulieren te maken voor aanvragen en wijzigingen die door derden op hun website kunnen worden opgenomen. In dit proces ligt er een bottleneck zowel qua doorlooptijd als foutgevoeligheid bij de mappingtemplates die bepalen hoe de data uit de formulieren in de webservices worden ingevuld. Om de kosten en fouten te beperken wil Verne deze mappings voor een zo groot mogelijk deel geautomatiseerd genereren.

In dit hoofdstuk wordt uitgewerkt hoe het zoeken naar de oplossing vorm krijgt als afstudeerproject.

Onderzoeksvragen

Om het probleem van Verne op te lossen zal de volgende hoofdvraag beantwoord moeten worden, het antwoord op die vraag zal worden samengesteld uit de antwoorden op de deelvragen.

Hoofdvraag

Op welke wijze kan Verne de bestaande koppeling van verschillende formulieren aan webservices automatiseren om tijd en kosten te besparen?

Deelvragen

7. Hoeveel tijd kan Verne besparen door de mappings van de koppeling te laten genereren?
    a. Hoeveel tijd wordt er nu besteed aan de koppeling?
    b. Hoeveel tijd wordt er besteed aan de koppeling bij gebruik van de tool?
8. Welke functionaliteiten zijn er bij andere “data-mapping” tools die ook terug verwacht mogen/moeten worden bij uiteindelijke oplossing?
9. Hoe past het automatiseren van de datamappings binnen de huidige architectuur van FormsEngine, Publisher & de Broker?
10. Welke gegevens zijn nodig om datamappings te kunnen automatiseren?
11. Hoe wordt een mapping in de software bepaald?
12. Welke wijzigingen zijn er binnen de applicatielandschap van de FormsEngine nodig (FormsEngine/Publisher/Broker)?

Type opdracht en Methodieken

Bij het beantwoorden van de (deel-)vragen kunnen verschillende typen werkzaamheden worden onderscheiden. Deze opdrachttypes vragen allen om een eigen aanpak. De verschillende type opdrachten met hun aanpak zijn:

1. Effectmeting: Tijd die nodig is om een koppeling te maken.
   Deze meting wordt gebruikt om deelvraag 1 te beantwoorden. De volgende stappen zullen hiervoor doorlopen worden:
    a. Door medewerkers wordt geregistreerd hoeveel tijd ze aan welk onderdeel van welk project besteden. (deelvraag 1a)
    b. Door deze data te filteren kan per project gezien worden hoeveel tijd aan de data-koppeling is besteed. (deelvraag 1b)
    c. Zonder en met gebruik van de te ontwikkelen tool te vergelijken kan wordt duidelijk hoe veel tijd bespaard wordt.
2. Korte inventarisatie bestaande data-mapping tools:
   Dit deel wordt uitgevoerd als een literatuuronderzoek of deskresearch en zal antwoord geven op deelvraag 2. Daarbij worden de volgende stappen doorlopen:
    a. Inventarisatie van de te onderzoeken tools door interviews en gebruik van

zoekmachines
   b. Officiële documentatie van de relevante producten beschouwen
   c. Documenteren hoe deze producten zich verhouden tot de te ontwikkelen tool
3. Opstellen architectuur van data-mapper binnen bestaande FormsEngine landschap.
   Dit zal gebruikt worden om deelvragen 3 en 6 te beantwoorden.
   De volgende fases zullen doorlopen worden.
   a. Bestaande documentatie over het platform en de daarin gebruikte pakketten doornemen
   b. Overleggen met huidige product owner, architect & lead developer van het platform
   c. UML model invoeren van huidige situatie en beoogde einddoel
   d. Modellen verifiëren bij betrokkenen en waar nodig aanpassen
4. Ontwikkeling van de basis van de tool passend binnen de tijdslijnen van het afstudeerproject
   Hierbij worden de volgende stappen doorlopen:
   a. Verzamelen functionaliteiten, features en componenten op basis van Korte inventarisatie bestaande data-mapping tools, gesprekken met betrokkenen. Hierbij zullen deelvraag 4 en 5 beantwoord worden
   b. Aan het begin van het project wordt de backlog gevuld met alle producten en onderdelen die binnen de scope van het project horen.
   c. Per sprint wordt gekeken welke items worden opgepakt. Na elke sprint volgt een demo.
   d. Er wordt rekening gehouden met de planning van de deliverables binnen het project en de tijdslijnen van het afstudeerproject (zie planning in dit document)

Ontwikkelframeworks

Uit de kick-off meeting is gebleken dat het voor de gebruiker het handigst is als de oplossing als plugin op de Publisher kan worden toegepast. Om dit te faciliteren is het handig om in hetzelfde ontwikkelframework te werken, zeker gezien deze is ingericht op het toevoegen van functionaliteitenmiddels plugins in een applicatie.

Dit houdt concrete in dat de oplossing zal worden ontwikkelt als Grails applicatie (plugin). Grails is een op spring gebaseerd applicatie framework, waarbinnen hoofdzakelijk in de taal Groovy wordt geprogrammeerd, het is echter ook mogelijk met Java te programmeren of Java componenten te gebruiken. (Groovy werkt op Java VM). Daarbij zal Redis als database gebruikt worden.

Deliverables

Er wordt gezocht naar een oplossing die binnen de periode van het afstudeerproject geïmplementeerd kan worden.

Er wordt verwacht dat aan het einde van het project de volgende zaken zijn opgeleverd:

1) Resultaatmeting: een vergelijking van de doorlooptijden voor en na de implementatie van de tool.
2) Onderzoeksresultaten van bondig onderzoek naar bestaande data-mapping tools
3) Architectuur data-mapper binnen het huidige applicatielandschap
4) Implementatie van basisversie van de data-mapper
5) Scriptie

Kwaliteitseisen deliverables

De kwaliteitseisen die worden gesteld aan de deliverables vanuit de opdrachtgevers zijn:

Ad.1) Overzicht van de features wordt in een matrix vergelijking opgenomen met daarin tevens volgens MosCow methode gecategoriseerde waardoordeel vanuit Verne over de relevantie van de features.

Ad.2) Een voor alle betrokkenen duidelijke architectuurplaat volgens de UML standaard met heldere omschrijving van de interfaces en functies die gekoppeld zijn aan de FormsEngine landschap.

Ad.3) Applicatieontwikkeling volgens gangbare design patterns en principes zoals DRY (Don't Repeat Yourself). Tevens verwacht de opdrachtgever comments volgens geldige normen in de code die de functies en logica omschrijven.

Ad. 4) De resultaatmeting moet een analyse van de urenregistratie data doen, waaruit blijkt wat het voordeel is dat de tool oplevert.

Ad. 5) Scriptie voldoet aan de eisen zoals gesteld in het beoordelingsformulier uit de afstudeerleidraad (Hogeschool Utrecht Faculteit Natuur en Techniek Institute for ICT, 2014).

De opdrachtgever verwacht tevens een proactieve houding waar het gaat om het verkrijgen van feedback op deelresultaten. De projectorganisatie is daarop ook ingericht.

Afbakening project

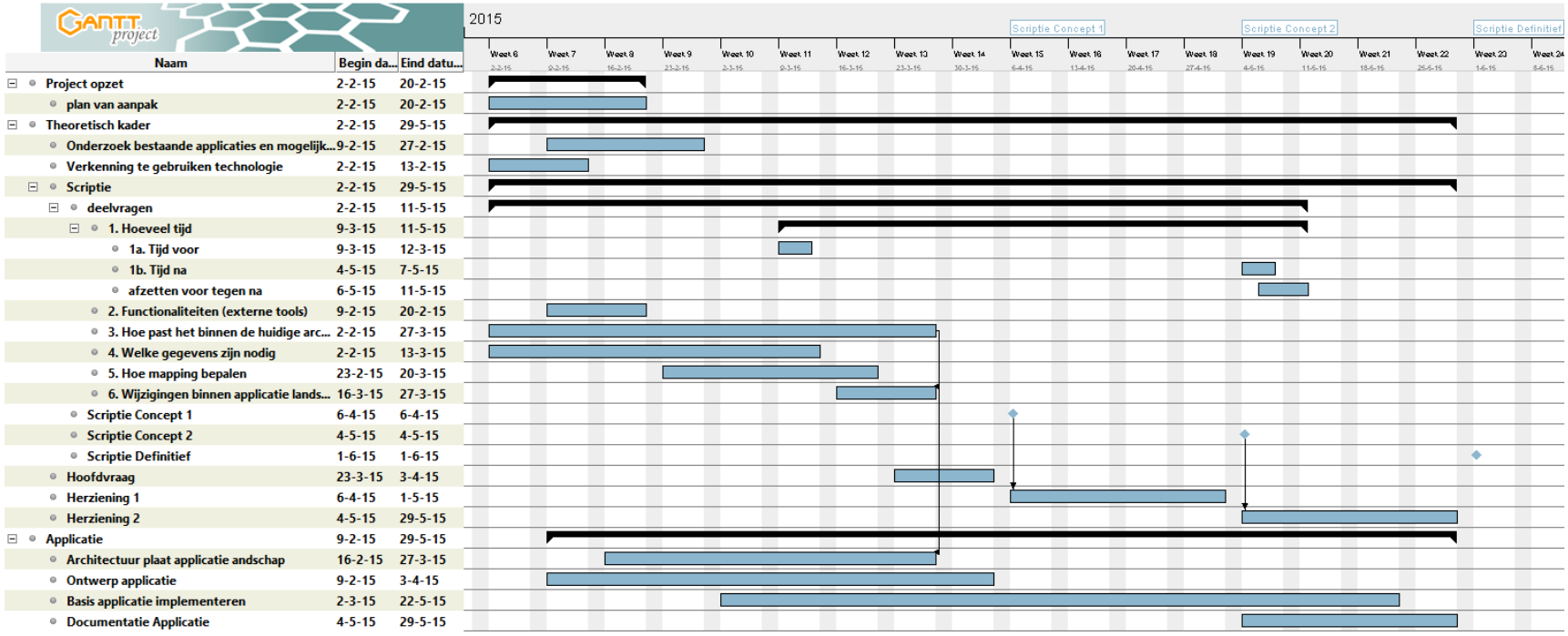
Het verzamelen en definiëren van de exacte functionaliteiten die de oplossing bevat gebeurd bij het beantwoorden van deelvragen twee, drie, vier en vijf. Het is daarom niet mogelijk op dit punt al per functionaliteit aan te geven welke wel of niet binnen de scope vallen. Dit zal bij het beantwoorden van de betreffende deelvragen aangegeven worden middels de MoSCoW methode.

Bij de beoordeling van de MosCow scores wordt rekening gehouden met de volgende factoren:

- Het afstudeerproject valt binnen een groter project met een bestaand applicatielandschap.
    - De ontwikkeling van de data mapper (na finalisering van de architectuur deliverables) is beperkt tot een standalone module / plugin.
    - Huidige applicatie architectuur ondersteunt de ontwikkeling van standalone modules / plugins.
- Van alle input en output informatiestromen van de data-mapper moet worden aangenomen dat deze tijdelijk handmatig moeten worden gekoppeld tussen systemen. Dit om het project te doen passen binnen de tijdslijnen van het afstudeerproject. Er zal dan ook geen sprake zijn van dataintegratie tussen de FormsEngine componenten en de data-mapper. Indien de voortgang voorspoediger loopt dan gepland, zou dit (gedeeltelijk) binnen de scope gehaald kunnen worden.

Planning

De grove planning is weergegeven in onderstaande Gantt chart.



Naam	Begin da...	Eind datu...
Project opzet	2-2-15	20-2-15
plan van aanpak	2-2-15	20-2-15
Theoretisch kader	2-2-15	29-5-15
Onderzoek bestaande applicaties en mogelijk...	9-2-15	27-2-15
Verkenning te gebruiken technologie	2-2-15	13-2-15
Scriptie	2-2-15	29-5-15
deelvragen	2-2-15	11-5-15
1. Hoeveel tijd	9-3-15	11-5-15
1a. Tijd voor	9-3-15	12-3-15
1b. Tijd na	4-5-15	7-5-15
afzetten voor tegen na	6-5-15	11-5-15
2. Functionaliteiten (externe tools)	9-2-15	20-2-15
3. Hoe past het binnen de huidige arc...	2-2-15	27-3-15
4. Welke gegevens zijn nodig	2-2-15	13-3-15
5. Hoe mapping bepalen	23-2-15	20-3-15
6. Wijzigingen binnen applicatie lands...	16-3-15	27-3-15
Scriptie Concept 1	6-4-15	6-4-15
Scriptie Concept 2	4-5-15	4-5-15
Scriptie Definitief	1-6-15	1-6-15
Hoofdvraag	23-3-15	3-4-15
Herziening 1	6-4-15	1-5-15
Herziening 2	4-5-15	29-5-15
Applicatie	9-2-15	29-5-15
Architectuur plaat applicatie andschap	16-2-15	27-3-15
Ontwerp applicatie	9-2-15	3-4-15
Basis applicatie implementeren	2-3-15	22-5-15
Documentatie Applicatie	4-5-15	29-5-15

Risico's

In onderstaande tabel worden de voornaamste risico's voor dit project beschreven.

Beschrijving risico	Impact	Kans (%)	Mitigerende activiteiten
Functionaliteiten die moeten worden ontwikkeld aan de FormsEngine of Publisher applicatie om functionaliteiten binnen de datamapper mogelijk te maken.	middel	laag/ middel	Dit kan ondervangen worden door handmatige in-/uitvoer van data, zoals bij de scope van het project aangegeven is. Tevens wordt het project als een aparte module / plugin beschreven waardoor afhankelijkheid van externe ontwikkeling geminimaliseerd wordt.
Kortstondige ziekte	laag	laag	Indien het een lichte ziekte betreft zal dit geen grote vertraging betekenen omdat alle betrokkenen de middelen hebben om vanuit huis contact te onderhouden en alle werkzaamheden te kunnen uitvoeren.
Langdurige, ernstige ziekte	Hoog	Zeer laag	Bij een ernstige, langdurige ziekte, zal een collega als vervanging dienen. Van de betrokkenen bij het project zijn de vervangende collega's al benoemd.
Het werken volgens scrum methodiek introduceert grote flexibiliteit in het inrichten van het project, nadeel hier van is dat dit ook onzekerheid inhoud.	Middel	Middel	Om te waarborgen dat het project wel binnen het kader van het afstudeerproject blijft, wordt de backlog aan het begin van het project gevuld. Vervolgens wordt per sprint gekeken welke items op dat moment aan bod komen.
Miscommunicatie	middel	Middel	Er is dagelijks overleg tussen de bedrijfsbegeleider en de uitvoerder om te overleggen wat de stand van zaken is en welke betrokkenen geïnformeerd of geraadpleegd moeten worden. Tevens zorgt een demo aan het einde van elke sprint ervoor dat geen langdurige periode kan zijn waarop er verschillen van inzicht kunnen zijn.

Voorlopige Bronnenlijst

altova. (sd). Opgehaald van altova: http://www.altova.com

Beckwith, B. (2013). *Programming Grails (Best practices for experienced Grails developers).* Sebastopol: oreilly.

Berg, K. P. (2007, 11 12). *client-side-wsdl-processing-with-groovy-and-gant.html*. Opgehaald van http://www.javaworld.com: http://www.javaworld.com/article/2077790/open-source-tools/client-side-wsdl-processing-with-groovy-and-gant.html

Buuren, T. v. (2015, 1 30). Verslag project kickoff automatische koppeling formulieren aan webservices 30-01-2015 bij MyCubes. Schiphol-Rijk, Noord Holland, Nederland.

easywsdl. (2015, 2 11). Opgehaald van easywsdl.ow2.org/: http://easywsdl.ow2.org/index.html

FreeMarker. (sd). *FreeMarker Java Template Engine - Overview*. Opgehaald van freemarker.org/: http://freemarker.org/

grails.org. (2015, 02 01). *grails.org/learn*. Opgehaald van grails.org/learn: https://grails.org/learn

Hogeschool Utrecht Faculteit Natuur en Techniek Institute for ICT. (2014). *AFSTUDEERLEIDRAAD BUSINESS IT & MANAGEMENT SOFTWARE ENGINEERING INFORMATION ENGINEERING SYSTEM AND NETWORK ENGINEERING TECHNISCHE INFORMATICA* . Utrecht: Hogeschool Utrecht.

jackson-module-jsonSchema. (2014, 2 8). Opgehaald van https://github.com/FasterXML/jackson-module-jsonSchema: https://github.com/FasterXML/jackson-module-jsonSchema

JacksonTreeModel. (2015, 02 10). Opgehaald van http://wiki.fasterxml.com: http://wiki.fasterxml.com/JacksonTreeModel

jax-ws RuntimeWSDLParser.html. (sd). Opgehaald van https://jax-ws.java.net: https://jax-ws.java.net/nonav/2.2.7/javadocs/com/sun/xml/ws/wsdl/parser/RuntimeWSDLParser.html

parse-wsdl-java-api.htm. (2015, 2 12). Opgehaald van membrane-soa.org: http://www.membrane-soa.org/parse-wsdl-java-api.htm

Pavlovic, F. (sd). *jamper*. Opgehaald van sourceforge.net: http://jamper.sourceforge.net/

redis.io. (2015, 02 01). *redis.io/documentation*. Opgehaald van redis.io/documentation: http://redis.io/documentation

woden. (2015, 2 1). Opgehaald van ws.apache.org: http://ws.apache.org/woden/

www.stylusstudio.com. (sd). *stylusstudio xml_mapper*. Opgehaald van stylusstudio.com: www.stylusstudio.com/xml_mapper.html

Bijlages

Contract afstudeeropdracht

Contract afstudeeropdracht
Institute for ICT
Nijenoord 1, 3552 AS, UTRECHT

NB: Dit contract dient te worden opgenomen als vast onderdeel van het plan van aanpak

Datum :	19 maart 2015	
Naam student :	Twan van Buuren	
Opleiding :	Informatica (SE) voltijd	Variant: voltijd / deeltijd / duaal
Adres student :	Binnenstraat 3d	
Postcode / Woonplaats student :	4117gr Erichem	
Studentnummer :	1604889	
Telefoonnummer privé :	0623375928	
E-mailadres :	twan@verne.nu	
Naam bedrijf (afstuderen) :	Verne	
Adres bedrijf :	Godfried Bomansstraat 2	
Postcode / Woonplaats bedrijf :	4103 WR Culemborg	
Naam bedrijfsbegeleider :	Hary Selman	
Telefoonnummer bedrijfsbegeleider :	+31 (0) 6 2000 6102	
E-mailadres bedrijfsbegeleider :	hary@verne.nu	
Beoogde datum van afstuderen :	Juni 2015	*maand en jaar invullen*
Geheimhouding geaccordeerd door HU op :	NVT	*indien van toepassing*

Ondergetekenden verklaren akkoord te gaan met de inhoud van aangehecht plan van aanpal

Handtekeningen

Student :	Twan van Buuren
Docentbegeleider :	Rory Sie
Bedrijfsbegeleider[9] :	Hary Selman

[9] Door ondertekening van dit formulier verklaart de bedrijfsbegeleider (en eventuele mede-begeleiders) over voldoende kennis te beschikken, op minimaal HBO-niveau, om de afstudeerder te begeleiden.

13.2 Evaluatie van eigen functioneren

Binnen dit project heb ik leren werken met nieuwe technologieën, waar ik zelf nog geen ervaring mee had. Zo heb ik voor het eerst ontwikkeld in het Grails framework met Groovy als programmeertaal en Redis als database.

Ik vond het erg leerzaam om hiermee te werken. In het begin kostte het extra tijd om de juiste werkwijze te vinden en me aan te passen aan een ander idioom dan ik gewend was. Na verloop van tijd lukte het steeds beter de functionaliteiten die de omgeving en technieken bieden te benutten.

Bij mijn eerste kennismaking heb ik meerdere keren aanpassingen gedaan waardoor de hele applicatie vastliep en ik de oorzaak pas na lang zoeken kon vinden. Al doende heb ik wel de werking van en filosofie achter het framework veel beter leren kennen. Ook heb ik geleerd breder te kijken naar de impact van wijzigingen in een omgeving die is gebaseerd op een standaard werkwijze.

Een ander voorbeeld wat daarbij aansluit is het werken volgens ‘test driven development’. Grails bied hier ‘out of the box’ zeer goede ondersteuning voor. In het begin was het erg onwennig en tijdrovend om steeds vanuit tests te beginnen, maar toen ik me de tools en werkwijze meer eigen had gemaakt ging het duidelijk zijn vruchten afwerpen. Zo kon ik bijvoorbeeld, al voordat ik een nieuwe functie af, had fouten die door die aanpassing ontstonden herkennen en corrigeren.

Bij het me eigen maken van de nieuwe omgeving heb ik veel profijt gehad van de korte maar gerichte aanwijzingen van John Li, die me daarmee steeds op het goede pad hielp. In zijn algemeenheid ben ik erg positief over de communicatie bij dit project. Door regelmatig korte vergaderingen te houden werd weinig tijd verspild en was iedereen goed op de hoogte van de stand van zaken. Als er toch een misverstand was omdat iemand een stukje informatie had gemist, werd dit steeds snel opgehelderd.

Mede daardoor verliep het verzamelen van de functionaliteiten en opstellen van de beoogde architectuur erg soepel. We hebben hier minder aan hoeven te passen in de loop van het project dan ik verwacht had. De planning en methodiek boden hier wel ruimte voor, maar deze stabiliteit gaf een zeker gevoel over het project.

Hoewel de planning hier op was afgestemd, namen de documentatie delen van het project meer tijd in beslag dan gevoelsmatig verwacht. Het is mede daardoor gelukt om de doelstellingen en resultaten volgens het plan van aanpak te bewerkstelligen. Ik kijk terug op een leerzame periode, waarvan ik de volgende punten zeker zal meenemen in toekomstige projecten:

- Een open oog houden voor nieuwe technieken en niet bang zijn deze in te voeren.
- Waar mogelijk vooraf testcases te definiëren en deze het liefst meteen te automatiseren alvorens nieuwe functionaliteit te creëren.
- Regelmatig korte overleggen houden om iedereen op de hoogte te houden, dit bespaart uiteindelijk veel tijd.
- Documentatie onderdelen extra ruim inplannen, dit kost vaak meer tijd dan op het eerste gezicht lijkt, bijvoorbeeld omdat er veel uitzoekwerk bij komt kijken.

13.3 Beoordeling afstudeeropdracht door de bedrijfsbegeleider

FACULTEIT NATUUR & TECHNIEK

Institute for ICT

Beoordeling afstudeeropdracht door de bedrijfsbegeleider(s)

Naam student: Twan van Buuren

Opleiding: Informatica (SE)

Variant: voltijd

Opdracht uitgevoerd bij: Verne

Onder leiding van: Hary Selman

Eerste examinator: Linda Terlouw

De bedrijfsbegeleider(s) wordt gevraagd voor de aspecten A en D een gemotiveerd oordeel (geen cijfer) te geven. In de kaders is aangegeven welke punten daarbij in overweging genomen kunnen worden. Bij de aspecten B en C wordt gevraagd om in de rechter kolom per criterium d.m.v. O (onvoldoende), V (voldoende), G (goed) of Z (zeer goed) een oordeel te geven, rekening houdend met de aangegeven indicatoren.

A. Aanpak

Het is erg prettig om met Twan samen te werken. Hij is pro-actief, goed gestructureerd en sterk in een discussie omdat hij zijn betoog met goede en heldere argumenten uiteen weet te zetten. Twan is zowel een denker als een doener; dat zijn eigenschappen die lastig te vinden zijn in een persoon. Twan is daarom zowel functioneel als technisch goed in ontwerp en het bepalen van de impact op hetgeen waarmee hij bezig is, en haalt tevens zijn deliverables binnen de de besproken tijd. Ook organisatorisch is Twan sterk en weet hij welke processen verbeterd moeten worden om zijn werk goed uit te kunnen voeren.

De eigenschappen van Twan zijn ook in de aanpak van zijn afstudeerproject goed terug te vinden. In de plan van aanpak heeft Twan helder de probleemstelling van Verne vertaald naar hoofd en deelvragen. De gebruikte methodieken en technieken zijn een logische gevolgtrekking van de inhoud van deelvragen.

Twan heeft het afstudeerproject in overleg met de opdrachtgever goed weten te passen binnen de planning en ontwikkelprocessen van Verne. Deze zijn tevens door Twan ook goed afgestemd met de tweede (technische) externe bedrijfsbegeleider (John Li).

Twan heeft bij het tot stand komen van de plan van aanpak goed met de opdrachtgever en 2e externe bedrijfsbegeleider overlegd om zowel zijn deliverables richting de hogeschool van Utrecht als dat van het project van Verne te kunnen waarborgen. Twan heeft daarbij een gezonde belans gevonden tussen zelfstandig opereren en afstemmen van de plan van aanpak met de opdrachtgever.

Z.O.Z.

Dit formulier dient door de afstudeerder tegelijk met de scriptie te worden ingeleverd.

Bovendien dient een ingescande versie per e-mail te worden toegestuurd aan de twee examinatoren.

B. Uitvoering

Object	Criterium	Indicatoren	O/V/ G/Z
Realiseren (deel)resultaten	Vakkundig	De gekozen ontwerpmethode en de eventuele realisatie is zo uitgevoerd dat deze: • effectief was, dus heeft geleid tot het gewenste (deel)resultaat; • efficiënt was, dus met minimaal mogelijke inspanning; • toelaatbaar was, dus voldeed aan in het vakgebied gebruikelijke gedragsregels.	Z
	Creatief	• Door exploratie en inventiviteit zijn alternatieve (deel) oplossingen bedacht. • De voorgestelde oplossingen getuigen van: originaliteit, eigenheid, out of the box aanpak, ideeën kunnen genereren, gevoel voor innovatie. • Tussen de oplossingen is een bewuste keuze gemaakt of is doorgezocht naar betere oplossingen.	G
	Logisch	Redeneringen zijn: • compleet; • taal- en rekenkundig kloppend; • vrij van onzakelijke argumenten en gegoochel met betekenissen.	Z
	Controleerbaar	• Compleet. • Achteraf nog gedetailleerd te volgen. • Transparant, duidelijk gecommuniceerd.	G
Beantwoorden (deel)vragen	Vakkundig	De gekozen onderzoeksmethode is zo uitgevoerd dat deze: • effectief was, dus leidt tot beantwoording van de (deel)vraag; • efficiënt was, dus met minimaal mogelijke inspanning; • toelaatbaar waren, dus voldeden aan de Gedragscode praktijkgericht onderzoek voor het HBO.	G
	Betrouwbaar	De uitvoering van het onderzoek is niet ongewenst beïnvloed door de toevallige: • voorgeschiedenis of gesteldheid van de onderzoeker; • onvolkomenheden in instrument of procedure; • context waarin of het toevallige tijdstip waarop het onderzoek plaatsvond.	G
	Logisch	De keten van bewijsvoering is: • compleet; • taal- en rekenkundig kloppend; • vrij van onzakelijke argumenten en gegoochel met betekenissen.	G
	Controleerbaar	• Compleet. • Achteraf nog gedetailleerd te volgen. • Transparant, duidelijk gecommuniceerd.	G
Aanvullingen theoretisch kader	Relevant	• De gehanteerde bergrippen, modellen en methoden zijn effectief en voor collega-professionals inzichtelijk beschreven.	G
	Verankerd	• De gehanteerde begrippen, modellen en methoden zijn aantoonbaar gebaseerd op actuele schriftelijke of internetbronnen die gebaseerd zijn op verifieerbare externe bronnen (voornamelijk wetenschappelijk).	G
Persoonlijk leiderschap	Initiatiefrijk	• Neemt initiatief om de gestelde doelen te bereiken en kiest afhankelijk van de situatie een leidinggevende rol. • Is zich bewust van de verschillende stijlen, conflictmodellen en onderhandelingstactieken. • Kan anderen met argumenten goed overtuigen. • Durft beslissingen te nemen in onzekere situaties en neemt weloverwogen risico's.	Z
Samenwerking	Cooperatief	• Neemt het initiatief tot samenwerking met anderen en vraagt uit zichzelf hoe hij/zij hulp kan bieden in het team. • Vraagt om inbreng van de teamleden, wisselt informatie, kennis en ideeën uit. • Werkt verder met de inbreng van anderen. • Creëert een omgeving die het meest gunstig is voor zichzelf en het team.	G
Communicatie	Transparant	• Spreekt en schrijft vloeiend en correct Nederlands op hoog niveau (C1)[10]. • Spreekt en schrijft indien van toepassing effectief Engels (B2).	V
Werken volgens plan van aanpak	Betrouwbaar	Uitgangspunt is dat efficiënt en effectief gewerkt wordt volgens planning: • afwijkingen worden onderbouwd en verantwoord; • afwijkingen zijn goedgekeurd door de opdrachtgever en de docentbegeleider.	G
	Controleerbaar	• De voortgang van het werk wordt inzichtelijk gemaakt voor de betrokkenen.	G

10 http://taalunieversum.org/onderwijs/gemeenschappelijk_europees_referentiekader/3/3/

C. Opgeleverde eindresultaten

Product	Criterium	Indicatoren	O/V/ G/Z
Beroepsproducten (kunnen ook diensten zijn)	Adequaat	• Voldoet aan de in het plan van aanpak afgesproken criteria (rekening houdend met hierover formeel afgesproken wijzigingen) • Voor zover in plan van aanpak geen criteria zijn afgesproken gelden de volgende criteria: ▪ levert daadwerkelijk een oplossing binnen doel van de opdracht; ▪ voldoet aan normen en richtlijnen uit het vakgebied.; ▪ voldoet aan normen van de wetgever; ▪ duurzaam.	Z
	Valide	• Geaccepteerd door de opdrachtgever. • Acceptabel voor betrokkenen.	Z
	Creatief	• De voorgestelde aanbevelingen getuigen van: originaliteit, eigenheid, out of the box aanpak, onafhankelijk denken, ideeën kunnen genereren, gevoel voor innovatie.	G
Aanbevelingen (in aanvulling op de afgesproken producten of diensten en indien van toepassing)	Adequaat	• Leveren een daadwerkelijke bijdrage aan doelstellig opdrachtgever. • Voldoen aan normen en richtlijnen uit het vakgebied. • Voldoen aan normen van de wetgever.	G
	Verankerd	• Zijn gebaseerd op de verzamelde gegevens, bevindingen en conclusies.	G
	Haalbaar	De voorgestelde aanbevelingen zijn te realiseren gegeven de: • omgeving waarin ze moeten functioneren; • de mensen die ze moeten uitvoeren; • beschikbare tijd, budget etc.	G
	Creatief	• De voorgestelde aanbevelingen getuigen van: originaliteit, eigenheid, out of the box aanpak, onafhankelijk denken, ideeën kunnen genereren, gevoel voor innovatie.	G
	Ethisch afgewogen	• Verouderen niet te snel. • Zijn weinig belastend voor natuur en milieu.	G
Kennis	Grondig	• Antwoorden op onderzoeksvragen uit het plan van aanpak zijn voor vakgenoten voldoende onderbouwd.	G
	Overdraagbaar	• Antwoorden op onderzoeksvragen uit het plan van aanpak zijn voor vakgenoten helder beschreven. • Hierbij gebruikte begrippen, modellen, methoden en oplossingen zijn gebaseerd op voor vakgenoten verifieerbare bronnen.	G
Impact	Gewenste impact	• Betrokkenen zijn overtuigd van de bruikbaarheid van de resultaten. • Betrokkenen gebruiken de resultaten.	Z

D. Scriptie

Verne is erg te spreken over het eindresultaat van zowel de ontwikkelopdracht als de scriptie van Twan.Twan heeft zeker aan de verwachtingen voldoaan die wij van de datamapping tool hadden binnen de scope van de opdracht. Dat komt deels omdat Twan goed de scope van het project heeft afgestemd met de opdrachtgever alsmede de snelheid en ervaring waarmee Twan ontwikkeld (ook in een voor hem onbekende ontwikkelomgeving / tools en talen). Twan heeft tevens het testdriven development zeer snel eigen gemaakt. Hierover zijn wij als opdrachtgever zeer te spreken.

Kijkend naar de inhoud van het scriptie document zijn wij ook erg te spreken. Advies om meer in opsommingen te werken om zijn betoog nog duidelijker uiteen te zetten heeft Twan goed opgepakt. Het scriptie document is goed leesbaar en ook individuele onderdelen waarvan verwacht mag worden dat ze zelfstandig leesbaar zijn, zijn goed en helder geschreven.

Twan heeft goed antwoord gegeven op de individuele deelvragen alsmede daardoor de hoofdvraag. Er is door het document heen goede onderbouwing van gemaakte keuzes zoals bijvoorbeeld de keuze voor de design patterns. Ook de aanbevelingen volgen logisch uit de conclusies en is met name leuk te zien dat hij daarbij out of de box denkt wanneer hij ook aanbeveelt de testscripts voor formulieren ook automatisch te laten genereren met de ontwikkelde tool.

Twan heeft een gezonde hoeveelheid aan zelfreflectie en heeft advies van de opdrachtgever (ook technisch) zeer snel opgepakt.

Handtekening bedrijfsbegeleider(s): Hary Selman

13.4 Data meting gewerkte uren

13.4.1 Zonder de Tool

Task ID	Titel	beschrijving / opmerkingen	aantal forms	uren	support/ fout herstel	uren per form	categorie
13851	Forms voor de Kroodle producten maken	4 verschillende forms	4.00	28.00	50%	10.50	simpel
13892	Proteq aanvraag form	Request service ingeregeld (nog geen mutatie, maar dat komt later wanneer er meer duidelijk is over het product E.G. specs	0.50	3.00	50%	9.00	simpel
13878 , 13972	[Bevindingen Scooter formulier] [Bevindingen scooterformulier 10/3/2015]	Bevindingen Scooter formulier	1.00	11.00	50%	16.50	complex bestaand
13636	Allianz: Create travel form EN calculation based on Proteq travel		1.00	6.00	50%	9.00	complex bestaand
13549	Forms voor insurance valley	gedeeltelijk ook ander werk in wi, alleen forms koppeling deel genomen	1.00	20.00	50%	30.00	complex nieuw
NA	Zelf formulier	geen workitem, adhoc werk, wel tijd geklokt	1.00	12.00	50%	18.00	complex bestaand
NA	Twinder reis verzekering formuler	geen workitem, adhoc werk, wel tijd geklokt, met zijn 2e in 2 sessies van 3 uur	1.00	12.00	50%	18.00	complex nieuw
gemiddeld			9.50	92.00	50%	14.53	

Categorie koppeling	aantal koppelingen	Gemiddelde duur inregelen koppeling
simpel	4.50	10.33
complex bestaand	3.00	14.50
complex nieuw	2.00	24.00

13.4.2 Met de Tool

Task ID	Title	beschrijving / operkingen	aantal forms	uren	support/ fout herstel	uren per form	categorie
koppel tbv meting	zipcode	test implementatie	1.00	0.75	25%	0.94	simpel
koppel tbv meting	proteq car	test implementatie	1.00	1.17	25%	1.46	simpel
koppel tbv meting	life	test implementatie	1.00	1.25	25%	1.56	complex bestaand
koppel tbv meting	annualTravel	test implementatie	1.00	1.08	25%	1.35	complex bestaand
koppel tbv meting	car	test implementatie	1.00	1.25	25%	1.56	complex bestaand
koppel tbv meting	inboedel	test implementatie	1.00	1.00	25%	1.25	complex bestaand
koppel tbv meting	scooter	test implementatie	1.00	0.92	25%	1.15	complex bestaand

koppel tbv meting	woonhuis	test implementatie	1.00	1.08	25%	1.35	complex bestaand
koppel tbv meting	life	test implementatie	1.00	12.00	25%	15.00	complex nieuw
koppel tbv meting	AnnualTravel	test implementatie	1.00	10.00	25%	12.50	complex nieuw
gemiddeld			10.00	30.50	25%	3.81	

Categorie koppeling	aantal koppelingen	Gemiddelde duur inregelen koppeling
simpel	2.00	1.20
complex bestaand	6.00	1.37
complex nieuw	2.00	13.75

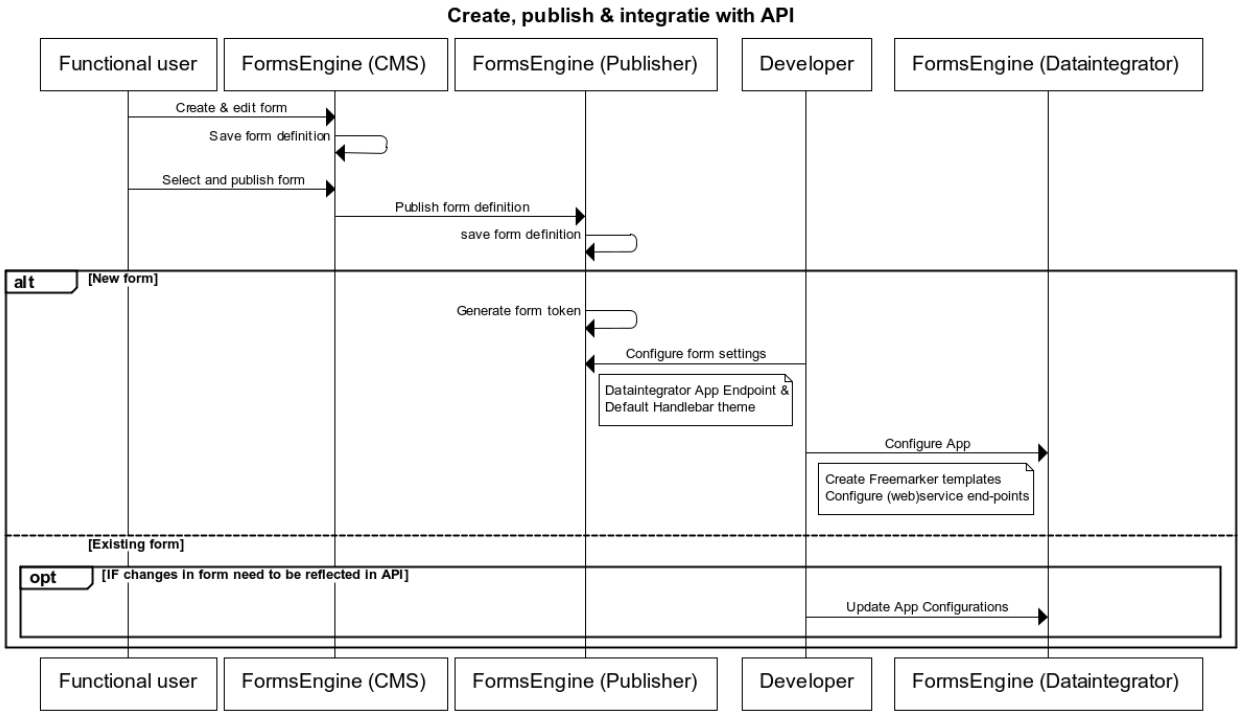
13.5 Uml diagrammen

13.5.1 Interacties

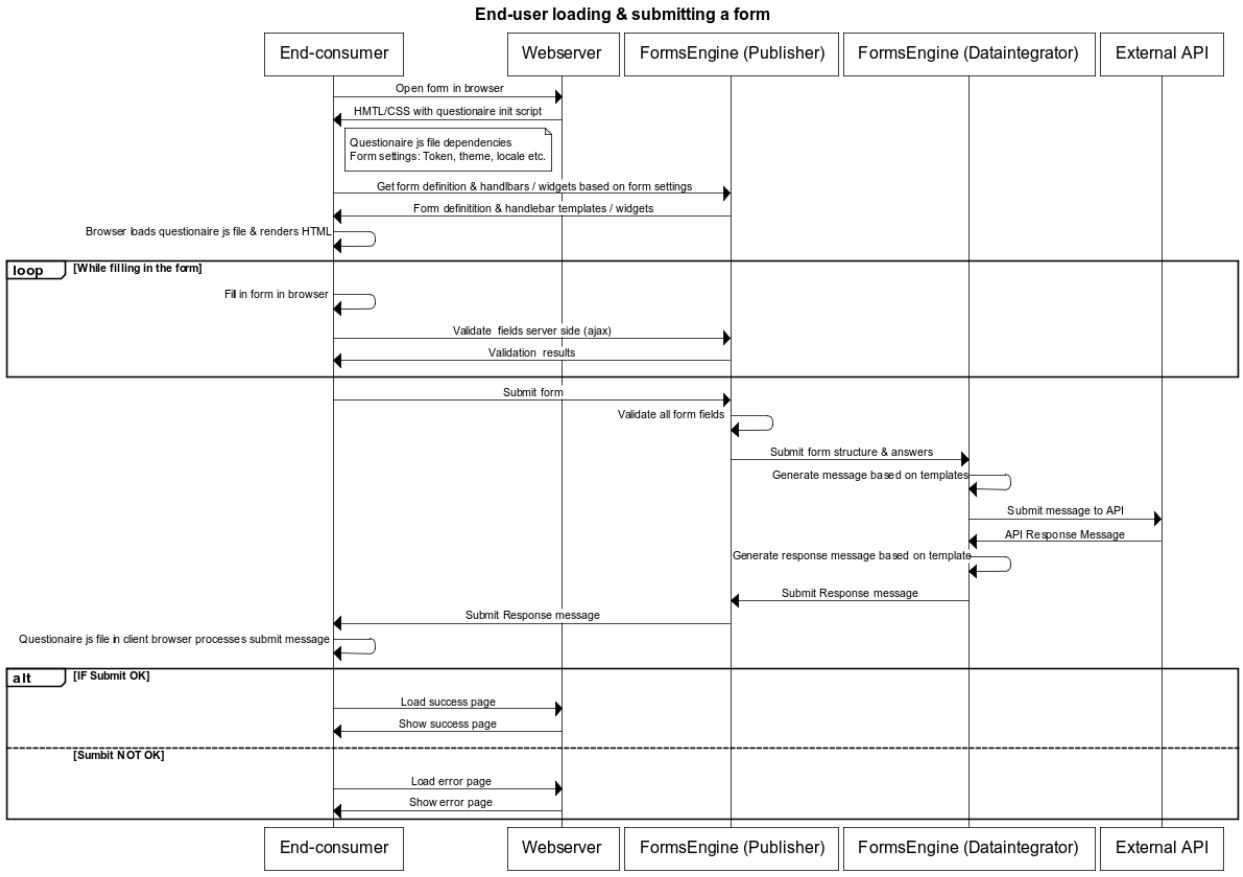
In de FormsEngine CMS worden form definities aangemaakt en beheert. Deze omgeving is volledig ingericht op het samenstellen van formulieren op een functioneel niveau. Dat wil zeggen dat een formulier wordt gemaakt via een gebruikers interface waarin de vragen als vragen worden ingevoerd, de gebruiker hoeft geen kennis van de opbouw van webpagina's te hebben. Op het moment dat (een versie van) een formulier af is kan de gebruiker deze publiceren op een geselecteerde Publisher.

Bij het publiceren wordt de definitie van het formulier op de Publisher geplaatst. Deze maakt het formulier als html beschikbaar. Deze kan dan door een externe site worden opgenomen om aan eindgebruikers te tonen. De opbouw van de html kan daarbij in de Publisher per geval bepaald worden, zodat deze aansluit bij de rest van een website.

Dit proces is in grote lijnen aangegeven in onderstaand diagram



De invoer op het formulier wordt vanuit de pagina weer teruggestuurd naar de Publisher. Deze stuurt dit door naar DataIntegrator. In DataIntegrator worden de gegevens omgezet naar een webservice call naar keuze, bepaald door instellingen en Freemarker templates. Deze call wordt uitgevoerd en de response wordt ook weer via Freemarker templates terugvertaald naar het formaat voor de Publisher. Dit wordt weer naar de Publisher teruggestuurd en in het formulier dat de eindgebruiker ziet verwerkt. Onderstaand diagram laat schematisch zien hoe dit in zijn werk gaat.



De Freemarker templates in DataIntegrator worden door een ontwikkelaar ingevuld en geplaatst.

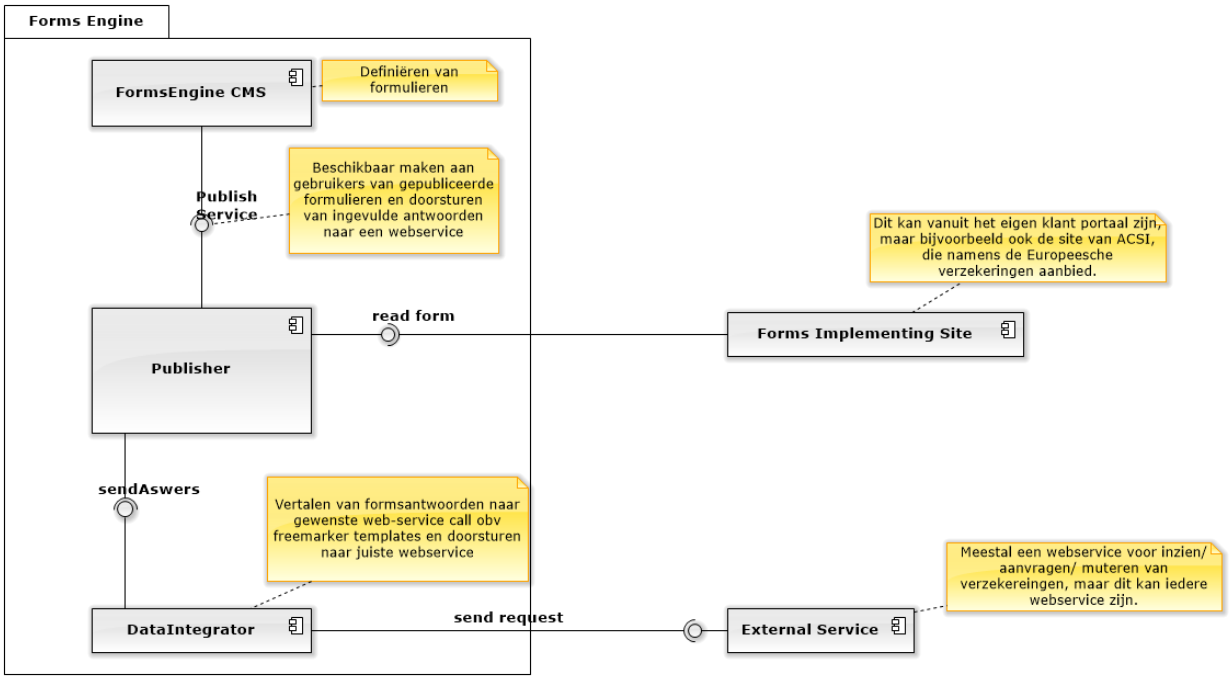
13.5.2 Huidige architectuur

In onderstaande Afbeelding is de architectuur weergegeven als component diagram. Daarbij zijn twee componenten opgenomen die geen deel uitmaken van de Forms engine, maar wel belangrijk zijn om de interactie te begrijpen. Dat zijn de “Forms implementing site” en de “External service”. De werking van de omgeving wordt verder uitgelegd in “Werking FormsEngine”

De Forms engine zelf bestaat uit drie componenten die middels services op elkaar aansluiten:

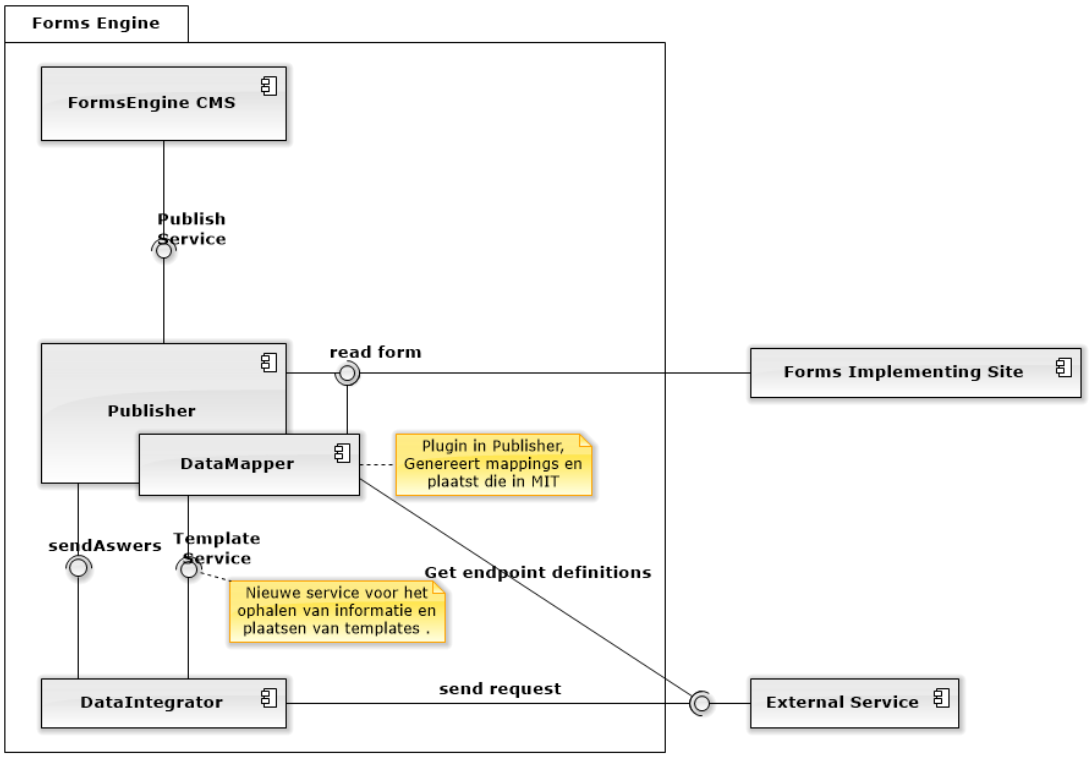
- FormsEngine CMS: Voor het samenstellen van formulieren.
- Publisher: Ontsluit de vanuit de FormsEngine CMS gepubliceerde formulieren via een webservice en stelt javascript beschikbaar om deze formulieren te integreren.
- DataIntegrator: Deze vertaalt het formulier naar een call naar externe services.

In de volgende afbeelding zijn deze componenten met hun interfaces weergegeven.



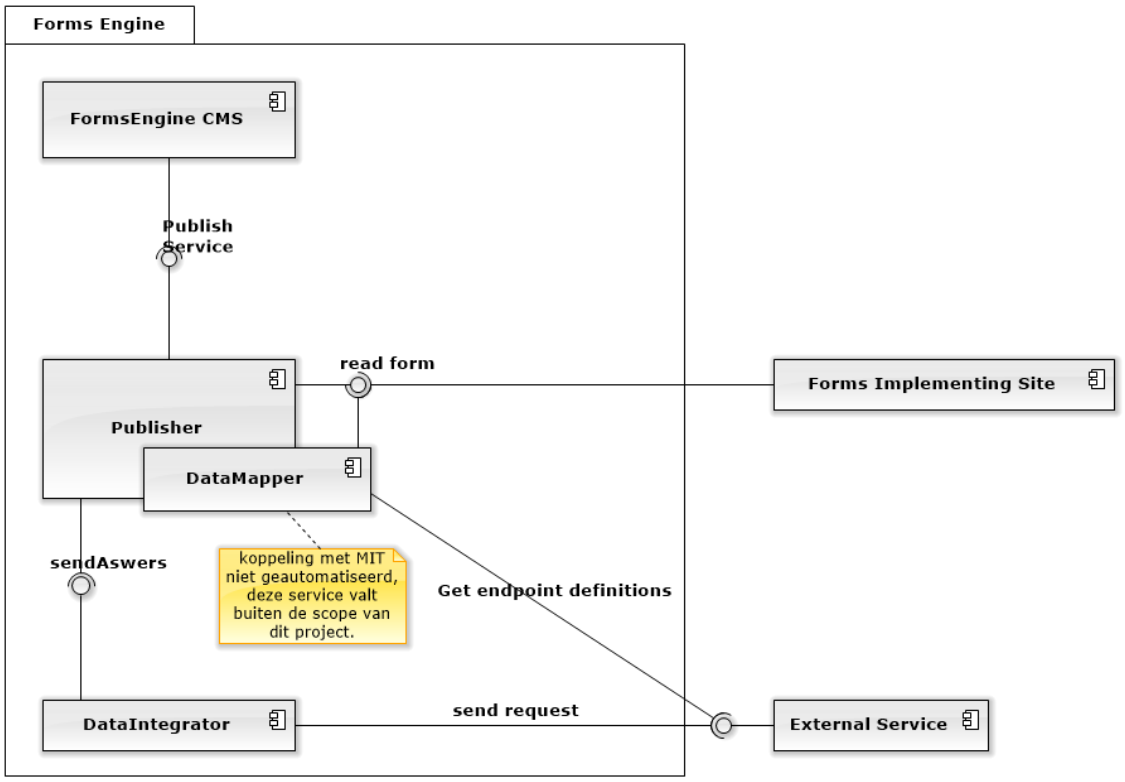
13.5.3 Uiteindelijke architectuur

In de volgende afbeelding zijn de toevoegingen aan de architectuur opgenomen.



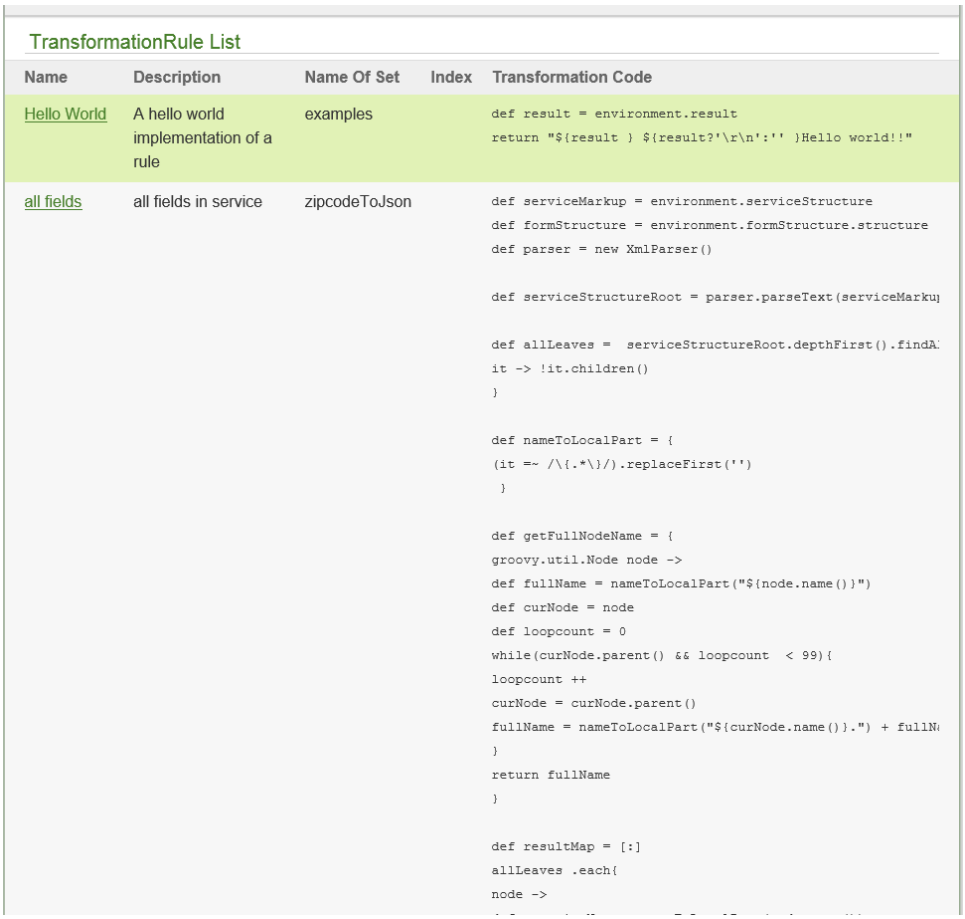
13.5.4 Architectuur binnen scope van afstudeer project

In onderstaande afbeelding is de architectuur weergegeven zoals die voor dit project beoogd is.



13.6 Screenshots gerealiseerde applicatie

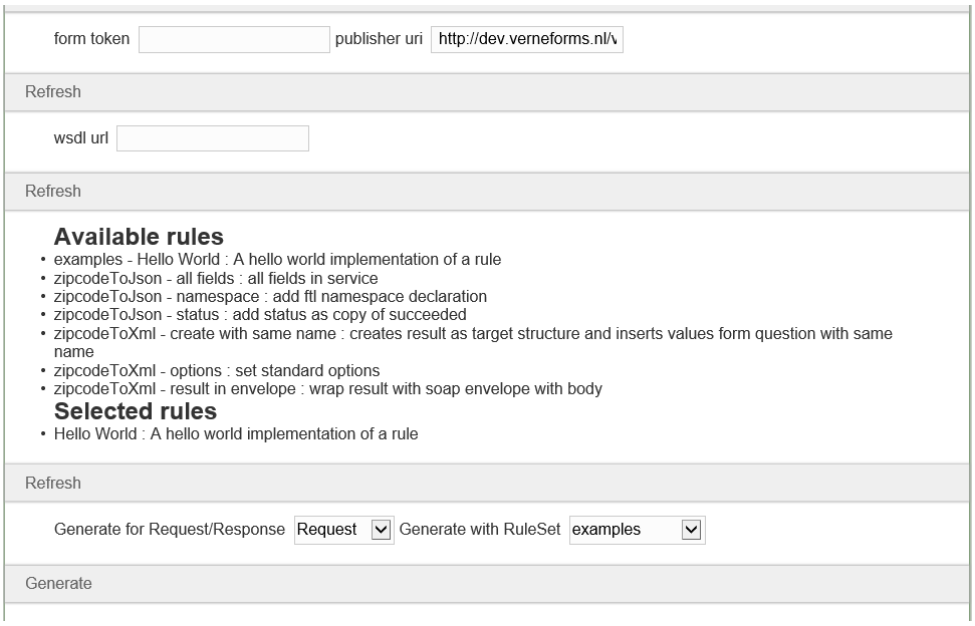
13.6.1 Prototype 0.1



TransformationRule List

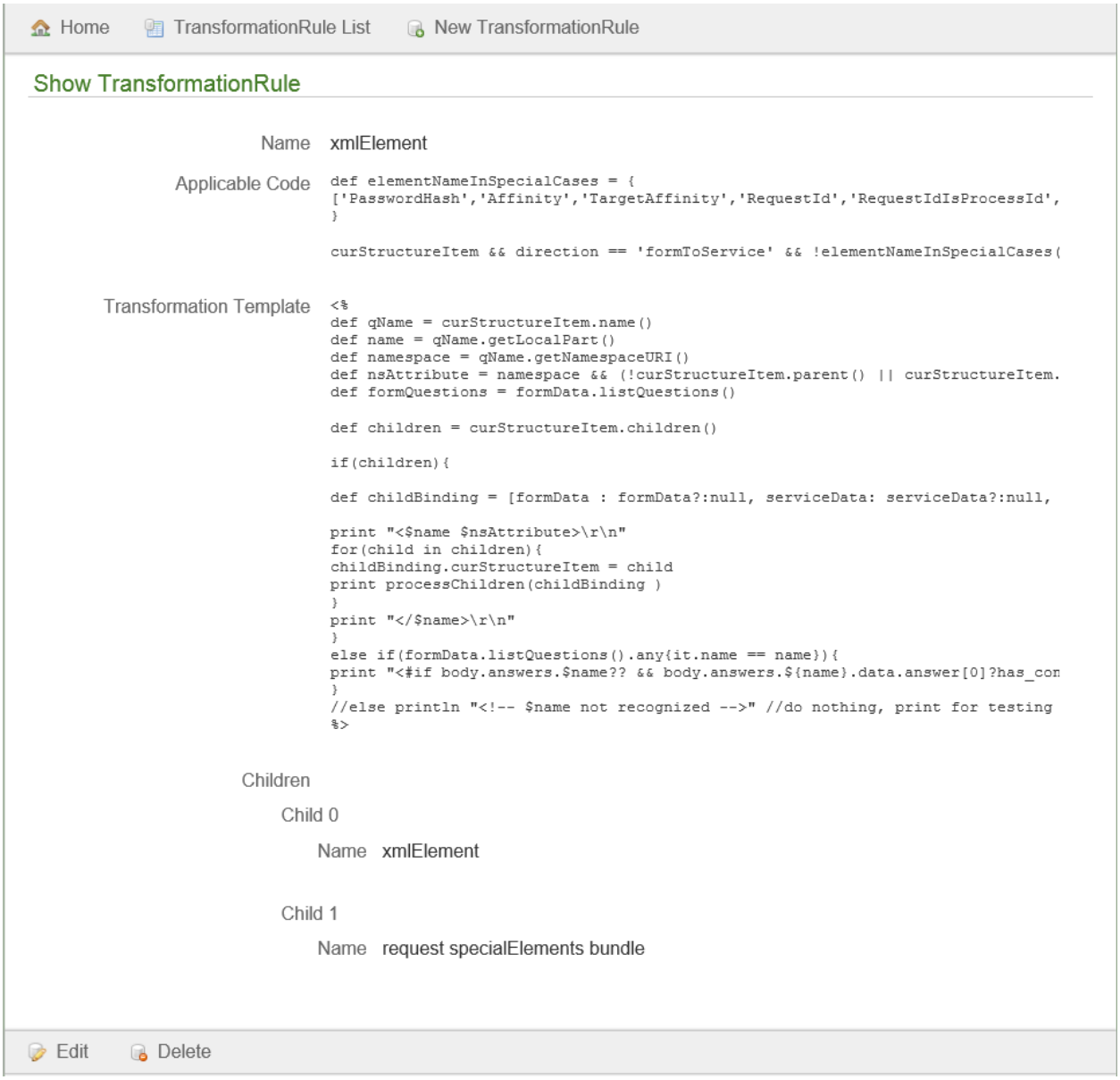
Name	Description	Name Of Set	Index	Transformation Code
Hello World	A hello world implementation of a rule	examples		def result = environment.result return "${result } ${result?'\r\n':'' }Hello world!!"

all fields	all fields in service	zipcodeToJson		def serviceMarkup = environment.serviceStructure
				def formStructure = environment.formStructure.structure
				def parser = new XmlParser()
				
				def serviceStructureRoot = parser.parseText(serviceMarku
				
				def allLeaves = serviceStructureRoot.depthFirst().findA
				it -> !it.children()
				}
				
				def nameToLocalPart = {
				(it =~ /\{.*\}/).replaceFirst('')
				}
				
				def getFullNodeName = {
				groovy.util.Node node ->
				def fullName = nameToLocalPart("${node.name()}")
				def curNode = node
				def loopcount = 0
				while(curNode.parent() && loopcount < 99){
				loopcount ++
				curNode = curNode.parent()
				fullName = nameToLocalPart("${curNode.name()}.") + fullN
				}
				return fullName
				}
				
				def resultMap = [:]
				allLeaves .each{
				node ->


form token
publisher uri
http://dev.verneforms.nl/v
Refresh
wsdl url
Refresh
Available rules
examples - Hello World : A hello world implementation of a rule
zipcodeToJson - all fields : all fields in service
zipcodeToJson - namespace : add ftl namespace declaration
zipcodeToJson - status : add status as copy of succeeded
zipcodeToXml - create with same name : creates result as target structure and inserts values form question with same name
zipcodeToXml - options : set standard options
zipcodeToXml - result in envelope : wrap result with soap envelope with body
Selected rules
Hello World : A hello world implementation of a rule
Refresh
Generate for Request/Response
Request
Generate with RuleSet
examples
Generate


Show Generator
generated
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" >
<soapenv:Body>
<FetchAddressInformation xmlns='https://verzekeringen.opverne.nl'>
<searchZipCodeServiceInfo>
<Address>
<#if (body.answers.Street.data.answer[0])?has_content ><Street>${body.ans
<#if (body.answers.HouseNumber.data.answer[0])?has_content ><HouseNumber>
<#if (body.answers.HouseNumberAddition.data.answer[0])?has_content ><Hous
<#if (body.answers.City.data.answer[0])?has_content ><City>${body.answers
<#if (body.answers.State.data.answer[0])?has_content ><State>${body.answe
<#if (body.answers.ZipCode.data.answer[0])?has_content ><ZipCode>${body.a
<#if (body.answers.CountryCode.data.answer[0])?has_content ><CountryCode>
<#if (body.answers.CountryName.data.answer[0])?has_content ><CountryName>
<#if (body.answers.IsPostbus.data.answer[0])?has_content ><IsPostbus>${bc
<#if (body.answers.ShouldValidate_Translated.data.answer[0])?has_content
<#if (body.answers.IsPostbus_Translated.data.answer[0])?has_content ><IsP
</Address>
<#if (body.answers.Organization.data.answer[0])?has_content ><Organization>
<#if (body.answers.Building.data.answer[0])?has_content ><Building>${body.a
<#if (body.answers.POBox.data.answer[0])?has_content ><POBox>${body.answers
<#if (body.answers.Language.data.answer[0])?has_content ><Language>${body.a
<#if (body.answers.Country_Format.data.answer[0])?has_content ><Country_For
<#if (body.answers.ExternalSearch.data.answer[0])?has_content ><ExternalSea
</searchZipCodeServiceInfo>
<options>
<#if body.answers.processId?? && body.answers.processId.data.answer
<RequestId>${body.answers.processId.data.answer[0]}</RequestId>
<RequestIdIsProcessId>true</RequestIdIsProcessId>
<UseCredentialsFromOriginalWlsProcess>true</UseCredentialsFromO
</#if>
<Credentials>
<UserId>${headers.connectorProperties.Credentials_UserId!''}</Us
<PasswordHash>${headers.connectorProperties.Credentials_Passwor
<Affinity>${headers.connectorProperties.Credentials_Affinity!''}
<#if body.answers.TargetAffinity?? && body.answers.TargetAffinit
<TargetAffinity>${body.answers.TargetAffinity.data.answer[0]}</T
<#else>
<TargetAffinity/>
</#if>
</Credentials>
<#if (body.answers.DisplayLanguage.data.answer[0])?has_content ><DisplayLan
<TestConfiguration>
<#if (body.answers.TestMethod.data.answer[0])?has_content ><TestMethod>${
<#if (body.answers.TestModeRecipientEmailAddress.data.answer[0])?has_cont
</TestConfiguration>
<#if body.answers.IsProcessStartedByBackOffice?? && boo
<IsProcessStartedByBackOffice>${body.answers.IsProcessS
<#else>
<IsProcessStartedByBackOffice>false</IsProcessStartedBy
</#if>
</options>
</FetchAddressInformation>
</soapenv:Body>
</soapenv:Envelope>

13.6.2 Prototype 0.2



13.7 Hypothetische rules bij design

13.7.1 Rules met toepasbaarheid en composite pattern voor complexe rules

Bij deze variant bevatten de classes voor de rules veel generieke intelligentie.

```
IsApplicable RuleRoot
        return !context.structure.hasParent()
Begin RuleRoot
def curNode = context.structure
        result.write("<soapenv:evelope xmlns\"hable\"><soapenv:body><${curNode.name}
xmlns:${curNode.namespace} >")
        result.write(RunApplicableRules(curNode.children()))
        result.write("</${curNode.name}></soapenv:body></soapenv:envelope>")
        return result
End RuleRoot

IsApplicable RuleAnyNode
        return true
Begin RuleAnyNode
        def curNode = context.structure
        result.write("<${curNode.name}>")
        result.write(RunApplicableRules(curNode.children()))
        result.write("</${curNode.name}>")
        return result
End RuleAnyNode

IsApplicable RuleListNode
        return true
Begin RuleListNode
        def curNode = context.structure
        result.write("<${curNode.name}>")
        result.write("<#list curNode.name as item>")

        result.write(RunApplicableRules(curNode.children()))

        result.write(</#list>)
        result.write("</${curNode.name}>")
        return result
End RuleListNode
```

13.7.2 structuur bepaald door statische code

Bij deze optie wordt de structuur niet door user/templates bepaald, maar door de classes die de rules bevaten/uitvoeren. Groot nadeel hiervan is dat er een andere code nodig is als de structuur die wordt doorlopen in source of target structuur verandert.

```
Begin StartPointCreateSoap
        def result = new StringWriter()
```

```
        result.write('<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"><soapenv:Body>')
        result.write(runSub())
        result.write('</soapenv:Body></soapenv:Envelope>')

        return result
End StartPointCreateSoap

Begin ComplexXmlNode
        def result = new StringWriter()
        def currentNode = context.targetStructure

        result.write("<${currenNode.name} xmlns:${currentNode.namespace}>")

        result.write(runSub())

        result.write("</${currenNode.name}>")

        return result
End ComplexXmlNode

Begin ListXmlNode
        def result = new StringWriter()
        def currentNode = context.targetStructure
        def answersArrayName = 'item'

        result.write("<${currenNode.name} xmlns:${currentNode.namespace}>")

        result,write("<#list (body.answers.${currenNode.name})! as ${answersArrayName} >")

                result.write(runSub())

        result.write("</#list></${currenNode.name}>")

        return result
End ListXmlNode

Begin XmlNodeValue
def result = new StringWriter()
        def currentNode = context.targetStructure

        result.write("<${currenNode.name}
xmlns:${currentNode.namespace}>\${(${context.answersArrayName}.${currenNode.name}.data.answer
[0])!}</${currenNode.name}>")

        return result
End SimpleXmlNode
```

13.7.3 user selected Templates

In dit concept schrijft de gebruiker zelf in zijn rule welke rule wordt uitgevoerd om (een deel van) de inhoud van het resultaat te bepalen.

```
Begin StartPointCreateSoap
	def result = new StringWriter()

	result.write('<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"><soapenv:Body>')
	context.answersArrayName = 'body.answers'
	result.write(runRule(ruleName: 'AnyXmlNode', context: context]))
	result.write('</soapenv:Body></soapenv:Envelope>')

	return result
End StartPointCreateSoap

Begin AnyXmlNode
	def result = new StringWriter()
	def currentNode = context.targetStructure

	if(currentNode.type == 'ComplexType'){
		result.write(runRule(ruleName: 'ComplexXmlNode', context: context]))
	}
	else if(currentNode.type == 'List'){
		result.write(runRule(ruleName: 'ListXmlNode', context: context]))
	}
	else{
		result.write(runRule(ruleName: 'XmlNodeValue', context: context]))
	}

	return result
End AnyXmlNde

Begin ComplexXmlNode
	def result = new StringWriter()
	def currentNode = context.targetStructure

	result.write("<${currenNode.name} xmlns:${currentNode.namespace}>")

	currentNode.children().each{
		node ->
		context.targetStructure = node
		result.write(runRule(ruleName: 'AnyXmlNode', context: context]))
	}

	result.write("</${currenNode.name}>")

	return result
```

```
End ComplexXmlNode

Begin ListXmlNode
	def result = new StringWriter()
	def currentNode = context.targetStructure
	def answersArrayName = 'item'

	result.write("<${currenNode.name} xmlns:${currentNode.namespace}>")

	if(currentNode.name == 'entitiies'){
		result.write(runRule(ruleName: 'EntitiesOwnerWithPartner', context: context))
	}

	result,write("<#list (body.answers.${currenNode.name})! as ${answersArrayName} >")

	context.answersArrayName = answersArrayName
	currentNode.children().each{
		node ->
		context.targetStructure = node
		result.write(runRule(ruleName: 'AnyXmlNode', context: context]))
	}

	result.write("</#list></${currenNode.name}>")

	return result
End ListXmlNode

Begin XmlNodeValue
def result = new StringWriter()
	def currentNode = context.targetStructure

	result.write("<${currenNode.name}
xmlns:${currentNode.namespace}>\${(${context.answersArrayName}.${currenNode.name}.data.answer
[0])!}</${currenNode.name}>")

	return result
End SimpleXmlNode
```

13.8 Gespreksverslagen

13.8.1 Verslag project kickoff automatische koppeling formulieren aan webservices 30-01-2015 bij MyCubes

13.8.1.1 Deelnemers

Hary Selman, John Li, Twan van Buuren

13.8.1.2 Besproken

Allereerst is de doelstelling van het project besproken. Vervolgens is de architectuur van de forms omgeving uiteengezet.

Daarna is bediscussieerd waar en op welke manier een oplossing het beste op de forms omgeving kan worden aangesloten.

De conclusie van het gesprek was dat een oplossing het beste als plugin op de forms-publisher kan worden aangesloten, informatie die vanuit de DataIntegrator module nodig is zal dan via een API moeten worden opgehaald. Dit betekent dat de oplossing binnen het grails framework ontwikkeld zal worden.

Uitdagingen die daarbij duidelijk zijn:

- Een wsdl moet geparsed worden. Een idee voor de oplossingsrichting zou opgedaan kunnen worden door naar SOAPUI te kijken.
- Er moet informatie over het betreffende form worden doorgegeven, de informatie die nu beschikbaar is zal wellicht nog aanvulling behoeven.
- Er moet gekeken worden hoe de informatie het beste kan worden doorgegeven aan de plugin: als object structuur of als xsd of JsonSchema.
- Als voor de laatste optie wordt gekozen: wat zijn goede libraries/ modules om dit te parsen
- Hoe worden de twee informatie bronnen aan elkaar gekoppeld/ hoe wordt de mapping bepaald?
    - Dit blijft dynamisch en aanpasbaar door dit te doen dmv een rule engine, die regels in een programma taal uitvoert, in die regels kan worden aangegeven wat de output moet zijn in nader te bepalen gevallen.
    - Door met regels te werken die aangepast en toegevoegd of verwijderd kunnen worden kan de koppeling ook continu verfijnd en of aangepast worden.

13.8.2 Verslag functionaliteit bespreking automatische koppeling formulieren aan webservices 03-02-2015

13.8.2.1 Deelnemers

Hary Selman, Twan van Buuren, Kevin Brussen

13.8.2.2 Besproken

Er is eerst besproken wat de doelstelling van het project is.

Aan de hand hiervan is bediscussieerd welke functionaliteiten verwacht worden om aan de doelstelling te voldoen.

De conclusie van het gesprek was dat de volgende functionaliteiten van belang zijn voor Verne om de doelstelling te behalen:

- De tool kan zelf omgaan met nieuwe structuren.
- Regels en conventies die zijn vastgelegd, moeten steeds voor een volgende transformatie kunnen worden hergebruikt.
- De regels en conventies/ logica van de tool moet aanpasbaar/ aanvulbaar zijn zonder de software opnieuw te moeten deployen.
- De tool moet zelf de benodigde en beschikbare informatie ophalen.