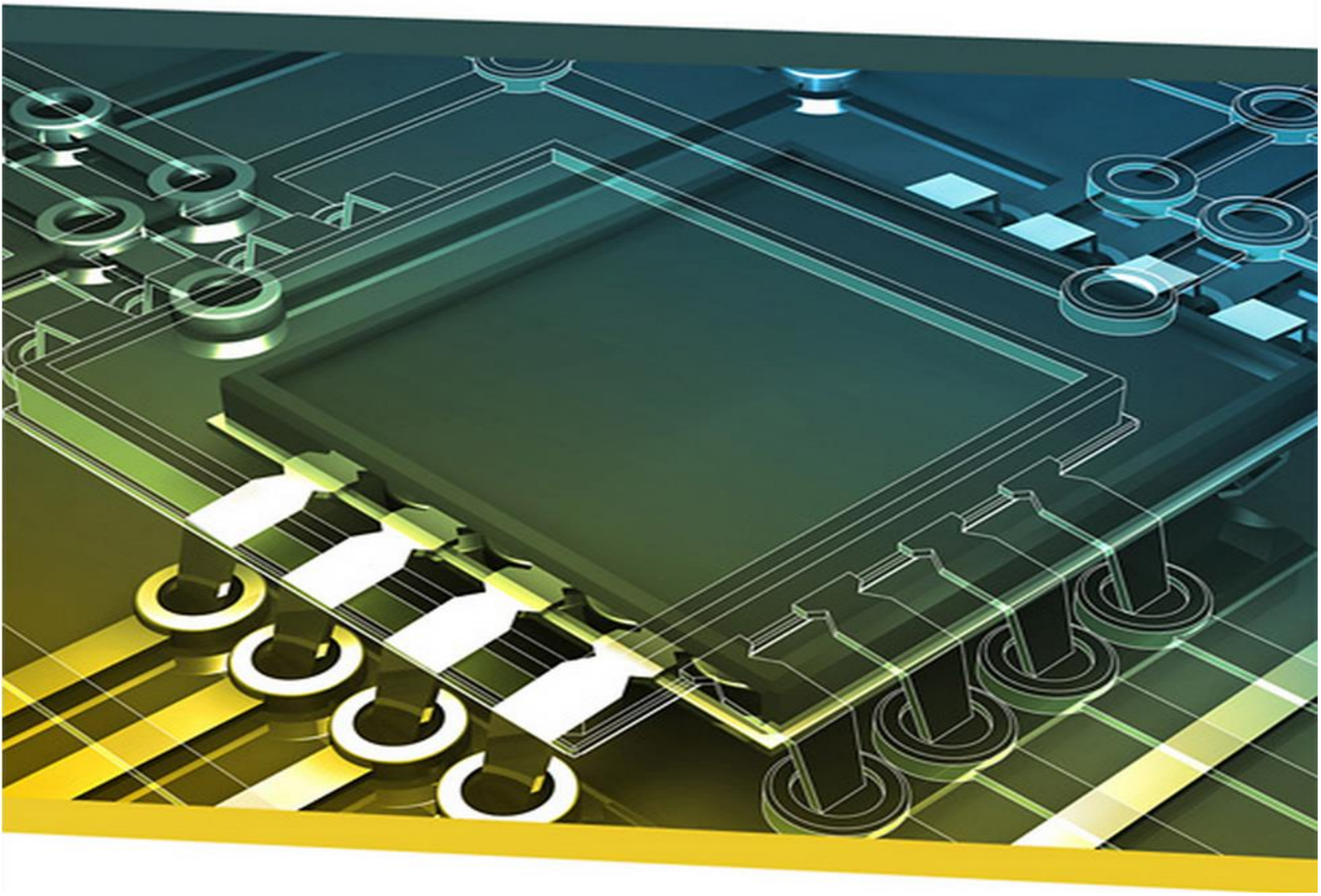


Project import binnen iDRM

Scriptie

Naam:	Floris Velleman
Student nummer:	1602376
1e examinator:	Alex Jongman
Bedrijf:	NP-Komplete Technologies

Project import binnen iDRM - Scriptie

Student

Naam:	Velleman
Voorletters:	F.M.V.
Roepnaam:	Floris
Studentnummer:	1602376
Docentbegeleider:	Alex Jongman

Bedrijf

Naam:	NP-Komplete Technologies
Adres:	De Rozentuin 18
Postcode:	5611 SW
Plaats:	Eindhoven
Inhoudelijk bedrijfsbegeleider:	Maarten Berkens
Technisch bedrijfsbegeleider:	Rob Gelderblom

Versiebeheer

Onderstaande tabel geeft een weergave van de voorgaande versies en gemaakte revisies van dit document.

Versie	Status	Datum	Omschrijving
2.0	Voltooid	27-05-2015	Bronnenlijst
1.8	Lopend	24-05-2015	Verbeteringen naar aanleiding van feedback.
1.7	Lopend	21-05-2015	Verbeteringen naar aanleiding van feedback.
1.6	Lopend	20-05-2015	Voorwoord en management samenvatting.
1.5	Lopend	19-05-2015	Reflectie opgenomen.
1.4	Lopend	18-05-2015	Conclusie opgenomen.
1.3	Lopend	16-05-2015	Testresultaten opgenomen.
1.2	Lopend	09-05-2015	Resultaten realisatiefase opgenomen.
1.1	Lopend	08-05-2015	Conclusies voor onderzoeksfase.
1.0	Lopend	07-05-2015	Technisch ontwerp opgenomen.
0.9	Lopend	06-05-2015	Resultaten technische analyse opgenomen.
0.8	Lopend	03-05-2015	Verbeteringen naar aanleiding van feedback.
0.7	Lopend	28-04-2015	Verbeteringen naar aanleiding van feedback.
0.6	Lopend	21-04-2015	Onderzoekresultaten analyse opgenomen.
0.5	Lopend	14-04-2015	Verdere verbeteringen aan onderzoeksfase.
0.4	Lopend	12-04-2015	Verbeteringen naar aanleiding van feedback.
0.3	Lopend	07-04-2015	Verbeteringen naar aanleiding van feedback.
0.2	Lopend	25-03-2015	Verbeteringen aan de structuur.
0.1	Lopend	23-03-2015	Opzet document.

Voorwoord

Een uitdagende opdracht als hoogtepunt van de opleiding Software Engineering aan de Hogeschool Utrecht. Tijdens deze opdracht heb ik een hoop geleerd en is een geweldig resultaat neergezet.

Het schrijven van een scriptie was uitdagend maar met de hulp van de begeleiders goed te doen. Ik heb het gevoel dat dit project mij ook verdere verdieping heeft gegeven in het doen van onderzoek. Dit is een vaardigheid die erg van pas zal komen in de toekomst.

Ik wil graag Alex Jongman bedanken voor alle tijd en moeite die hij heeft genomen voor het begeleiden van deze scriptie. Zijn concrete en verhelderende uitleg heeft ervoor gezorgd dat het document van hogere kwaliteit is geworden.

Op technisch gebied wil ik graag Rob Gelderblom bedanken. Hij was altijd een aanwezig aanspreekpunt voor lastige technische punten. Het controleren van de kwaliteit en het snelle vermelden van aandachtspunten heeft bijgedragen aan de kwaliteit van het uiteindelijke product.

Simon Klaver heeft vanuit zichzelf en de klant een hoop feedback weten te geven. Ik wil Simon bedanken voor alle tijd die hij beschikbaar heeft gemaakt en de moeite die hij heeft genomen om zo uitvoerig te kijken naar het product.

Eindhoven
21-05-2015
Floris Velleman

Management samenvatting

Voor NP-Komplete Technologies is een onderzoek uitgevoerd naar hoe een import widget kan worden gemaakt die alle relevante projectelementen in iDRM kan importeren en hierbij conflicten kan verminderen/oplossen.

Om dit te doen is er in eerste instantie een onderzoek gedaan naar hoe de huidige situatie verloopt. Hieruit is gebleken dat er verbeteringen mogelijk zijn op het gebied van import methodes. Dit is vervolgens uitgewerkt naar een daadwerkelijke interface en een architectuur in de vorm van een UML Klassendiagram.

Na het onderzoek verricht te hebben is er een prototype ontwikkeld op basis van de resultaten uit de onderzoeksfase. Dit prototype is getest en heeft volgens de testen een functionele werking die voldoet aan de wensen van de gebruikers die de import widget wilden.

De testen die zijn uitgevoerd voor de gebruikersvriendelijkheid van de import widget tonen aan dat er nog mogelijkheden zijn voor verbeteringen maar dat er voortgang is geboekt ten opzichte van de voorgaande versies.

Op basis van deze bevindingen is geconcludeerd dat de gemaakte import widget kan worden gebruikt voor het importeren van projectelementen in iDRM en dat de conflicten die hierbij ontstaan kunnen worden opgelost door de manier waarop deze worden afgehandeld in de import widget.

Verbetering is mogelijk door nog een iteratie uit te voeren. Deze verbeteringen bevatten minder belangrijke onderdelen die de import widget completer maken.

Op het gebied van gebruikersvriendelijkheid kan iteratief gewerkt worden en zou een hogere gebruikersvriendelijkheid mogelijk zijn door het gebruikte proces te herhalen.

Inhoudsopgave

2. Inleiding 8
3. NP-Komplete Technologies 9
4. Opdracht 10
4.1. Probleemstelling 10
4.2. Doelstelling 11
4.3. Hoofd- en deelvragen 12
4.4. Aanpak 12
5. Onderzoeksfase 13
5.1. Analyse huidige situatie 13
5.1.1. Bevindingen 13
5.1.2. Conclusie 15
5.2. Analyse huidige project structuur 16
5.2.1. Analyse bestanden 16
5.2.2. Samenhang 17
5.2.3. Conclusie 19
5.3. Requirements 19
5.3.1. Requirements en eisen 20
5.3.2. Actors 21
5.3.3. Importeren van een project 23
5.3.4. Conclusie 23
5.4. Analyse van de huidige project import 23
5.4.1. Bevindingen 24
5.4.2. Conclusie 24
5.5. Analyse van de mogelijke conflicten 25
5.5.1. Bevindingen 25
5.5.2. Conclusie 26
5.6. Verbeterde architectuur 26
5.6.1. Bevindingen klassendiagram 27
5.6.2. Bevindingen design 28
5.6.3. Conclusie 29
6. Realisatiefase 30
6.1. Import widget 30
6.1.1. Iteratie 1 30

6.1.2. Iteratie 2 32
6.1.3. Conclusie 32
7. Validatiefase 34
7.1. Validatie architectuur 34
7.1.1. Iteratie 1 34
7.1.2. Iteratie 2 35
7.1.3. Conclusie 35
7.2. Userinterface testen 35
7.2.1. Iteratie 1 36
7.2.2. Iteratie 2 36
7.2.3. Conclusie 37
8. Conclusie 39
9. Aanbevelingen 40
10. Reflectie 41
10.1. Onderzoek 41
10.2. Realisatie 41
10.3. Validatie 42
10.4. Leerpunten 43
11. Bronnen 44
12. Bijlage 45
12.1. Bijlage 1 – Plan van aanpak 45
12.2. Bijlage 2 – Functioneel ontwerp 68
12.3. Bijlage 3 – Technisch ontwerp 90
12.4. Bijlage 4 - Testplan 113
12.5. Bijlage 5 – Testrapport 128
12.6. Bijlage 6 – Test scenario en bevindingen 140
12.7. Bijlage 7 – Testscripts 143

2. Inleiding

Als afsluitend project is er in de periode van 2 februari 2015 tot 2 juni 2015 een afstudeeropdracht uitgevoerd in opdracht van de Hogeschool Utrecht. Dit is een onderdeel van de bachelor Software Engineering. Deze afstudeeropdracht is gedaan bij NP-Komplete Technologies in Eindhoven.

De klanten van NP-Komplete Technologies hebben een probleem aangegeven. Dit probleem komt vanuit het ontbreken van functionaliteit in het import/export proces. In het hoofdstuk Opdracht wordt gekeken naar hoe dit tot een daadwerkelijke doelstelling heeft geleid.

Deze opdracht zorgt uiteindelijk voor hoofd- en deelvragen die gedurende de afstudeerperiode zijn onderzocht. De bevindingen en conclusies die ontstaan zijn door het onderzoek naar een antwoord op de vragen worden als eerste behandeld. Dit is terug te vinden in het hoofdstuk Onderzoek en biedt verdieping in zowel de resultaten als het proces. Hierbij komen verschillende methodieken naar voren en afwegingen die gemaakt zijn waardoor een resultaat mogelijk is ontstaan.

Op basis van de documentatie en kennis die is opgedaan tijdens de onderzoeksfase is vervolgens het volgende hoofdstuk opgesteld. Het hoofdstuk Realisatie betreft een vergelijkbare structuur als die van het hoofdstuk Onderzoek. Hierbij worden bevindingen aangehaald en wordt uiteindelijk bepaald wat gelukt is in de geplande tijd.

Gezien deze resultaten moeten worden gevalideerd is het hoofdstuk Validatie opgenomen. In dit hoofdstuk worden de resultaten beschreven van de verschillende testen die zijn uitgevoerd op het product dat is opgeleverd door de realisatiefase. Dit hoofdstuk geeft uiteindelijk een weergave van de kwaliteit van de uitvoering van de opdracht door middel van test bevindingen.

Als laatste onderdeel wordt met behulp van de voorgaande hoofdstukken een conclusie geschreven. Deze conclusie probeert een antwoord te geven op hoofdvraag. Dit gebeurt aan de hand van de conclusies die eerder zijn gemaakt op basis van de bevindingen. De aanbevelingen en suggesties voor vervolgonderzoek worden in een los hoofdstuk beschreven.

Tot slot volgt een reflectie waarin wordt gekeken naar wat goed en minder goed ging gedurende de afstudeerstage.

3. NP-Komplete Technologies

NP-Komplete Technologies is een software bedrijf binnen de branche EDA (Electronic Design Automation voor chip design) en bevindt zich in Eindhoven (NP-Komplete Technologies, 2015). Het bedrijf is opgericht in 2009 (Bedrijfsinformatie, 2009), maar de meeste personeelsleden werkten al samen in voorloper Takumi Technology (Takumi, 2004).

De naam NP-Komplete Technologies komt van nondeterministic polynomial time (Wikipedia, 2015). Maar gebruikt hiervoor Komplete in plaats van Complete omdat dit bedrijf al bestond.

Het is een bedrijf met ongeveer tien tot vijftien werknemers. De opdeling van het bedrijf vindt zich terug in de beroepen die mensen hebben. Er zijn geen afdelingen binnen het bedrijf. Er zijn binnen het bedrijf programmeurs maar ook AE's (application engineers). Deze application engineers werken nauw samen met klanten en helpen bij installatie. Soms werken ze zelfs bij klanten met de programma's van NP-Komplete Technologies.

De huidige software die NP-Komplete Technologies maakt staat bekend als iDRM wat staat voor integrated design rule management (Sage, 2015). Dit programma wordt gemarket door het bedrijf Sage Design Automation (hoofdkantoor in Santa Clara, CA). Deze software wordt gebruikt om chip design te testen. Ook ontwikkelt NP-Komplete Technologies nog een aantal ondersteunende tools die bepaalde opdrachten in de branche makkelijker maken. De klanten mogen niet worden vermeld vanwege een non-disclosure agreement.

De student neemt de rol aan van software engineer binnen de organisatie. Het afstudeerproject zelf staat los van andere projecten binnen de organisatie maar is wel onderdeel van het uiteindelijke programma.

4. Opdracht

NP-Komplete Technologies bouwt iDRM, een physical design rule check tool. Het programma wordt uiteindelijk geleverd aan chip fabrikanten.

iDRM is een programma waarmee bepaalde design rules van chips kunnen worden gecontroleerd op basis van een regel. Deze regel is gebaseerd op een patroon (pattern) die toepassing heeft op een bepaalde laag uit de bestaande verzameling van lagen van de chip (layermap). De lagen van deze chip geven een basis om een specifiek figuur te vinden (het pattern). Een regel kan op verschillende lagen zoeken. Het programma helpt chip fabrieken met het controleren van deze regels in chip designs zodat een goede chip kan worden gemaakt.

De klanten en application engineers die iDRM gebruiken hebben een probleem aangegeven.

Werk van klanten in iDRM is georganiseerd in projecten. Onderdelen van het ene project zijn vaak bruikbaar in een ander project. Bij het importeren van een project in iDRM kunnen patterns niet zonder layermap worden geïmporteerd. Dit zorgt ervoor dat de layermap wordt meegenomen. Dit levert vaak een resultaat wat niet gewenst is. Ook wordt niet aangegeven wat er precies is veranderd.

Zo is het mogelijk dat een layer op naam wordt overschreven waardoor de bestaande mapping van patterns een hele andere betekenis krijgt. Ook kan het dat er meerdere layermaps zijn die eigenlijk hetzelfde doen.

De klanten willen dat er kan worden gekozen voor een import met of zonder layermap. Ook zouden ze graag extra functionaliteit willen in deze import. Zo willen ze bijvoorbeeld kunnen kiezen welke projectelementen er wel en niet wordt geïmporteerd. De klanten willen de mogelijkheid om conflicten te kunnen verminderen en ze makkelijker te kunnen oplossen.

Vanuit het bedrijf komt naar voren dat een mogelijke interface gebruikersvriendelijk moet zijn omdat klanten duidelijk hebben gemaakt dat het bestaande programma op sommige punten niet erg vriendelijk is voor gebruikers.

Dit houdt voornamelijk in dat een nieuw product dit standpunt zou moeten meenemen in het design van de interface.

4.1. Probleemstelling

Er kan al vrij snel gesteld worden dat het hier gaat om het uitbreiden en verbeteren van een bestaand proces. Er is een tekort aan functionaliteit in de huidige import.

Dit komt naar voren doordat er een beperkt aantal opties is bij het importeren. Aangezien export hier indirect mee te maken heeft is het mogelijk het op beide punten te verbeteren. Het daadwerkelijke probleem zit dan ook bij het import/export proces.

Er is te weinig functionaliteit in het huidige import/export proces om te doen wat de application engineers en klanten willen doen. Het niet aangeven van conflicten tijdens het importeren is een gebrekkige functionaliteit in het huidige proces. Dit kan worden opgemaakt uit het feit dat er wel stappen worden doorlopen maar dat er enkel een aantal hoofdlijnen worden gerapporteerd.

Ook wordt vermeld dat er een probleem is met het mergen omdat hier geen keuze kan worden gemaakt per element. Dit impliceert vrijwel direct dat er ook vraag is naar de mogelijkheid om te kiezen voor andere methode van importeren/exporteren.

Zo zou het wellicht mogelijk moeten kunnen zijn een object te kopiëren of om een object te overschrijven.

De manier waarop en hoe een mogelijk nieuw product wordt gebruikt vereist ook aandacht. Zo wordt vermeld dat minstens een deel van de gemaakte functionaliteit niet gemakkelijk te gebruiken is door klanten. Dit is een probleem waarmee rekening moet worden gehouden bij het ontwikkelen van een mogelijk nieuw product.

Er zijn verbeteringen nodig die minimaal de frustraties van de twee groepen wegnemen.

4.2. Doelstelling

Dit probleem leidt tot het bepalen van doelstellingen die het probleem verhelpen. Om dit probleem te verhelpen zou het importeren/exporteren van projectelementen moeten worden verbeterd. Om dit te doen is het nodig verschillende import/export opties te hebben.

Echter zal hier eerst een goede structuur voor moeten worden opgesteld. Dit kan gebeuren door een analyse te maken van de huidige projectstructuur en de huidige code base die wordt gebruikt voor import en export. Deze elementen kunnen worden omschreven als relevante projectelementen aangezien deze gerelateerd zijn aan het import/export proces.

Ook moet het mogelijk zijn om te kiezen tussen welke onderdelen er wel en niet worden geïmporteerd/geëxporteerd. De wijzigingen die ontstaan zouden expliciet vermeld moeten worden.

Er zal een dialog kunnen worden gemaakt waarmee deze opties kunnen worden gekozen. Ook zou deze dialog een eenvoudig bruikbare interface krijgen waarmee zowel de klanten als application engineers kunnen omgaan. Dit is dan ook direct de doelgroep voor de applicatie.

Deze dialog zou tegelijkertijd werk uit handen van de gebruiker moeten halen. Een deel van het doel is dan ook om te bepalen welke afhankelijkheden er zijn tussen de projectelementen en deze op te nemen in de dialog.

Na dit project moet er een eindproduct zijn waardoor projectelementen kunnen worden geïmporteerd/geëxporteerd op de door de klanten gespecificeerde manier. Dit eindproduct moet gebruikersvriendelijk zijn en met behulp van de gebruiker in staat zijn conflicten op een flexibele wijze op te lossen.

4.3. Hoofd- en deelvragen

In dit hoofdstuk wordt de hoofdvraag gedefinieerd. Op basis van deze hoofdvraag wordt vervolgens bepaald welke deelvragen er nodig zijn om dit te beantwoorden. Omdat de hoofdvraag een proof of concept oplevert is er sprake van een productopdracht. De hoofdvraag is als volgt geformuleerd:

"Met behulp van welke architectuur kan een import/export wizard die relevante projectelementen bevat worden gemaakt in iDRM zodat conflicten kunnen worden opgelost of verminderd?"

Deze hoofdvraag zal worden beantwoord op basis van de deelvragen. Deze zijn als volgt geformuleerd:

1. Welke architectuur kan worden gebruikt voor het maken van een import/export wizard binnen iDRM?
    a. Hoe zit de huidige import/export architectuur in elkaar binnen iDRM?
2. Hoe kan een import/export wizard op een voor de doelgroep gebruikersvriendelijke manier worden getoond binnen iDRM?
3. Welk relevante projectelementen worden binnen de IDRM import/export bewaard?
4. Welke conflicten kunnen worden onderscheiden bij het importeren van een project binnen iDRM?

4.4. Aanpak

De ontwikkelstraat bij NP-Komplete Technologies is ad hoc. Dit houdt in dat er vrijwel geen documentatie wordt gemaakt die de software beschrijft. Er is wel een user guide en een zeer beknopte beschrijving maar deze geven maar beperkt aan wat het gedrag van de software is. De aanpak binnen het bedrijf valt niet makkelijk onder te brengen bij een bepaalde methodiek.

In tegenstelling tot wat er standaard bij het bedrijf gebruikt wordt zal er zoals vermeld in het plan van aanpak gedurende het afstudeerproject gebruik worden gemaakt van agile.

Er is gekozen voor deze aanpak om ervoor te zorgen dat het voor alle betrokkenen duidelijk is in welke fase momenteel gewerkt wordt en welke resultaten verwacht kunnen worden. Ook zorgt deze aanpak ervoor dat de kwaliteitscriteria in de eerste fase kunnen worden vastgesteld en worden bevestigd in de laatste fase.

Het afstudeerproject zal worden opgedeeld in een aantal fasen:

- Onderzoek
- Realisatie
- Validatie

Hierbij wordt iteratief gewerkt gedurende de realisatie en validatie. De requirements die voortkomen vanuit de onderzoeksfase worden gebruikt om de prioriteiten aan te geven voor de opvolgende fases. De onderzoeksfase zal eerst moeten worden doorlopen voordat er in andere fases gewerkt kan worden.

5. Onderzoeksfase

Zoals beschreven in het plan van aanpak zal in eerste instantie onderzoek worden gedaan. Ook vanuit de opdracht komt dit naar voren.

Dit zal beginnen door analyses te maken van de huidige situaties. Er kan op dit punt in kaart worden gebracht welke functionaliteit er gewenst is en op welke manier deze uitgewerkt moet worden. Dit biedt een goede houvast voor het ontwikkelen van een prototype en zal uiteindelijk mogelijk een antwoord kunnen geven op de deelvragen.

Er zal gedurende deze fase eerst een functioneel ontwerp worden gemaakt. Zodra dit functioneel ontwerp van voldoende kwaliteit is kan er onderzoek worden gedaan naar de huidige technische opzet van het import/export proces. Er kan vervolgens met behulp van de resultaten van de technische analyses een technisch ontwerp worden gemaakt.

Door te kiezen voor deze aanpak is er een basis waarmee kan worden gewerkt in de realisatiefase en validatiefase.

5.1. Analyse huidige situatie

Om te kunnen bepalen wat gemaakt moet worden is het belangrijk om te weten hoe de huidige import/export situatie werkt. Om dit te doen zal als eerst een analyse moeten worden gemaakt van de huidige situatie. Door deze analyse kan duidelijk worden waar verbeteringen plaats moeten vinden in het proces.

Hoewel deze analyse niet direct een deelvraag beantwoordt is het een benodigde context voor het bepalen van een correcte architectuur. Dit geeft namelijk een beter beeld van de werking van het proces aan de betrokkenen wat zal kunnen bijdragen aan de manier waarop problemen worden benaderd vanuit de gebruikers/application engineers.

De analyse zelf kan worden uitgevoerd door te achterhalen op welke manier de klanten het programma momenteel gebruiken om import/export taken uit te voeren. Dit kan gebeuren door in gesprek te gaan met de gebruikers en te bepalen wat ze doen tijdens het proces.

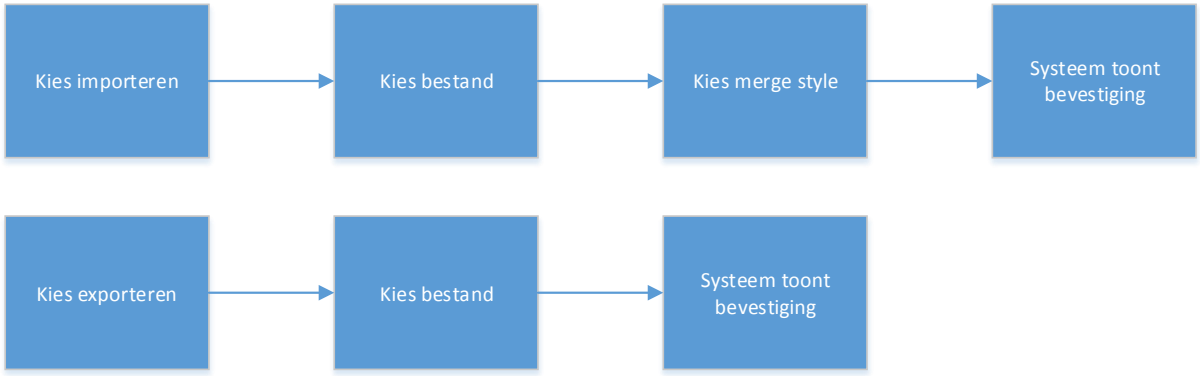
Ook kan door de user guide te lezen wellicht worden opgemaakt hoe dit proces verloopt. Het bedrijf zelf zal ook informatie kunnen hebben over wat de normale flow is van dit proces. Gezien er weinig bestaande documentatie is over dit proces is er gekozen voor deze benadering.

Door het combineren van deze informatie zal een business proces diagram (McKibben, 2015) kunnen worden gemaakt dat op een abstracte wijze weergeeft hoe de processen verlopen.

5.1.1. Bevindingen

Om een duidelijk proces in kaart te brengen is gestart met het vinden van alle import en export opties binnen de interface van iDRM. Vervolgens is er op basis van de werking van deze onderdelen gekeken naar een proces dat door alle import en export mogelijkheden wordt aangehouden. Er is gezocht naar overeenkomsten die tot een minimale abstractie kunnen leiden om zo duidelijk mogelijk te zijn in de beschrijving van het proces.

Door te kijken naar welke mogelijkheden er zijn voor het importeren en exporteren binnen iDRM is een abstract proces diagram opgesteld. Hierin komt terug wat er bij elk van de import/export mogelijkheden overeenkomt.



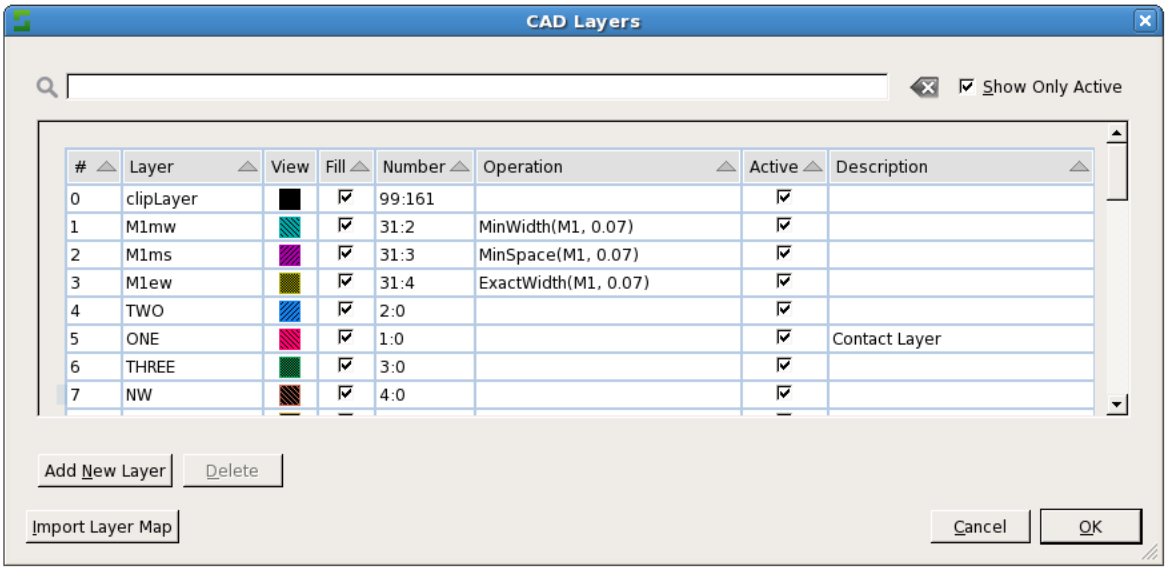
figuur 1 - Procesgang

Tijdens de analyse is duidelijk geworden dat de merge style simpelweg een vaste import optie is. Dit houdt in dat als er een conflict wordt gevonden (een object met dezelfde naam bijvoorbeeld) er in vrijwel alle gevallen wordt gekozen voor het hernoemen van objecten. Het vervolgens veranderen van de relaties die deze elementen met elkaar kunnen hebben wordt in sommige gevallen achterwege gelaten. Dit zorgt er dan ook voor dat er dubbele elementen zijn in het project die soms niet kloppen.

Het importeren van elementen lijkt als prioriteit te hebben dat er geen fouten mogen ontstaan in het project. Als er een lastige situatie wordt gemaakt worden bepaalde elementen gewoon overgeslagen door het huidige proces. De import lijkt binnen het programma altijd goed te gaan maar wat er precies goed is gegaan wordt niet vermeld. Vanuit de gebruikers is duidelijk geworden dat het resultaat meestal niet goed is.

Dit komt voornamelijk doordat er een tekort aan mogelijkheden lijkt te zijn (zo kan er niet worden gekozen voor het importeren zonder bijvoorbeeld layers mee te nemen).

Los hiervan is meerdere elementen importeren enkel een optie binnen het menu (door middel van project import). Zo is er de mogelijkheid tot exporten van vrijwel elk element binnen het programma. Er is echter geen mogelijkheid om op al die plekken ook weer te kunnen importen.



figuur 2 – Cad Layers scherm

Voor het importeren van layers is er een optie in het CAD Layers scherm. Er is hier echter geen export knop terwijl dit hier best een optie had kunnen zijn. Deze knop is onder het file menu geplaatst en export de layermap (zie figuur 2). Dit is inconsistent ten opzichte van de andere import/export mogelijkheden.

Gedurende de analyse van de export is duidelijk geworden dat de export niet veel anders doet dan een specifiek onderdeel exporteren. Ook bestaat de mogelijkheid om het hele project te exporteren. Hierbij wordt gespecificeerd naar welke file moet worden geëxporteerd. Dit bestand wordt vervolgens gevuld met de informatie die moet worden geëxporteerd. Er wordt voor een complete project export aangegeven of het gelukt is of niet.

De gebruikers geven aan dat de export niet zo zeer een probleem is maar dat het in combinatie met deze import wel voor problemen zorgt. De problemen die de gebruikers hebben tijdens het importeren/exporteren zijn tijdens gesprekken aan het licht gekomen en vertalen zich in de volgende handelingen:

Project	Actie
Leeg project	Import bestaand project.
Leeg project	Hergebruik elementen uit ander project.
Bestaand project	Verbeter op basis van ander project.
Bestaand project	Alles importeren dat geen conflicten geeft.
Bestaand project	Alles hernoemen dat conflicten geeft.
Bestaand project	Kies welke elementen (als elementen samenhangen dan worden die andere ook meegenomen).
Bestaand project	Importeren van alle patterns (zonder de layers, die moeten op unspecified komen voor patterns).

Dit zijn actie die de gebruikers het meeste uitvoeren en hierbij ondervinden ze problemen met het huidige import proces. Hierbij is het opvallend dat de gebruiker zo veel mogelijk conflicten uit de weg probeert te gaan. En wordt een duidelijk prioriteit aangegeven in de vorm van patterns en layers.

Om deze acties te kunnen uitvoeren moet er kunnen worden gekozen tussen wat wel en niet wordt geïmporteerd. Hierbij moet wel rekening worden gehouden met de relaties die mogelijk bestaan tussen projectelementen.

5.1.2. Conclusie

Gezien de export doet wat er verwacht wordt van een export kan gesteld worden dat het probleem zich voornamelijk bevindt in het import proces. Vanuit de analyse is duidelijk geworden dat er in dit import proces niet altijd de mogelijkheid is om de juiste import keuzes te maken.

Dit zorgt ervoor dat er veranderingen moeten worden gemaakt door de gebruikers aan het export bestand (waardoor het wel juist wordt geïmporteerd). Het verbeteren van het import proces zal er dan ook voor kunnen zorgen dat dit niet meer nodig is. Dan is het export proces geen probleem meer.

Om deze verbeteringen te kunnen maken is het van belang om te kunnen kiezen welke elementen wel en niet worden geïmporteerd. Dit zou er voor kunnen zorgen dat het import proces beter verloopt. Deze verbeteringen zouden moeten inspelen op de scenario's die aangegeven zijn door de gebruikers.

Door te bepalen op welke plekken de import gebruikt wordt is een beeld gevormd van welke projectelementen er zijn binnen iDRM. Dit kan van pas komen bij de analyse van de huidige structuur.

5.2. Analyse huidige project structuur

Na het uitvoeren van een analyse van de huidige situatie kan worden bepaald hoe de huidige project structuur in elkaar zit. Dit geeft mogelijk een abstracte basis van de huidige import/export architectuur. Deze basis zal op een later punt kunnen worden uitgewerkt naar een technische benadering die een deelvraag beantwoordt.

Het uitvoeren van deze analyse zal gebeuren door te beschrijven hoe een projectstructuur in elkaar zit. Dit gebeurt op basis van bestanden die het project heeft en de samenhang die hierin aanwezig is. Deze samenhang kan worden afgeleid uit het programma (iDRM) dat wordt gebruikt door het bedrijf. Tevens zullen de medewerkers van NP-Komplete Technologies hier nog meer informatie over kunnen verschaffen. Ook zal hier duidelijk worden welke import en export functionaliteit er momenteel in het programma zit en voor welke projectelementen dit van toepassing is. Omdat er geen overzicht is van hoe een projectstructuur in elkaar zit is gekozen voor deze aanpak.

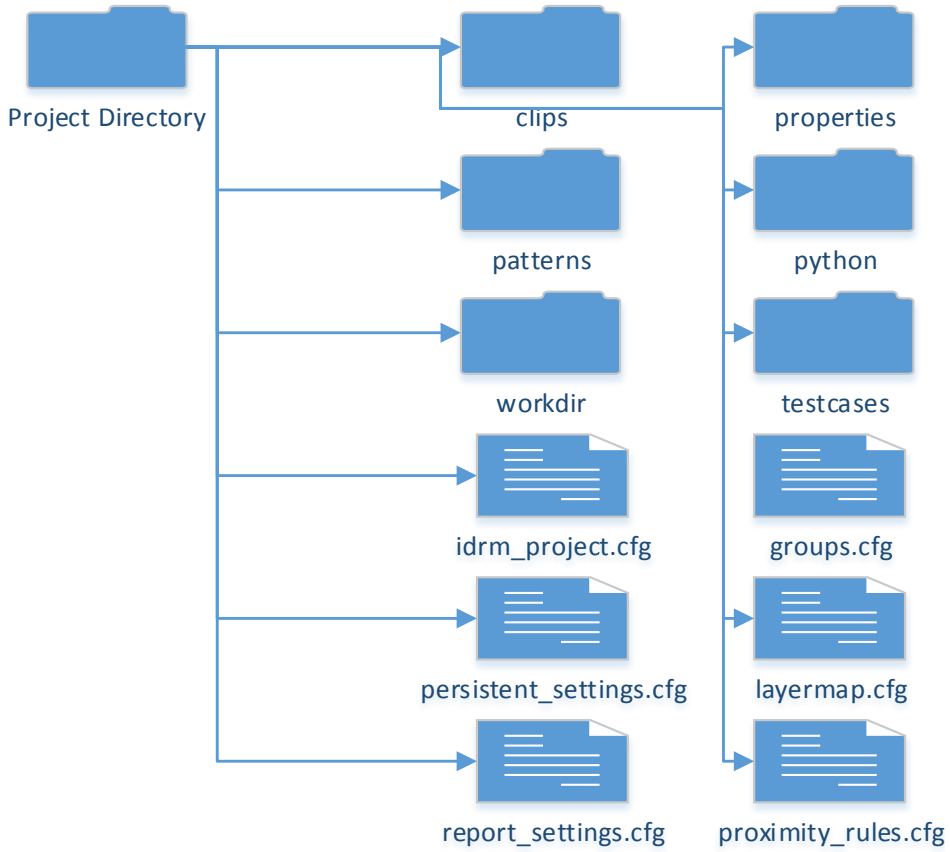
Door het maken van deze analyse zal er de mogelijkheid zijn om de functionele eisen aan het systeem verder te benadrukken. Dit kan door de begrippen en samenhang te gebruiken die zijn gevonden door deze analyse.

Hoewel de technische samenhang gedurende deze analyse nog niet helemaal duidelijk zal zijn kan dit verder worden uitgewerkt door in het technisch ontwerp deze elementen te vergelijken met de daadwerkelijke code.

Uiteindelijk zal door middel van deze analyse een diagram worden gemaakt dat weergeeft hoe elementen aan elkaar verbonden zijn. Ook zal dit een tabel opleveren waarin per project element groep is weergegeven welke import/export methode er voor is. Deze taak zal de derde deelvraag beantwoorden. De technische representatie wordt in een latere analyse beantwoord.

5.2.1. Analyse bestanden

Om deze analyse uit te voeren is een project opgesteld waarin de volledige project structuur wordt gebruikt:



figuur 3 - Projectstructuur

Bestanden die eindigen op cfg zijn data bestanden. De cfg bestanden en folders moeten worden meegenomen tijdens het import proces. Om verdere duidelijkheid te krijgen over de structuur die hier is beschreven is voor elk van de subfolders een nieuwe diagram worden opgesteld. Deze zijn terug te vinden in het functioneel ontwerp (bijlage 2).

GDS files worden gebruikt in verschillende subfolders om een stukje van een chip layout in te bewaren.

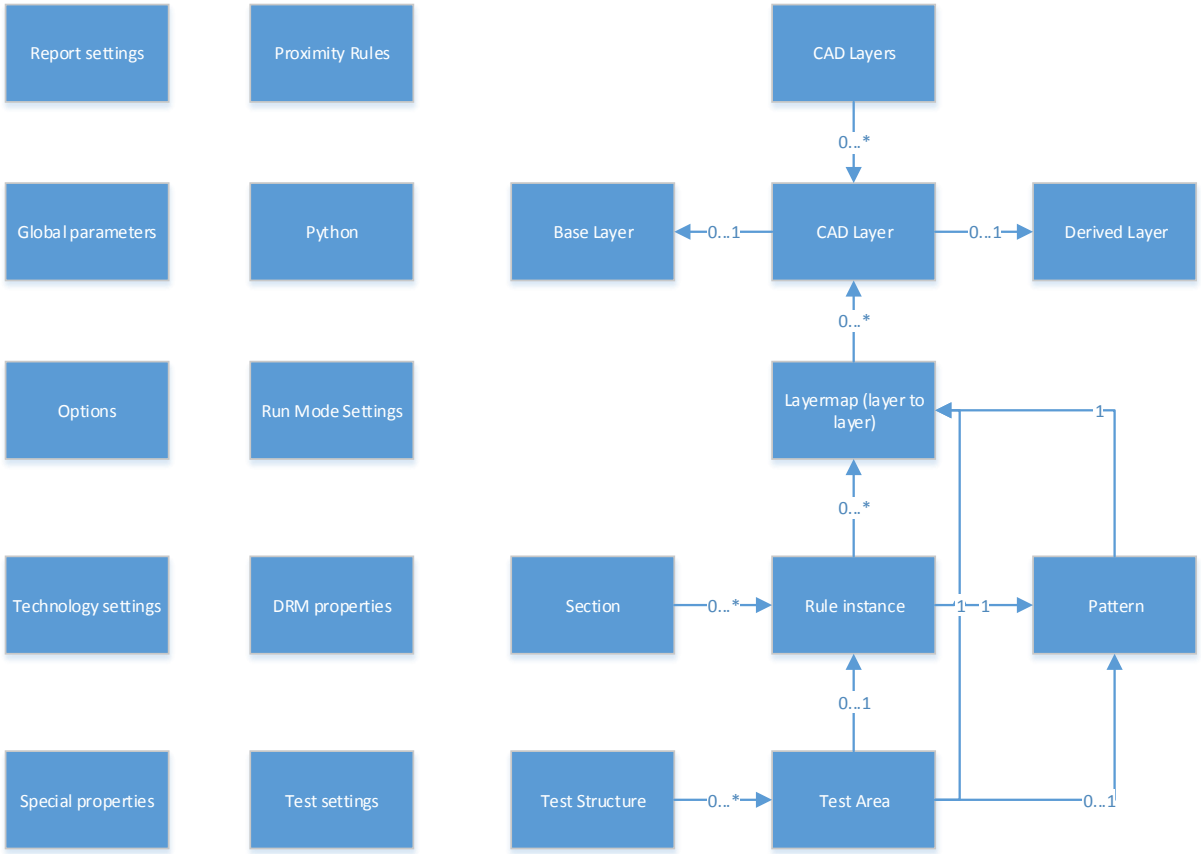
Er is per folder gekeken naar waar deze in het programma voorkomt en hoe het werkt. Dit geeft een beter beeld van hoe er kan worden uitgebreid en wat er beter kan. De workdir map is momenteel leeg (hier worden bestanden neergezet die tijdens het programma worden aangemaakt en gebruikt).

Verdere specifieke bevindingen van de analyse zijn terug te vinden in het functioneel ontwerp.

5.2.2. Samenhang

Op basis van de gesprekken die gevoerd zijn met zowel de klant als het bedrijf is een diagram opgesteld waarin alle afhankelijkheden staan die te maken hebben met import/export.

Dit diagram is door de analyse van de inhoud van de bestanden verbeterd. Het diagram geeft een weergave van de samenhang van de bestanden in de projectstructuur.



figuur 4 – Samenhang projectelementen

Momenteel worden bijna al deze elementen gebruikt binnen het programma. De import en export functionaliteit van deze onderdelen verschilt. Om duidelijkheid te maken wat er momenteel mogelijk is met de bestaande elementen is een tabel opgesteld.

Hierbij is gekeken of het element binnen het programma import/export functionaliteit heeft of dat het enkel bij het gehele project gebeurt. Ook is gekeken hoe het importeren van de bestanden werkt. Zo is het mogelijk dat bepaalde instellingen alleen worden ingeladen bij een leeg project.

Indien dit als nee is opgegeven werkt de import/export bij elk project (leeg of met elementen). Een belangrijk punt in dit diagram is het pattern. Het pattern is eigenlijk een rule instance die een pattern heeft. Dit kan echter samen worden gezien als een pattern voor het importeren/exporteren.

Onderwerp	Import in UI	Export in UI	Import/Export door project	Leeg project
CAD Layers (Layers algemeen)	Ja	Ja	Ja	Nee
Layermap	Ja	Nee	Ja	Nee
Section	Nee	Ja	Ja	Nee
Rule instance	Nee	Ja	Ja	Nee
Pattern	Nee	Ja	Ja	Nee
Test structure	Nee	Ja	Ja	Nee
Test area	Nee	Ja	Ja	Nee
Global parameters	Ja	Ja	Ja	Nee
Options	Nee	Nee	Ja	Ja

Technology settings	Nee	Nee	Nee	Nee
Special properties	Ja	Nee	Ja	Ja
Run Mode Settings	Ja	Ja	Nee	Ja
DRM properties	Nee	Nee	Ja	Ja
Test settings	Nee	Nee	Ja	Ja
Python	Nee	Nee	Ja	Nee
Report settings	Nee	Nee	Ja	Ja
Proximity Rules	Nee	Nee	Nee/Ja	Nee

Een opvallend punt in deze tabel is technology settings aangezien hier helemaal niets mee kan gebeuren. Dit is blijkbaar een bug omdat dit wel zou moeten worden ingeladen bij een leeg project.

5.2.3. Conclusie

Naar aanleiding van de tabel die is opgesteld en de projectstructuur die deze bevestigdt kan worden geconcludeerd dat de in de project structuur gedefinieerde elementen zijn opgenomen in de tabel. Dit beantwoord de derde deelvraag zoals vermeld in het onderdeel hoofd- en deelvragen:

> *"Welk relevante projectelementen worden binnen de IDRM import/export bewaard?"*

De relevante projectelementen die binnen de IDRM import/export worden bewaard zijn:

- CAD Layers
- Layermap
- Section
- Rule instance
- Pattern
- Test structure
- Test area
- Global parameters
- Options
- Special properties
- Run Mode settings
- DRM properties
- Test settings
- Python
- Report settings
- Proximity rules

Opvallend is dat Technology settings hierbij niet is opgenomen omdat dit niet werkt. Het samenhang diagram dat is opgesteld geeft een duidelijke basis waarmee kan worden gewerkt tijdens de technische analyse. De relaties uit dit diagram zullen tevens ondergrond kunnen bieden voor aannames die worden gemaakt in vervolgonderzoek.

5.3. Requirements

Zodra de analyses zijn verricht kan worden gestart met het beschrijven van de functionele eisen aan de nieuwe architectuur. Omdat de voorgaande taken bijdragen aan het duidelijk in kaart brengen van de begrippen en context worden deze opgenomen als eerste onderdeel van het functioneel ontwerp. Vervolgens is het van belang te bepalen welke eisen er zijn (requirements). Op basis van deze eisen kan worden bepaald welke actoren er zijn binnen dit proces. Dit zal leiden tot het ontwikkelen van use cases op basis van de eisen die er zijn.

Het in kaart brengen van de functionele eisen gebeurt door te overleggen met de opdrachtgever en klant. De opdrachtgever zal uiteindelijk bepalen welke onderdelen opgenomen moeten worden en welke lagere prioriteit hebben. Dit gaat ook op voor het use case diagram indien er meerdere actors blijken te zijn. Vanuit deze requirements kan vervolgens worden bepaald welke use cases er zijn.

Zodra er duidelijkheid is over wat er gemaakt moet worden kunnen er use cases worden gemaakt. Deze bevatten scenario's, activity diagrammen en een schets van hoe de user interface er minimaal uit moet zien om dit te kunnen uitvoeren. Door middel van deze aanpak kan er worden getest op basis van de use case paden en daarom is er ook besloten het op deze manier te behandelen.

Dit zal uiteindelijk de volgende onderdelen kunnen opleveren:

- Requirements
- Functionele eisen
- Actors
- Use case diagram
- Use cases

Deze onderdelen zullen samen met de analyses een functioneel ontwerp vormen. Dit document zal een basis bieden voor het technisch ontwerp. Ook zal een basis worden geboden voor het beantwoorden van de sub vraag van de eerste deelvraag.

5.3.1. Requirements en eisen

Gezien er een beperkt aantal scenario's bekent zijn vanuit de analyse van de huidige structuur is er besloten om in eerste instantie te bepalen welke requirements er zijn vanuit de gebruikers. Het vergaren van deze requirements in opgedeeld in drie fases (Christel, 1992):

1. Requirements eliciting
2. De requirements analyseren
3. Requirements vastleggen

Op basis van deze fasering is in eerste instantie een scope afgesproken waarin gewerkt zal worden. Deze afdekking neemt de veranderingen in het huidige systeem maar in beperkte mate mee. Als er gedurende het project grote veranderingen worden gemaakt worden deze niet meegenomen.

De scope van het project beperkt zich tot het importeren en exporteren van een geheel project en de hieraan gerelateerde functionaliteiten. Na overleg met de gebruikers zijn de volgende requirements opgesteld naar aanleiding van de eerder gestelde problemen:

- Er moet een gebruikersinterface zijn voor het importeren en exporteren van projectelementen.
- Er moet per project element een keuze zijn voor de manier waarop het wordt geïmporteerd in het geval er een conflict is (origineel behouden, overschrijven, hernoemen, toevoegen aan het project).
- Per project element moet de optie bestaan deze wel of niet te importeren/exporteren.
- Bij het importeren van patterns moet het mogelijk zijn zonder layers te importeren.
- Bij het importeren van rule instances moet het mogelijk zijn zonder layers te importeren.
- Bij het uitschakelen van een element tijdens het importeren moet het mogelijk zijn de hieraan verbonden elementen snel te kunnen veranderen door een andere link te maken voor deze objecten.
- Zoekfunctie voor het filteren van projectelementen.
- Het programma mag niet crashen bij normaal gebruik (uitgaande dat er hier altijd een redelijke hoeveelheid geheugen is).
- De gebruikersinterface moet fout ontwijkend zijn.
- Binnen de interface moet voor de gebruiker snel duidelijk zijn hoe er gewerkt moet worden met de interface.

- De interface mag niet gebruik maken van elementen die de testbaarheid van de interface in gevaar brengen.
- De code moet robuust zijn en berekend op fouten vanuit de gebruiker.

Gezien dit vrij uiteenlopende eisen zijn is er belang bij prioriteiten. Deze prioriteiten zijn opgesteld in overleg met de bedrijfsbegeleider en leiden tot een concrete fasering (kies opties op basis van prioriteiten).

Deze prioriteiten zijn de functionele eisen en zijn opgesteld op basis van de MoSCow methode (MoSCoW Method, 2015). De volgende eisen zijn als must-have opgesteld:

- Er moet een gebruikersinterface zijn voor het importeren van projectelementen (Rule instance(s), Layer(s) en Pattern(s)).
- Er moet per project element een keuze zijn voor de manier waarop het wordt geïmporteerd in het geval er een conflict is (origineel behouden, overschrijven en hernoemen).
- Per project element moet de optie bestaan deze wel of niet te importeren.
- Bij het importeren van patterns moet het mogelijk zijn zonder layers te importeren.
- Bij het importeren van rule instances moet het mogelijk zijn zonder layers te importeren.
- Bij het uitschakelen van een element tijdens het importeren moet het mogelijk zijn de hieraan verbonden elementen snel te kunnen veranderen door een andere link te maken voor deze objecten.
- Het programma mag niet crashen bij normaal gebruik (uitgaande dat er hier altijd een redelijke hoeveelheid geheugen is).
- De interface mag niet gebruik maken van elementen die de testbaarheid van de interface in gevaar brengen (Elk interface object moet een object naam hebben).

Voor de should-have geldt dat er om een aantal handigheden wordt gevraagd zowel als extra mogelijkheden voor het importeren.

- Er moet een gebruikersinterface zijn voor het importeren van projectelementen (Rule instance(s), Layer(s), Pattern(s) en Test area(s)).
- Bij het importeren van test area's moet het mogelijk zijn te importeren zonder rule instance(s).
- De gebruikersinterface moet fout ontwijkend zijn.

De could-have heeft voornamelijk betrekking op het bevatten van alle import elementen. Ook wordt hier de nadruk gelegd op onderdelen die handig zijn voor het gebruik:

- Groeperingsmechanisme voor de projectelementen.
- Binnen de interface moet voor de gebruiker snel duidelijk zijn hoe er gewerkt moet worden met de interface.
- De code moet robuust zijn en berekend op fouten vanuit de gebruiker.

De won't-have betreft het exporteren en linken van het project.

5.3.2. Actors

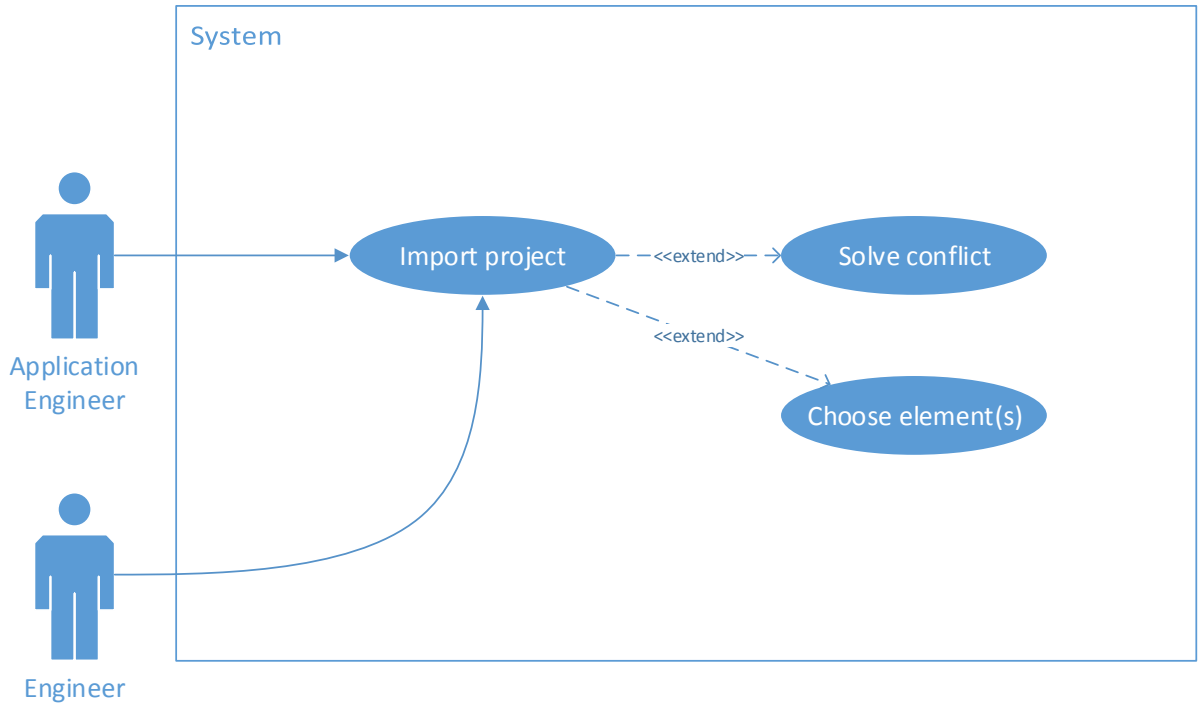
Er zijn een aantal actors binnen dit project. Zo is de klant een actor maar ook de medewerkers van NP-Komplete Technologies (AE's).

Naam:	Application engineer
Rol:	Gebruiker
Beschrijving:	Een application engineer is een medewerker van NP-Komplete technologies. Deze application engineer gaat om met de klanten en geeft demo's. Tevens geven deze gebruikers cursussen en werken soms zelfs bij de klanten. Ze zijn de best voorgelichte gebruikers op het gebied van het programma.

Naam:	Engineer
Rol:	Gebruiker
Beschrijving:	Een engineer is iemand met een technische opleiding en enige ervaring met het programma. Deze gebruikers zijn meestal verantwoordelijk voor het gebruik van het programma binnen de organisatie van de klant en werken hier ook.

Deze actoren zijn vervolgens gekoppeld aan de eerder vastgelegde requirements. Dit is gedaan om afdekking te hebben voor specifieke functionaliteit die een bepaalde groep wel zou mogen gebruiken.

Het UML Use case diagram (Ambler, 2014) dat is opgesteld is als volgt:



figuur 5 – UML Use case diagram

Er is hierbij gebruik gemaakt van een vrij abstracte laag als het zou worden vergeleken met de vereiste functionaliteit. Hier is voor gekozen om er voor te zorgen dat er nog genoeg vrijheid is voor de gebruikersvriendelijke kant van het project.

De gebruikers staan los met vergelijkbare functionaliteit omdat de meningen verschilden gedurende de gesprekken die er zijn geweest. Omdat zowel de engineers als application engineers behoren tot de gebruikers is het handig om deze te splitsen zodat er later de mogelijkheid is te kijken of er verbanden zijn omtrent de eisen die ze stellen rondom de gebruikersinterface.

5.3.3. Importeren van een project

Er zijn gedurende het project drie use cases opgesteld. Deze zijn terug te vinden in het functioneel ontwerp. De volgende use cases zijn opgesteld:

- Import project
- Solve conflict
- Choose element(s)

Deze use cases zijn opgebouwd uit een specificatie, een activity diagram en een schermontwerp. De specificatie is opgebouwd op basis van een template (Cockburn, 2001) maar heeft een aantal afwijkingen.

Het schermontwerp is op een vrij abstract niveau en geeft enkel een weergave van een indeling die alle functioneel vereiste mogelijkheden bevat.

5.3.4. Conclusie

Het functioneel ontwerp dat is ontwikkeld tijdens deze fase bevat de onderdelen die nodig zijn om door te werken aan een technische ontwerp. Dit zorgt ervoor dat er een daadwerkelijke architectuur zal kunnen worden gemaakt.

De analyses die zijn opgenomen in het functioneel ontwerp geven tevens een vrij abstract antwoord op de subvraag van de eerste deelvraag. Deze kan en zal verder moeten worden uitgewerkt om de vraag te beantwoorden.

Het use case diagram geeft aan dat de actors vrijwel dezelfde rechten hebben. Deze zijn los gebleven omdat ze tijdens de gesprekken niet altijd dezelfde mening deelden en wellicht apart moeten worden behandeld bij het testen van de gebruikersinterface.

Nu het functioneel ontwerp is gemaakt kan worden doorgewerkt naar een specifiekere beschrijving van hoe de functionaliteit er uit moet zien. Zo kan worden gestart met de technische analyses en vervolgens het technisch ontwerp.

5.4. Analyse van de huidige project import

Om te begrijpen waar verbeteringen moeten worden aangebracht is het essentieel om te begrijpen waar moet worden ingehaakt in de bestaande code. Om dit te doen zal een analyse moeten worden gemaakt van hoe de huidige project import/export werkt. Dit zal er uiteindelijk voor kunnen zorgen dat er een klassendiagram kan worden opgesteld en dat hierin verbeteringen kunnen worden gemaakt. Hierbij kan gebruik worden gemaakt van de bevindingen die in het functioneel ontwerp staan.

Deze analyse kan worden uitgevoerd door in de huidige code te zoeken vanuit alle gevonden import opties. Deze opties zijn bekend vanuit de tabel die is gemaakt van de analyse van de huidige project structuur. Door alle in dit proces gevonden geassocieerde elementen in een UML Klassendiagram (Class Diagram, 2015) te zetten kan worden weergegeven van waaruit gewerkt wordt. Vervolgens kan de gemaakte analyse uit het functioneel ontwerp worden gebruikt om te kijken hoe de code zich verhoudt tot de project structuur. Er is voor deze aanpak gekozen omdat er geen documentatie is over hoe de code werkt maar wel dat er import functionaliteit is. Deze functionaliteit kan gebruikt worden om deze punten te vinden in de code.

Dit zal uiteindelijk een klassendiagram opleveren van waaruit een duidelijk beeld wordt gegeven van welke elementen er worden gebruikt in het programma. Dit zorgt er uiteindelijk voor dat er een diagram is van hoe de huidige import/export architectuur in elkaar zit binnen iDRM. Dit beantwoordt

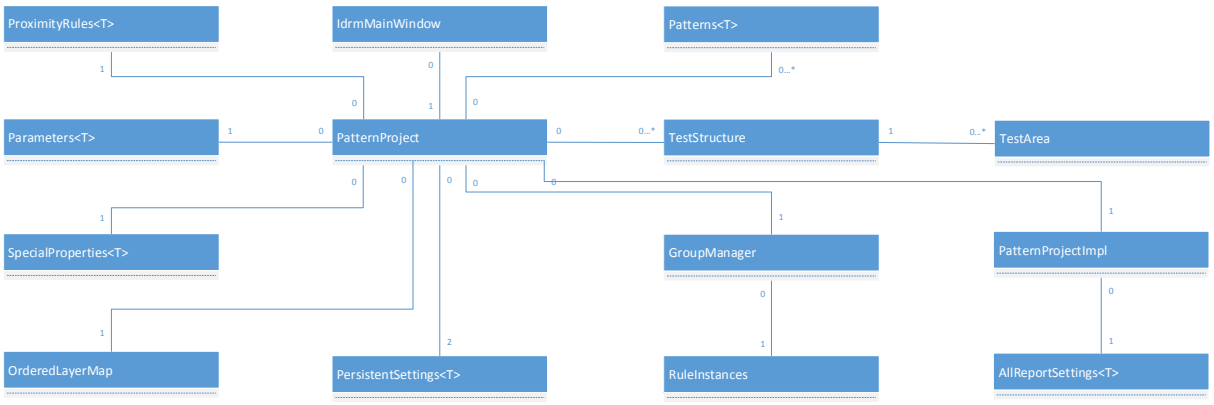
dan de sub vraag van de eerste deelvraag. Ook zal er duidelijkheid zijn over welke relevante projectelementen worden gebruikt in de import/export van iDRM wat de derde deelvraag beantwoordt.

5.4.1. Bevindingen

Tijdens deze analyse is gekeken naar hoe de daadwerkelijk relaties in de code zich verhouden tot het opgestelde relatie diagram. Om dit punt te vinden in de code is gewerkt vanuit de interface file (vanuit de import knop).

Door dit te volgen is al snel duidelijk geworden waar en hoe wordt geïmporteerd. Het diagram dat is opgesteld in het functioneel ontwerp was te abstract om direct te verwoorden in klassen. Er is op basis van de bekende concepten echter wel een vertaalslag gemaakt die terug te vinden is in het technisch ontwerp.

De technische samenhang die is ontdekt in de code is weergegeven in een UML Klassendiagram (Holub, 2011). Om te voorkomen dat het diagram te groot wordt en onoverzichtelijk is zijn de bijbehorende functies en data weg gelaten. Ook omdat deze in sommige gevallen niet bestaan.



figuur 6 – UML Klassendiagram

De notatie <T> staat hier echter niet voor de naam en ook niet voor een generic type. Aangezien de implementatie toestaat dat er type aliases (Typedef, 2015) worden gebruikt zijn deze opgenomen in het klassendiagram (dit zijn in vrijwel alle gevallen wrappers voor lijsten).

Als deze klasse namen echter in enkelvoud worden gezet is dat ook gelijk het onderliggende type (ProximityRules<T> is een lijst van ProximityRule instanties).

Vanuit dit diagram is al snel duidelijk dat de PatternProject klasse te veel verantwoordelijkheden heeft en een anti-pattern is (God Object, 2015).

De al bestaande functionaliteit loopt via deze architectuur en wordt aangestuurd vanuit de PatternProject klasse. Het is hierbij opvallend dat dit wel abstractie lagen bevat en dat er een losse reader klasse is (CfgIOHelper) waarin de daadwerkelijke data wordt ingeladen.

5.4.2. Conclusie

Gedurende deze analyse is bekend geworden dat de huidige import-/export architectuur binnen iDRM is opgebouwd uit een godklasse die alle bijbehorende elementen bevat. Het klassendiagram dat is opgesteld geeft een duidelijke weergave van hoe de huidige import/export architectuur is opgebouwd en beantwoordt daarmee de subvraag van de eerste deelvraag.

De derde deelvraag valt ook op te maken uit de analyse die is gemaakt. Zo worden de volgende elementen bijgehouden binnen de iDRM import/export:

- Pattern
- Group manager
- Rule instance
- Layer
- Persistent setting
- Report setting
- Test structure
- Test area
- Parameter
- Special property

Proximity Rule is niet opgenomen in dit overzicht omdat hier geen import/export functionaliteit voor aanwezig is. Deze bevestiging voor de derde deelvraag geeft de mogelijkheid om voor het analyseren van de conflicten.

5.5. Analyse van de mogelijke conflicten

Om te bepalen welke conflicten er mogelijk zijn bij het import/export proces moet een analyse worden gemaakt die aangeeft op welke onderdelen conflicten kunnen ontstaan. Deze conflicten kunnen dan gebruikt worden voor het verbeteren van het huidige import/export proces.

Een analyse van de mogelijke conflicten zal gebeuren door te beschrijven welke vergelijkingen er plaats vinden in de code. Er zal bepaald moeten worden welke projectelementen een conflict kunnen vormen en op welke manier dit gebeurt. Dit kan gebeuren door in de functies te kijken van de projectelementen (waar vergelijkingen voorkomen).

Ook zal moeten worden gekeken welke relaties deze projectelementen hebben met andere projectelementen. Dit kan van invloed zijn op de manier waarop geïmporteerd dient te worden.

Dit zal uiteindelijk per project element een tabel opleveren met daarin de velden en of situaties dat een conflict kan optreden en hoe deze momenteel wordt opgelost. Dit overzicht geeft dan een antwoord op de vierde deelvraag.

5.5.1. Bevindingen

Gedurende de analyse van de code is gebleken dat er een aantal factoren zijn waarvoor geldt dat ze een conflict kunnen opleveren. Zo wordt bepaald of een element uniek is door middel van de naam. Indien een naam voor een project element al bestaat is het echter nog niet direct een conflict. Een uitzondering hierop is Layer die twee waarden heeft waarop deze fout kan gaan (naam en nummer).

Pas als de inhoud op bepaalde punten afwijkt van het andere element wordt gesproken van een conflict. Indien een project element dubbel blijkt te zijn wordt deze niet geïmporteerd door het huidige import proces.

In het technisch ontwerp is terug te vinden welke specifieke conflicten kunnen ontstaan bij het importeren. De projectelementen waarmee een conflict kan ontstaan zijn als volgt:

Naam project element	Key(s) conflict(en)	Relatie(s) conflict(en)
Pattern	Naam	RuleInstance, TestArea
Layer	Naam, Nummer	Pattern, RuleInstance, TestArea
GroupManager	Naam (path moet hetzelfde zijn)	RuleInstance, TestArea
PersistentSetting	Naam	
RuleInstance	Naam	TestArea
ReportSetting	Naam	
TestStructure	Naam	
Parameter	Naam	

TestArea	Naam	TestStructure
SpecialProperty	Naam	

Tijdens de analyse is ook gebleken dat lang niet elk onderdeel een duidelijke manier had om te vergelijken. Dit toont aan dat er mogelijk nieuwe functies moeten worden toegevoegd om deze projectelementen op een consistente manier te vergelijken.

Dit zou in sommige gevallen een design verbetering kunnen zijn doordat er minder code staat in de godklasse waar het nu in gebeurt.

Zoals eigenlijk al verwacht vanuit de opdracht zijn er weinig opties gevonden rondom het importeren. Het huidige proces slaat simpelweg onderdelen over als ze een conflict veroorzaken wat tot probleem situaties kan leiden doordat de relaties met andere objecten niet worden veranderd.

Dit zorgt er dan ook voor dat het project in een corrupte staat kan eindigen. Ook is het mogelijk dat onderdelen die duidelijk dubbel zijn (maar geen conflict veroorzaken door een andere naam te hebben) gewoon geïmporteerd worden zonder een optie om dit wel of niet te doen.

5.5.2. Conclusie

Gezien de hoeveelheid mogelijke conflicten en de hoeveelheid die correct wordt behandeld kan gesteld worden dat er verbeteringen nodig zullen zijn aan de manier waarop wordt vergeleken en de manier waarop wordt geïmporteerd.

Tijdens de analyse zijn alle mogelijke conflicten die deze elementen kunnen veroorzaken bepaald. De daadwerkelijke details van de conflicten die zich kunnen voordoen en welke eisen hieraan zijn verbonden zijn terug te vinden in het technisch ontwerp.

Dit kan echter worden samengevat tot:

- Conflict op basis van Naam
- Conflict op basis van Nummer (alleen voor Layer)
- Conflict op basis van relatie met andere projectelementen

Dit beantwoord de vierde deelvraag door alle mogelijke conflicten te onderscheiden. Nu bekend is welke objecten gebruikt kunnen worden en wat er fout kan gaan is er de mogelijkheid om verbeteringen aan te brengen.

5.6. Verbeterde architectuur

Zodra de analyses uitgevoerd zijn kan er worden gewerkt aan het technisch ontwerp. Op basis van de voorgaande taken is er op dit punt genoeg duidelijkheid om een verbeterde architectuur te maken voor een import/export wizard binnen iDRM. Ook kan er een design worden opgesteld waarin alle functionele eisen terugkomen.

Door in eerste instantie de resultaten van de twee voorgaande analyses in het technisch ontwerp te stoppen is een duidelijke context aanwezig. Vervolgens kan op basis van de gestelde requirements een klassendiagram worden gemaakt. Zodra dit is gebeurd, kan er een design worden gemaakt dat aansluit op de actors die zijn bepaald in het functionele ontwerp.

Dit kan in eerste instantie gebeuren aan de hand van onderzoek op het gebied van usability. Echter staat onderzoek hierbij niet centraal maar gaat het om het gemak van de gebruiker. Het is dan ook daarom dat dit onderdeel in een latere fase kan worden verbeterd op basis van de input die de gebruikers leveren.

Dit vormt gezamenlijk het technisch ontwerp en zal antwoord geven op de eerste en tweede deelvraag. De kwaliteit van deze benadering wordt in de validatiefase bepaald.

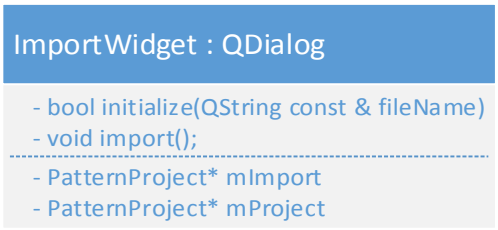
Zodra het bedrijf en de klant akkoord gaan met de afgesproken scope en requirements kan de realisatiefase beginnen.

5.6.1. Bevindingen klassendiagram

Op basis van de gevonden architectuur en de wensen van de gebruikers kan worden gesteld dat er behoefte is aan een interface voor het importeren van projectelementen. Deze projectelementen moeten bereikbaar zijn vanuit deze interface. Om ervoor te zorgen dat er niet nog een godklasse ontstaat kunnen instanties worden gebruikt van PatternProject (project klasse).

Dit zorgt ervoor dat er eenvoudig onderscheid kan worden gemaakt tussen het import project en het huidige project. Er is echter tijdens de analyse duidelijk geworden dat de huidige code in deze klasse referenties bevat naar de interface die niet beschikbaar zullen zijn voor de import data. Dit heeft dan ook als nadeel dat er sommige stukken uit de bestaande code zullen moeten worden veranderd (soms misschien zelfs redundant).

Hoewel dit ervoor kan zorgen dat er meer code bij PatternProject zal komen zorgt het er ook voor dat er een correcte opdeling is als wordt uitgegaan van het MVC pattern (Model View Controller, 2015).

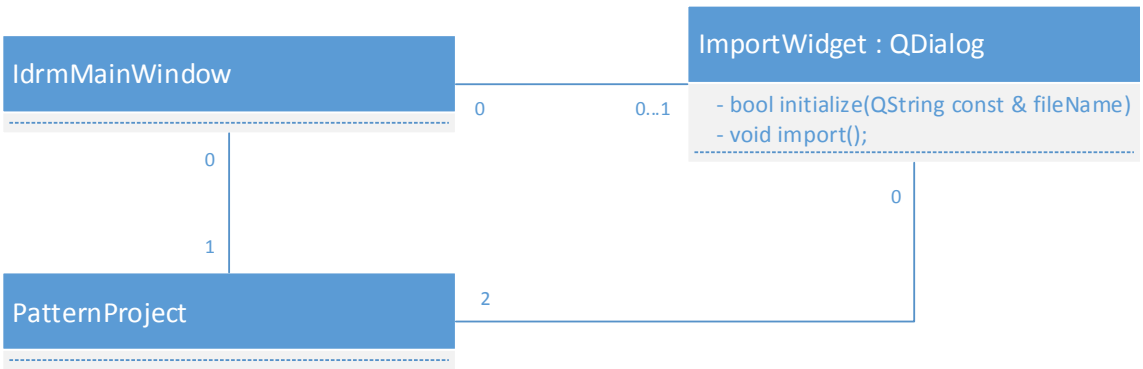


figuur 7 – Klasse voor de import

Er wordt hier gebruik gemaakt van pointers om te voorkomen dat de gigantische hoeveelheid data in de klassen wordt gekopieerd. Er kan in de daadwerkelijke implementatie echter beter gebruik worden gemaakt van smart pointers om ervoor te zorgen dat er geen memory leaks zijn.

Deze klasse wordt aangeroepen vanuit de interface van het programma en krijgt op die plek ook de pointers naar de projecten. Dit zorgt ervoor dat de import niet meer in PatternProject zit en dat er in de toekomst mogelijk makkelijker kan worden opgesplitst.

Dit zorgt ervoor dat er uiteindelijk het volgende UML Klassendiagram ontstaat:



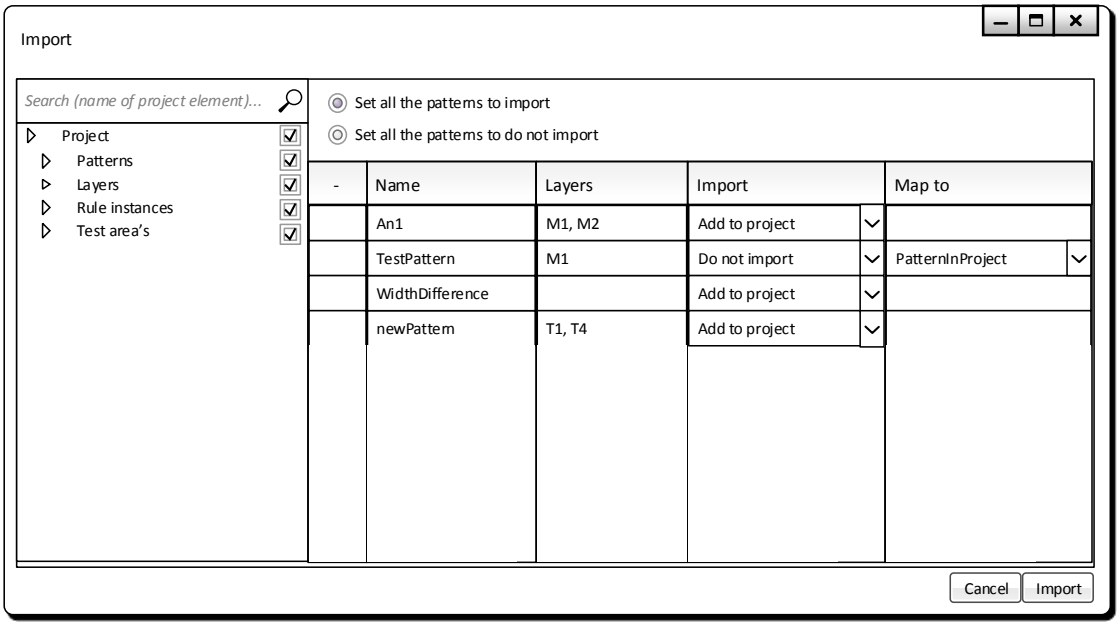
figuur 8 – Verbeterde architectuur

5.6.2. Bevindingen design

Het design van de import widget is verbeterd door het uitvoeren van gebruikers testen. De resultaten van deze testen zijn terug te vinden in het testrapport. De huidige en voorgaande versies van het design kunnen worden teruggevonden in het technisch ontwerp.

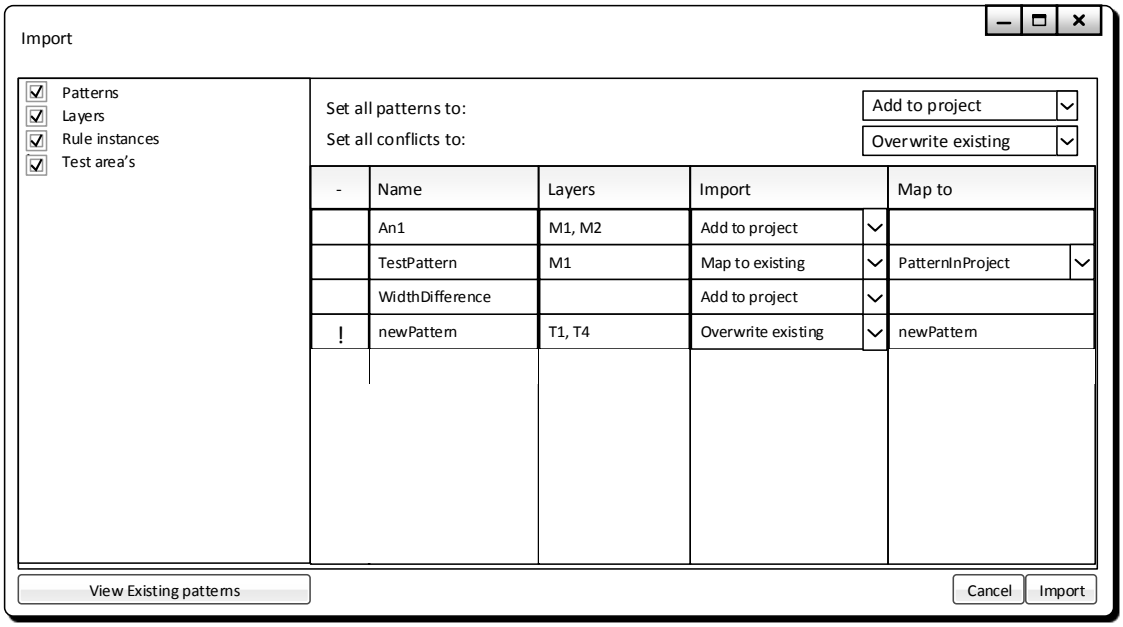
In eerste instantie is gekeken naar welke overeenkomsten er zijn in andere import widgets en hoe dit van toepassing kan zijn in het design van de import widget. Ook is gekeken naar literatuur die toelicht welke afwegingen er kunnen worden gemaakt ten opzichte van specifieke widgets.

Gedurende het proces is besloten om de designs te maken met wireframe diagrammen zodat er makkelijk nog iets veranderd kan worden. Het eerste design is gebaseerd op de informatie die is verzameld vanuit de literatuur en de functionele requirements.



figuur 9 – Eerste design

Op basis van de testen die zijn uitgevoerd en de feedback die is gegeven door de gebruikers is echter een ander design ontstaan. Het uiteindelijke design dat is gemaakt is:



figuur 10 – Verbeterde design

De veranderingen hebben zich zowel op de functionaliteit als de gebruikersvriendelijkheid van de interface gericht en hebben daardoor tot een beter resultaat kunnen leiden.

De feedback van de tweede iteratie is opgenomen in dit laatste design en geeft een betere weergave van de toekomstige uitstraling van de import widget (iteratie drie).

5.6.3. Conclusie

Op basis van de gemaakte bevindingen kan geconcludeerd worden dat een verbeterde architectuur voor een import widget mogelijk bestaat uit slechts een interface en een uitbreiding van de bestaande code. Indien deze architectuur de functionele eisen weet te beantwoorden dan is deze architectuur het antwoord op de eerste deelvraag die is opgesteld.

Op de vraag: “Hoe kan een import/export wizard op een voor de doelgroep gebruikersvriendelijke manier worden getoond binnen iDRM?” is moeilijk antwoord te geven. Er is samen met de gebruikers gewerkt aan het verbeteren van de interface en dit heeft mogelijk geleid tot een import widget die gebruikersvriendelijk is.

Deze vraag zal echter enkel bevestigd worden door de gebruikerstesten. Indien deze uitwijzen dat deze interface gebruikersvriendelijk is voor de doelgroep kan gesteld worden dat de interface op een gebruikervriendelijke manier getoond kan worden binnen iDRM voor de doelgroep. Hoe dit gebeurt is dan ook door met behulp van de gebruikers de interface te verbeteren.

Met behulp van het technisch ontwerp en het functioneel ontwerp kan gewerkt worden aan de realisatiefase.

6. Realisatiefase

Op basis van wat er is gemaakt en bevonden in de onderzoeksfase kan worden gewerkt aan een daadwerkelijke implementatie. Gedurende meerdere iteraties worden functionaliteiten zoals beschreven in het functioneel ontwerp en technisch ontwerp uitgewerkt.

Dit gebeurt op basis van de prioriteit die is aangegeven door het bedrijf. Door deze prioriteit te handhaven zal er tijdens de ontwikkeling eerst aan functionaliteit worden gewerkt die voor het bedrijf van groot belang is. Ook zal hierdoor sneller een product ontstaan die het oude proces kan vervangen.

In dit onderdeel wordt per iteratie gekeken naar welke functionaliteit er is opgenomen in de import widget.

6.1. Import widget

De import widget wordt gemaakt zodat deze kan worden bevestigd. Ook wordt deze gemaakt omdat dit een vraag is vanuit de opdrachtgever. Dit gebeurt door in eerste instantie het klassendiagram te implementeren.

Op basis van deze structuur kan vervolgens gewerkt worden naar de requirements die zijn gesteld (dit gebeurt op basis van de use cases). Er wordt in deze fase dus een meetbaar product gemaakt waarmee de eerste deelvraag mogelijk kan worden beantwoord.

Uiteindelijk zal er een import widget worden opgeleverd die is opgenomen in iDRM. Zodra de import widget is gemaakt kan de validatiefase starten.

6.1.1. Iteratie 1

Tijdens de eerste iteratie is eerst een opdeling gemaakt van wat er gemaakt moet worden. Vervolgens is een volgorde opgesteld waarin specifieke onderdelen gemaakt moeten worden. Er is besloten om eerst de must-have functionaliteiten, zoals beschreven in het functioneel ontwerp, te verwerken.

Vervolgens is er een volgorde aangebracht:

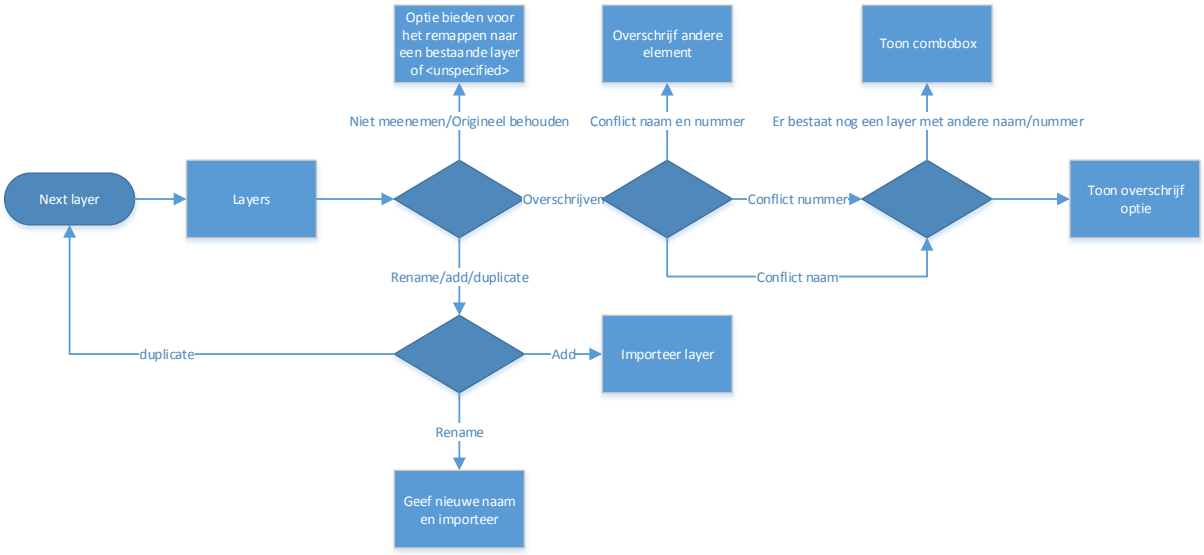
- Er moet een gebruikersinterface zijn voor het importeren van projectelementen (Rule instance(s), Layer(s) en Pattern(s)).
- Per project element moet de optie bestaan deze wel of niet te importeren.
- Bij het uitschakelen van een element tijdens het importeren moet het mogelijk zijn de hieraan verbonden elementen snel te kunnen veranderen door een andere link te maken voor deze objecten.
- Er moet per project element een keuze zijn voor de manier waarop het wordt geïmporteerd in het geval er een conflict is (origineel behouden, overschrijven en hernoemen).
- Bij het importeren van patterns moet het mogelijk zijn zonder layers te importeren.
- Bij het importeren van rule instances moet het mogelijk zijn zonder layers te importeren.
- Het programma mag niet crashen bij normaal gebruik (uitgaande dat er hier altijd een redelijke hoeveelheid geheugen is).
- De interface mag niet gebruik maken van elementen die de testbaarheid van de interface in gevaar brengen (Elk interface object moet een object naam hebben).

In deze volgorde is gestart met werken. In eerste instantie zijn wel alle headers toegevoegd aan de treeview maar zijn enkel voor de aangegeven onderdelen interfaces geïmplementeerd.

Het maken van de interface bleek vrij weinig werk te zijn en het weergeven van de data was niet erg moeilijk. De grote hoeveelheid permutaties die er waren voor het importeren gaven echter problemen.

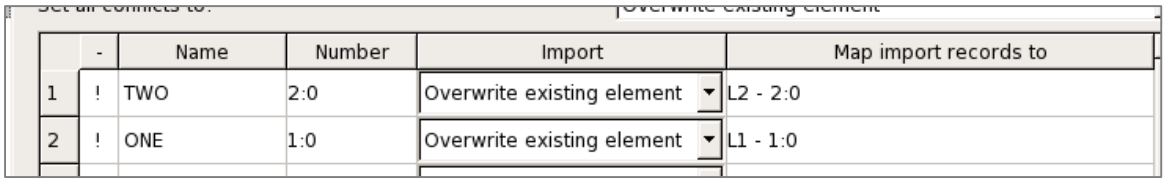
Zo zitten de projectelementen uit de must have functionaliteit allemaal aan elkaar vast. Dit zorgt ervoor dat als een element niet wordt meegenomen er een hele rij met dingen moet gebeuren.

Layers zorgden hierbij voor de grootste problemen aangezien een layer meerdere keys heeft waarop het een conflict kan vormen. Om dit probleem te visualiseren is het volgende diagram opgesteld:



figuur 11 – Layer import diagram

Hoewel dit op deze manier goed is te overzien kwam er nog bij dat elementen die aan deze layer vastzaten vervolgens ook aangepast moesten worden. Dit gebeurde voor vrijwel alle elementen uit de must have.



	-	Name	Number	Import	Map import records to
1	!	TWO	2:0	Overwrite existing element	L2 - 2:0
2	!	ONE	1:0	Overwrite existing element	L1 - 1:0

figuur 12 – Layer import mogelijkheid

Gezien dit probleem nog vrij beperkt was met drie mogelijke types viel het in dit stadium nog mee met de hoeveelheid mogelijkheden. Dit zal echter niet zo zijn als de tests er ook nog bij moeten.

Het programmeren zelf verliep vrij soepel aangezien er al vrij veel kennis rondom de taal en omgeving was. Een aantal van de must have functionaliteiten kon dan ook vrij eenvoudig worden geïmplementeerd.

Er bleek gedurende het proces echter dat de patterns een grafische weergave van het object hebben. Deze weergave zat direct vast aan de interface en de engine waardoor het erg veel tijd zou nemen om dit goed te implementeren. Er is samen met de bedrijfsbegeleider besloten een mogelijke verbetering niet te implementeren aangezien dit te veel tijd zou kosten.

6.1.2. Iteratie 2

In de tweede iteratie is voornamelijk aandacht besteed aan het uitwerken van de should-have functionaliteiten uit het functioneel ontwerp:

- Er moet een gebruikersinterface zijn voor het importeren van projectelementen (Rule instance(s), Layer(s), Pattern(s) en Test area(s)).
- Bij het importeren van test area's moet het mogelijk zijn te importeren zonder rule instance(s).
- De gebruikersinterface moet fout ontwijkend zijn.

Hierbij is het verschil dat tests moeten worden toegevoegd. Om deze tests toe te voegen is een vergelijkbare aanpak gekozen als in de eerste iteratie. Er is eerst gewerkt vanuit de interface en vervolgens is de daadwerkelijke functionaliteit opgenomen.

Dit bleek zoals voorzien een behoorlijk groot probleem te zijn voor de rule instances. Voornamelijk omdat een rule instance een pattern moet hebben. Dit betekent dat als er niets in het project zit en er worden geen patterns meegenomen vanuit het import project dat de rule instances niet correct zijn (omdat ze geen pattern hebben).

Toen dit scenario echter was afgevangen was de functionaliteit vergelijkbaar met die van de voorgaande elementen en berekend op fouten van de gebruiker.

Omdat er toch een product wordt opgeleverd waar gebruik van gaat worden gemaakt is besloten om de overige onderdelen ook te importeren. Dit heeft ervoor gezorgd dat de python hooks en de global parameters ook in de interface zijn opgenomen en dezelfde functionaliteiten hebben als de andere elementen in de import widget. De overige elementen worden ook meegenomen maar dit gebeurt door middel van de oude import.

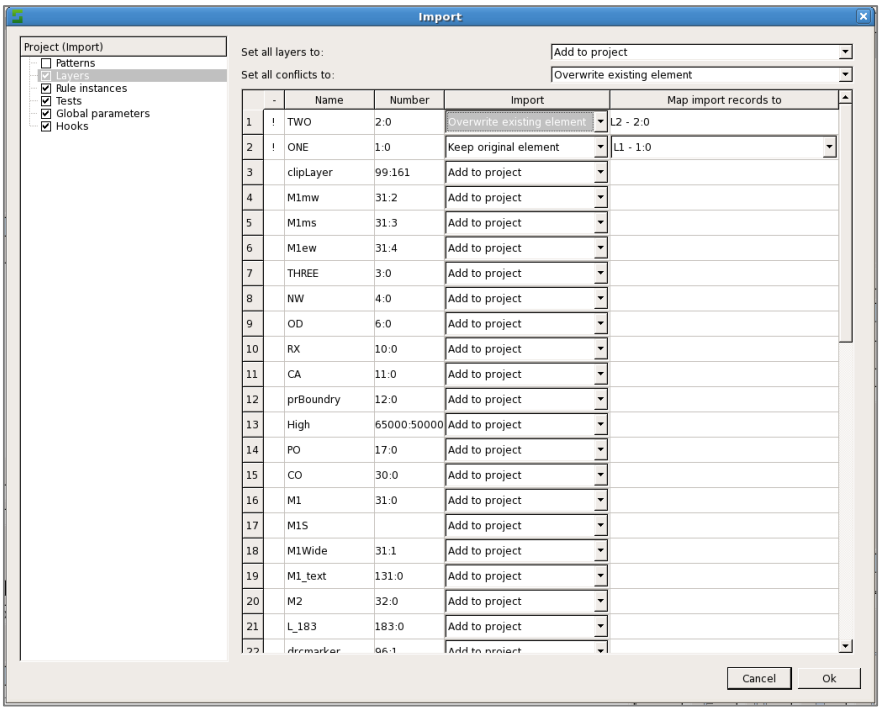
Dit heeft er uiteindelijk voor gezorgd dat de import volledig is en doet wat de naam "Project import" suggereert.

Om ervoor te zorgen dat de interface fout ontwijkend werkt is er besloten altijd de mogelijkheid te hebben om op importeren te drukken. Dit houdt in dat alle elementen die worden geïmporteerd automatisch bij het openen van het scherm een correcte waarde krijgen die ervoor zorgt dat een import altijd goed verloopt.

Dit is ook doorgevoerd in de acties die de gebruiker kan uitvoeren in de interface. Zo kan er geen situatie worden gemaakt waarin de relaties tussen objecten fout gaan.

6.1.3. Conclusie

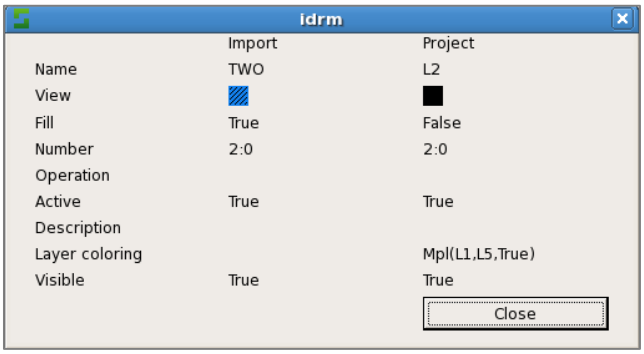
Uiteindelijk is er een import widget gemaakt die de functionaliteiten heeft om de must-have en should-have eisen te beantwoorden. Hoewel de samenhang tussen de projectelementen zorgde voor soms problematische scenario's zijn alle projectelementen in de import widget opgenomen.



figuur 13 – Import interface

Extra informatie over een object en mogelijk het object in het project is weergegeven in een detail scherm dat zichtbaar wordt indien er op een rij twee maal wordt geklikt.

Ook zijn verschillende opties toegevoegd die ervoor gezorgd hebben dat acties die de gebruikers willen uitvoeren eenvoudig kunnen worden uitgevoerd.



figuur 14 – Detail scherm

7. Validatiefase

Gedurende de iteraties zijn er verschillende functionaliteiten uitgewerkt. Om te kunnen bepalen of deze functioneren zoals beschreven zijn er testen uitgevoerd. Met behulp van de bevindingen van deze testen kan uiteindelijk worden gewerkt naar een antwoord op de hoofdvraag.

Het testen is opgedeeld in het testen van de architectuur en het testen van de userinterface. Dit is gedaan omdat het andere soorten testen oplevert. Ook zijn er andere bevindingen gezien de analyse van de resultaten varieert per test type.

7.1. Validatie architectuur

Tijdens de validatiefase zal in eerste instantie worden getest of het product dat in de realisatiefase is ontwikkeld voldoet aan de requirements. Dit gebeurt door een testplan te maken op basis van de use cases.

Met behulp van dit testplan kan vervolgens bepaald worden of de gemaakte import/export wizard van voldoende kwaliteit is. Dit kan worden getest aan de hand van een automatische testing tool of desnoods handmatig. Op basis van de bevindingen die worden gemaakt kan een testrapport worden ingevuld. Het testrapport zal uiteindelijk aan kunnen geven welke functionaliteit succesvol is opgenomen in de import/export wizard.

Op basis hiervan zal een dergelijk testrapport kunnen aangeven of de gemaakte architectuur correct is voor het implementeren van een import/export wizard in iDRM en of deze conflicten weghaalt of oplost.

7.1.1. Iteratie 1

Tijdens de eerste iteratie is gewerkt aan het maken van het testplan. Er is al vrij snel bevonden dat uitputtend testen van de applicatie extreem lang zal duren gezien de hoeveelheid mogelijkheden die er zijn. Vanwege deze reden is gekozen voor het testen aan de hand van de door de gebruiker gespecificeerde cases.

Hierbij is echter wel input data gebruikt die een relatief groot vlak dekt. Dit is gedaan door middel van pair-wise testing. Aan het eind van de eerste iteratie is op basis van deze cases een testrapport opgesteld. De bevindingen zijn terug te vinden in het testrapport.

Gedurende de eerste iteratie is gebleken dat nog niet alle functionaliteit correct was uitgewerkt. De testresultaten van de eerste iteratie zijn als volgt:

Test case	Status
Import bestaand project in leeg project.	
Hergebruik elementen uit ander project in een leeg project.	
Verbeter een bestaand project op basis van een import project.	
Import een project en neem conflicten niet mee.	
Import een project en hernoem alles dat een conflict geeft.	
Importeer patterns en neem layers niet mee.	

Hoewel dit inderdaad betekent dat er nauwelijks iets goed is gegaan tijdens de testen is dit voornamelijk doordat de testen voor tests hierin al zijn opgenomen. Gezien deze functionaliteit nog niet is gemaakt tijdens de eerste iteratie gaan de testen hiervoor ook fout.

Deze testen waren gemakkelijk te doen doordat er aan het eind van de iteratie automatische testen zijn opgesteld. Dit is gebeurt met behulp van Testify een tool die intern is ontwikkeld. Met deze tool kunnen user interface testen worden gemaakt. De test scripts hiervoor zijn terug te vinden in de bijlage.

De specifieke criteria die zijn gehandhaafd zijn terug te vinden in het testrapport.

7.1.2. Iteratie 2

Aan het eind van de tweede iteratie is behoorlijke voortgang gemaakt met de functionaliteit en is nogmaals de architectuur en functionaliteit getest. Dit heeft uiteindelijk de volgende resultaten opgeleverd:

Test case	Status
Import bestaand project in leeg project.	
Hergebruik elementen uit ander project in een leeg project.	
Verbeter een bestaand project op basis van een import project.	
Import een project en neem conflicten niet mee.	
Import een project en hernoem alles dat een conflict geeft.	
Importeer patterns en neem layers niet mee.	

Deze testen zijn uitgevoerd met behulp van de test scripts die zijn opgesteld aan het einde van de eerste iteratie.

Tijdens de testen zijn nog wel gebreken in de interface gevonden. Deze fouten zaten niet direct in de import widget maar in het scherm van iDRM dat de data moet tonen.

7.1.3. Conclusie

Op basis van de bevonden resultaten kan gesteld worden dat de functionaliteit aan het einde van de tweede iteratie genoeg mogelijkheden heeft om de testen succesvol uit te voeren.

Hoewel deze testen niet alle onderdelen testen kan geconcludeerd worden dat de voor de gebruiker belangrijke functionele eisen werken. Dit komt doordat de testen zijn opgebouwd uit de functionaliteit die de gebruiker graag zou willen.

Het gebruik van automatische testen heeft er mogelijk voor gezorgd dat de resultaten van de tweede iteratie sneller zijn bevonden doordat de al ontwikkelde tests konden worden gebruikt. Ook is er op een vergelijkbare wijze getest doordat hetzelfde testscript is aangehouden.

Er kan dus op basis van de testresultaten gesteld worden dat deze architectuur functioneert voor de door de gebruiker opgestelde use cases.

7.2. Userinterface testen

Bewijzen of iets gebruikersvriendelijk is blijkt nog niet heel makkelijk te zijn. De definitie van gebruikersvriendelijk is als volgt:

"handig in het gebruik, doordat bij het ontwerp rekening met de gebruiker gehouden is"
(Gebruikersvriendelijk, 2015)

Gezien er tijdens de onderzoeksfase al gestart is met het ontwerp van het programma en de gebruiker die het moet gaan gebruiken is het laatste al beantwoord. Of dit er echter voor zorgt dat het ook handig is in het gebruik is een veel interessantere vraag.

Een hoog niveau van gebruikersvriendelijkheid kan voordelen opleveren als verbeterde efficiëntie en productiviteit (Karat, 1994). Deze en andere aspecten leiden ertoe dat er maatstaven zijn waarmee kan worden gemeten.

Zo kan worden bepaald of bepaalde verbeteringen een toename veroorzaken in bijvoorbeeld productiviteit. Er is hierbij gekozen voor het testen van de interface door de gebruikers. De verbeteringen die hierbij worden opgemerkt kunnen bijdragen aan een goede gebruikersvriendelijkheid.

De bevindingen die dit oplevert zijn terug gevoerd naar het technische. Dit kan de tweede deelvraag valideren door bevindingen bij het testen die uitwijzen dat de import/export wizard gemakkelijk te gebruiken is.

In dit onderdeel wordt per iteratie beschreven welke testresultaten er zijn en waar dit toe heeft geleid.

7.2.1. Iteratie 1

Gedurende de eerste iteratie is gestart met het opstellen van een test voor het testen van de gebruikersinterface. Aangezien er niet genoeg tijd beschikbaar is geweest voor het afnemen van deze testen is er besloten om direct naar feedback te vragen van de klanten en aan het eind van de tweede iteratie de daadwerkelijke user testen te doen.

De bruikbare feedback is geanalyseerd en opgenomen in de volgende lijst:

Feedback
Het detail scherm moet alle velden laten zien anders is het onduidelijk
Importeren van patterns is traag.
Lijst met remap opties is erg lang en niet overzichtelijk.
Do not import is onduidelijk en zou beter Map to existing kunnen zijn.
Het is niet snel duidelijk in het detail scherm wat het verschil is bij conflicten.
Tooltip toevoegen voor detail scherm.

Deze feedback bevat een aantal sterke punten en maakt duidelijk dat er onderdelen zijn van de interface die ondanks het vooronderzoek niet handig werken.

Deze punten zijn opgenomen in de functionaliteit die wordt behandeld in de tweede iteratie. Dit zorgt ervoor dat er direct een verbetering kan worden gemaakt. De feedback die is opgenomen in de tabel is verbeterd in het design en verwerkt in het technisch ontwerp.

7.2.2. Iteratie 2

Tijdens de tweede iteratie is hard gewerkt aan het afnemen van de gebruikers testen en het bepalen van verbeteringspunten. De testen hebben uiteindelijk een hoop informatie opgeleverd.

Hoewel er waarschijnlijk nieuwe punten bij zullen komen na elke nieuwe iteratie is het belangrijker om te bepalen of er verbetering is voor de gebruikers.

In de voorgaande iteratie zijn verschillende punten vermeld en deze zijn verbeterd in de huidige versie. De feedback die is opgeleverd aan de hand van de testen in deze iteratie en overlegd is met de gebruikers is als volgt:

Feedback
Map to existing is onduidelijk en zou beter Do not import kunnen zijn.
Verander de naam van tree elementen naar de workspace (Test patterns i.p.v. Tests).
'-' als header voor een conflict kan beter leeg blijven.
Als een projectonderdeel niet in het project zit zou die grijs kunnen worden.

Deze feedback is verkregen door met de gebruikers test scenario's te doorlopen. De specifieke antwoorden die zijn gegeven zijn terug te vinden in de bijlage. Door een inschatting te maken van hoe belangrijk een specifiek punt is kan worden voorkomen dat er wijzigingen worden gemaakt die specifiek tot een individu behoren.

Dit bleek een probleem te zijn na het afnemen van deze feedback omdat er duidelijk naar voren is gekomen dat de vermelding Map to existing niet duidelijk is. Dit was echter feedback op de eerste iteratie. Om dit bij de nieuwe resultaten te voorkomen is per onderdeel dat niet door meerdere mensen is aangegeven overlegd met alle gebruikers.

Gezien er geen tijd meer is geweest om een derde iteratie te doen is besloten deze feedback te bewaren zodat het bedrijf hier nog aan verder zou kunnen werken.

Om te bepalen of de verbeteringen uit de eerste iteratie succesvol zijn is getest met andere testpersonen om ervoor te zorgen dat er geen gewenning ontstaat en daardoor de productiviteit toeneemt. Gezien het beperkte aantal werknemers zorgde dit er voor dat slechts vier mensen het konden testen.

Tijdens de testen viel op dat het vinden van het detailscherm met behulp van een tooltip toch een eenvoudigere taak was dan voorheen. De remap lijsten bleken overzichtelijker dan voorheen doordat vrijwel alles werd laten zien.

Erg opvallend was dat er veel verwarring ontstond door de optie Map to existing en dat het merendeel van de testers vroeg wat die optie precies zou doen.

7.2.3. Conclusie

Op basis van de bevindingen die zijn gemaakt kan worden geconcludeerd dat de architectuur van de import widget een correcte implementatie biedt voor de use cases die zijn gegeven door de gebruikers. Hoewel dit geen garantie biedt voor de complete functionele werking kan wel gesteld worden dat een breed vlak is getest.

Dit betreft dan voornamelijk de opties die kunnen worden gebruikt binnen de import widget. Voor elk van de elementen is elke mogelijke import optie uitgevoerd en is het verwachte resultaat opgeleverd.

Voor de gebruikersinterface kan worden gezegd dat de veranderingen die zijn gemaakt ervoor hebben gezorgd dat er meer bruikbare feedback is opgeleverd.

De punten die zijn verbeterd aan de gebruikersinterface in iteratie twee hebben er in sommige gevallen voor gezorgd dat de interface eenvoudiger is in het gebruik.

Gezien de gebruikerstesten pas kunnen worden gedaan aan het eind van een iteratie zijn er nog een aantal feedback onderdelen die nog niet in de daadwerkelijke applicatie zijn verwerkt.

8. Conclusie

Er kan met behulp van de bevindingen en voorgaande conclusies nu een antwoord worden gegeven op de hoofdvraag. Zoals aangegeven in het plan van aanpak zal de hoofdvraag worden beantwoord door de deelvragen.

De relevante projectelementen die terug zijn te vinden in een iDRM import/export zijn bepaald aan de hand van een analyse. Uit de opsomming die dit uiteindelijk heeft opgeleverd valt op te maken dat de huidige architectuur deze allemaal bevat. Deze projectelementen zijn weergegeven in hoofdstuk 5.2.

Er is tijdens de analyse van de code duidelijk geworden dat er een hoop conflicten zijn die zich voor kunnen doen. Sommige conflicten komen in de bestaande code niet voor omdat deze mogelijkheden worden ontweken. Alle mogelijke conflicten die zich kunnen voordoen met de projectelementen van iDRM zijn uiteindelijk opgenomen in een overzicht (hoofdstuk 5.5) en geven duidelijk aan dat sommige conflicten in de huidige import erg grote problemen opleveren (soms zelfs crashes).

Vanuit de conclusies in de onderzoeksfase kan bepaald worden dat de architectuur die wordt aangedragen (hoofdstuk 5.6) vrij eenvoudig is maar toch door de functionele testen weet te komen. Hoewel in de realisatiefase wordt aangegeven dat het uitwerken hiervan nog niet eenvoudig bleek te zijn is het testen succesvol verlopen.

Als het gaat om de gebruikersvriendelijke manier van weergeven kan worden opgemaakt uit de gebruikerstesten (hoofdstuk 7.2) dat er daadwerkelijke verbeteringen zijn in de interface en dat de manier waarop het kan worden gebruikt eenvoudiger is ten opzichte van voorgaande versies.

Er kan op basis van deze conclusies dus worden gesteld dat de architectuur zoals beschreven in dit document kan worden gebruikt om relevante projectelementen te importeren in iDRM. Gezien de testen niet hebben uitgewezen dat er in situaties crashes voorkwamen of dat een manier van importeren niet mogelijk was kan worden gesteld dat conflicten nu daadwerkelijk kunnen worden opgelost.

In plaats van conflicten automatisch te laten afhandelen is gekozen voor het handmatig kunnen kiezen van hoe een conflict wordt afgehandeld. Dit zorgt er niet direct voor dat er een vermindering is van conflicten maar het zorgt er wel voor dat er meer vrijheid is in de manier waarop conflicten worden afgehandeld.

9. Aanbevelingen

Op het gebied van aanbevelingen en suggesties voor vervolgonderzoek komt voornamelijk de functionaliteit naar boven die nog niet is opgenomen in de import widget. Hoewel deze nu alle onderdelen van een project bevat is er nog niet goed genoeg gekeken naar de could-have functionaliteiten en in hoeverre deze een bijdrage leveren aan de gebruikersvriendelijkheid.

Een aanbeveling die op het gebied van functionaliteit echter wel naar voren komt is het integreren van export/link functionaliteit in de huidige import widget. Dit kan op verschillende manieren gebeuren maar zal vrij eenvoudig kunnen worden opgenomen in de import widget. Een alternatief is dat er een losse klasse voor wordt gemaakt die leunt op een base klasse tussen de twee.

Dit kan worden uitgevoerd door een Software Engineer binnen NP-Komplete Technologies. Gezien dit gebruik maakt van een hoop onderdelen die al zijn gemaakt is het mogelijk dit in een kortere tijd te doen dan dit project.

Los van ontbrekende functionaliteit zou het uitvoeren van meer testen onder de gebruikers ervoor kunnen zorgen dat de interface nog bruikbaarder wordt. Het iteratief verbeteren hiervan kan leiden tot een betere procesgang. Ook zou op het gebied van testen kunnen worden gekeken naar functionele testen die voortkomen uit de conflict mogelijkheden. Hier was gedurende het project te weinig tijd voor.

Een potentieel vervolgonderzoek zou zich kunnen richten op het exporteren van een project. Dit gebeurt momenteel door het gehele project te exporteren. De import widget zou echter een base klasse kunnen hebben die zou kunnen werken voor zowel import als export. Hierbij zou dan de mogelijkheid kunnen bestaan om alleen bepaalde onderdelen te exporteren. Tijdens het project is besloten dit niet te doen aangezien dit niet direct het probleem opleverde.

Om ervoor te zorgen dat het onderzoek nogmaals kan worden uitgevoerd is de originele code en het originele project bewaard gebleven (backup is beschikbaar via SVN). Dit is nodig aangezien er anders geen mogelijkheid is de analyses nogmaals uit te voeren.

10. Reflectie

Tijdens het project is duidelijk geworden dat het plan van aanpak niet op elk punt kon worden aangehouden. Dit kwam voornamelijk door de resultaten die de analyses opleverden. Zo werd al vrij snel duidelijk dat het probleem het beste zou kunnen worden opgelost door een import widget te maken. Dit heeft ervoor gezorgd dat de vragen van het onderzoek zijn veranderd.

Ook zijn er ten opzichte van het plan van aanpak wat meer opdelingen gemaakt in de taken die moeten worden uitgevoerd. Hoewel deze nog steeds grotendeels overeenkomen met de onderdelen die vermeld zijn in het plan van aanpak is er besloten om deze gedurende het project op te delen in meerdere kleine taken. De scriptie geeft ze echter wel weer als de taak die in het plan van aanpak is beschreven.

In dit onderdeel wordt per fase van het project gekeken naar welk proces is doorlopen en waarom hiervoor gekozen is.

10.1. Onderzoek

Tijdens de onderzoeksfase is er een hoop druk geweest. De planning voor deze fase was zes weken waarvan vijf voor het functioneel en technisch ontwerp. Dit hield echter ook in dat de analyses en het plan van aanpak moest worden gemaakt in deze periode.

Of dit een impact heeft gehad op de kwaliteit van de resultaten is moeilijk te bepalen gezien er niet kan worden uitgewezen dat bepaalde functionaliteit niet werkt.

Tijdens deze fase is er gewerkt met de prioriteit zoals die beschreven is in het plan van aanpak. Dit werkte goed en de kennis die nodig was voor opvolgende onderdelen was inderdaad aanwezig.

Om te kunnen garanderen dat deze producten van goede kwaliteit zijn is nauw samengewerkt met andere Software Engineers bij NP-Komplete Technologies.

Gezien er vanuit een vrij abstract punt is gestart is telkens meer invulling gegeven aan het beeld dat er is van hoe het systeem werkt. Door dit beeld terug te koppelen naar eerdere onderdelen in deze fase kan worden bevestigd of onderdelen correct zijn.

Een alternatief voor deze aanpak is uiteraard het uitwerken van de vermoedens maar dit zou het waarschijnlijker maken dat er fouten in de analyses zitten. Gezien er gedurende deze periode een tekort aan tijd was leek het niet onmogelijk dat dit ook zou kunnen gebeuren voor andere fases van het project. Het vooruit schuiven van verbeteringen leek dan ook geen goede optie. Vanuit die redenering is gekozen voor de originele procesgang.

Deze procesgang bleek uiteindelijk goed te werken toen duidelijk werd dat de functionele testen zijn geslaagd en dat de hoofdvraag is beantwoord. Vanuit dit standpunt kan worden gesteld dat de procesgang tijdens de onderzoeksfase bij heeft gedragen aan het beantwoorden van de hoofdvraag en het succesvol verlopen van het project.

Of een andere procesgang in dit geval een verandering had gebracht aan de uiteindelijke resultaten kan moeilijk bepaald worden. Deze resultaten komen op een vergelijkbare manier voort uit de bevonden resultaten maar zijn niet direct gevalideerd. Indien de architectuur niet bleek te kloppen zou duidelijk zijn dat de alternatieve procesgang een betere zou zijn geweest.

10.2. Realisatie

Tijdens de realisatie was minder te merken van de tijdsdruk. Dit kwam waarschijnlijk doordat er een duidelijke afbakening is gemaakt van wat minimaal moest worden gemaakt. Het proces dat tijdens de

realisatie aangehouden is vind zich terug in deze vereiste functionaliteit. Zo is gedurende de eerste iteratie gewerkt richting de must-have functionaliteiten zoals die beschreven zijn in de onderzoeksfase.

Er is tijdens de eerste iteratie besloten deze functionaliteit te implementeren op een manier waarin en het aan alle eisen van die iteratie voldoet. Er is dus nog geen rekening gehouden met de should-have en could-have functionaliteit.

Dit heeft er voor gezorgd dat het eenvoudiger was om de import widget te maken. In verhouding tot de planning is duidelijk geworden dat er hierdoor meer tijd over is gebleven. Dit heeft er echter voor gezorgd dat de volgende iteratie langer duurde doordat bepaalde onderdelen moesten worden herschreven/veranderd.

Gedurende de tweede iteratie is bepaald dat dit zich zou kunnen herhalen voor de functionaliteit die in de derde iteratie moet worden uitgewerkt. Daarom is er op dit punt besloten om rekening te houden met zowel de should-have functionaliteit als de could-have functionaliteit.

De aanleiding van deze vreemde aanpak komt terug vanuit de onderzoeksfase. De werkdruk in deze fase heeft ervoor gezorgd dat alle speling in de planning is verdwenen. Door in de eerste iteratie deze werkwijze te hanteren is er meer tijd vrij gekomen. Een groot deel van deze tijd is echter weer opgenomen door het moeten veranderen voor de tweede iteratie.

Of dit er voor heeft gezorgd dat het uiteindelijk niet mogelijk is geweest om nog een derde iteratie te doen is niet zeker. Er kan worden gesteld dat door al vanaf het begin dit proces op de latere wijze te doorlopen er minder had hoeven worden veranderd. Dit neemt echter niet weg dat dit meer tijd had gekost en in de krappe tijd die stond voor twee iteraties geen extra iteratie had kunnen plaatsen.

10.3. Validatie

Tijdens de validatie is vooral gewerkt naar het beantwoorden van de deelvragen. Het daadwerkelijke bevestigen stond hier bovenaan. Om dit voor elkaar te krijgen is besloten de architectuur van de import widget als meest relevante onderdeel te zien.

Door dit te doen kan de hoofdvraag worden beantwoord gezien de gebruikersvriendelijkheid niet direct in de hoofdvraag is opgenomen.

Door in de eerste iteratie te kijken naar hoe kan worden getest en hier een duidelijke omgeving voor op te zetten is duidelijk geworden dat testen in nieuwe iteraties snel kunnen worden uitgevoerd.

Zo is besloten gebruik te maken van automatische testing tools om ervoor te zorgen dat er na elke iteratie snel kan worden bepaald of de functionaliteit voldoet of nog onderdelen ontbreekt. Deze beslissing komt voornamelijk vanuit het bedrijf gezien er binnen het bedrijf standaard zo gewerkt wordt.

Voor het testen van de gebruikersinterface is gedurende de eerste iteratie besloten om hiervoor nog geen template op te stellen. Dit kwam voornamelijk doordat gebruikers de import widget al geprobeerd hadden en feedback verstuurde via email. Er is op dit punt besloten om met deze feedback te werken. Deze aanpak is minder goed dan de aanpak die voor de opvolgende iteratie is gekozen.

In de tweede iteratie zijn slechts vier mensen gevraagd om mee te doen aan de gebruikerstesten dit had wellicht beter gekund door meer mensen te vragen. In dit geval was het mogelijk geweest om beter te kijken naar overeenkomsten (met vier mensen is het lastig in te schatten wat extreme waardes zijn).

10.4. Leerpunten

Vanuit een persoonlijk perspectief heb ik me gedurende het project erg proberen te verdiepen in de bedrijfslogica. Deze geeft mij extra verdieping in hoe chips werken en waarom juist deze validatie nodig is bij het ontwikkelen ervan.

Het heeft me geleerd hoe een chip er van binnen uit zal kunnen zien en waarom dit zo gebeurt. Deze kennis zal wellicht van pas kunnen komen als ik nog eens met VHDL (VHDL, 2015) aan de slag ga.

Ook heb ik het software ontwikkeling traject doorgelopen. Ik ben van mening dat elke keer dat dit gebeurt je een hoop nieuwe dingen leert. Dit gaat dan over verschillende methodieken maar ook programmeer vaardigheden.

Hoewel ik al vrij veel kennis had van C++ en Qt heb ik toch weer een paar nieuwe functies gebruikt en eens iets heel anders geprobeerd te bouwen. Het doorlopen van het hele software ontwikkelingsproces is iets dat ik persoonlijk niet vaak heb gezien bij kleine bedrijven die werken met C++. Ik heb gedurende het project gemerkt dat er een hoop kennis van aanwezig is in het bedrijfsleven maar dat er meestal afwegingen worden gemaakt ten opzichte van de hoeveelheid documentatie.

Ik heb vanuit deze instelling geleerd dat er voor kleine projecten niet altijd noodzaak is voor erg veel documentatie en dat bepaalde onderdelen soms zelfs redundant zijn.

De ervaring die ik heb opgedaan met het uitvoeren en beschrijven van onderzoek zullen in de toekomst erg handig zijn. Deze vaardigheden komen terug in verschillende werelden waaronder masteropleidingen.

Mijn begeleider geeft naar mijn idee een goed beeld van hoe een scriptie eruit moet zien maar ook wat er inhoudelijk goed/minder goed is. Hierdoor heb ik telkens kunnen verbeteren en heb ik naar mijn mening een goede scriptie kunnen schrijven.

11. Bronnen

(2009, January 1). Opgehaald van http://drimble.nl/bedrijf/eindhoven/20452160/np-komplete-technologies-bv.html

Ambler, S. W. (2014). *UML 2 Use Case Diagrams: An Agile Introduction.* Opgehaald van Agile Modeling: http://www.agilemodeling.com/artifacts/useCaseDiagram.htm

Christel, M. (1992, September). *Issues in Requirements Elicitation.* Opgehaald van SEI: http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=12553

Class Diagram. (2015). Opgehaald van Wikipedia: http://en.wikipedia.org/wiki/Class_diagram

Cockburn, A. (2001). Writing effective use cases. *Addison-Wesley*, 68. Opgehaald van http://alistair.cockburn.us/get/2465

Gebruikersvriendelijk. (2015). Opgehaald van Van Dale: http://www.vandale.nl/opzoeken?pattern=gebruikersvriendelijk&lang=nn#.VVnq9FW8PGc

God Object. (2015). Opgehaald van Wikipedia: http://en.wikipedia.org/wiki/God_object

Holub, A. (2011, 09 26). *Holub*. Opgehaald van UML Quick Reference: http://www.holub.com/goodies/uml/

Karat, C.-M. (1994). *A Business Case Approach to Usability Cost Justification.* New York: Academic Press, Inc.

McKibben, K. (2015). *Business Process*. Opgehaald van Laserfiche: https://www.laserfiche.com/solutionexchange/how-to-diagram-your-business-process/

Model View Controller. (2015). Opgehaald van Wikipedia: http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

MoSCoW Method. (2015). Opgehaald van Wikipedia: http://en.wikipedia.org/wiki/MoSCoW_method

Sage. (2015). *iDRM*. Opgehaald van http://www.sage-da.com/idrm.html

Takumi. (2004). Opgehaald van http://www.takumi-tech.com/

Technologies, N.-K. (2015, 5 25). Opgehaald van http://www.np-komplete.com/

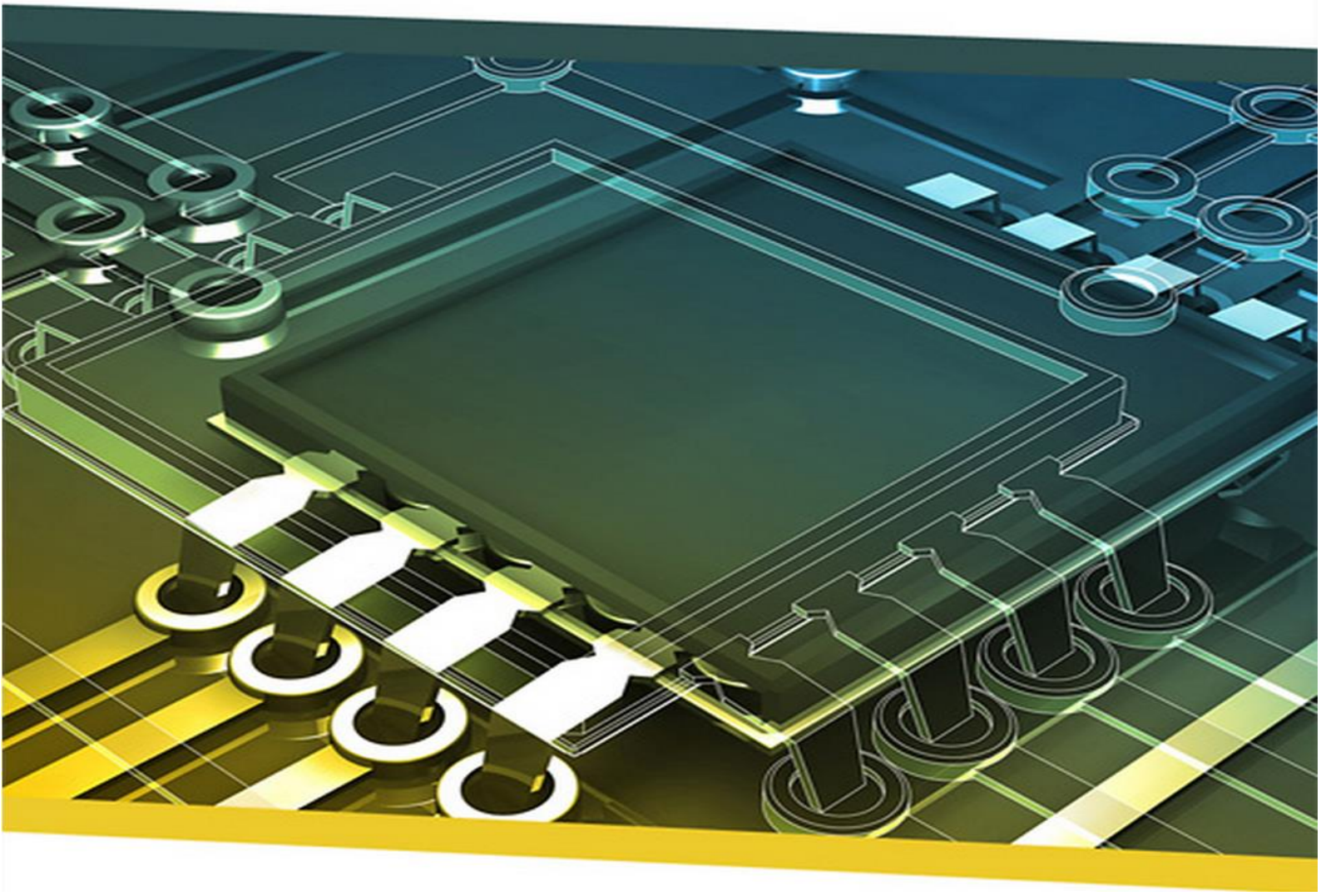
Typedef. (2015). Opgehaald van Wikipedia: http://en.wikipedia.org/wiki/Typedef

VHDL. (2015). Opgehaald van Wikipedia: http://en.wikipedia.org/wiki/VHDL

Wikipedia. (2015, 5 25). Opgehaald van http://en.wikipedia.org/wiki/NP-complete

12. Bijlage

12.1. Bijlage 1 – Plan van aanpak



Project import/export binnen iDRM

Plan van aanpak

Student

Naam:	Velleman
Voorletters:	F.M.V.
Roepnaam:	Floris
Studentnummer:	1602376
Docentbegeleider:	Alex Jongman

Bedrijf

Naam:	NP-Komplete Technologies
Adres:	De Rozentuin 18
Postcode:	5611 SW
Plaats:	Eindhoven
Inhoudelijk bedrijfsbegeleider:	Maarten Berkens
Technisch bedrijfsbegeleider:	Rob Gelderblom

Contract afstudeercontract

Instituut voor ICT
Nijenoord 1, 3552 AS, UTRECHT

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

Datum:	18-03-2015

Naam student:	Floris Velleman
Opleiding:	Informatica
Variant:	voltijd / ~~deeltijd~~ / ~~duaal~~
Adres student:	Fröbellaan 18
Postcode / Woonplaats student:	3706TG Zeist
Studentnummer:	1602376
Telefoonnummer privé:	06-40431356
E-mailadres:	florisvelleman@hotmail.com

Naam bedrijf (afstuderen):	NP-Komplete technologies
Adres bedrijf:	De Rozentuin 18
Postcode / Woonplaats bedrijf:	5611SW Eindhoven
Naam bedrijfsbegeleider:	Rob Gelderblom
Telefoonnummer bedrijfsbegeleider:	+31(0)40-2110009
E-mailadres bedrijfsbegeleider:	r.gelderblom@np-komplete.com

Beoogde datum van afstuderen:	Juni 2015
Geheimhouding geaccordeerd door HU op:	n.v.t.

Ondergetekenden verklaren hiermee akkoord te gaan met de inhoud van bijgesloten PvA.

Handtekeningen

Student :

Docentbegeleider :

Bedrijfsbegeleider[1] :

[1] **Door ondertekening van dit formulier verklaart de bedrijfsbegeleider minimaal een hbo- of vergelijkbare opleiding te hebben.**

Versiebeheer

Onderstaande tabel geeft een weergave van de voorgaande versies en gemaakte revisies van dit document.

Versie	Status	Datum	Omschrijving
2.0	Voltooid	12-03-2015	Afstudeercontract opgenomen.
1.9	Lopend	12-03-2015	Algemene verbeteringen en spelling.
1.8	Lopend	11-03-2015	Verbeteringen op basis van feedback.
1.7	Concept	09-03-2015	Theoretisch kader.
1.6	Lopend	08-03-2015	Verbeteringen aan document structuur.
1.5	Lopend	05-03-2015	Planning opgenomen.
1.4	Lopend	04-03-2015	Verbeteringen op basis van feedback.
1.3	Concept	02-03-2015	Risico en fasering herschreven.
1.2	Lopend	27-02-2015	Aanpak en methodiek herschreven.
1.1	Lopend	25-02-2015	Verbeteringen naar aanleiding van feedback.
1.0	Concept	20-02-2015	Verbeteringen naar aanleiding van feedback van bedrijfsbegeleider.
0.9	Lopend	20-02-2015	Verbeteringen aan kwaliteit.
0.8	Lopend	20-02-2015	Inhoud toegevoegd: ▪ Aanpak en methodiek ▪ Risico
0.7	Lopend	19-02-2015	Verbeteringen naar aanleiding van feedback.
0.6	Lopend	18-02-2015	Verbeteringen naar aanleiding van feedback.
0.5	Concept	16-02-2015	Concept versie
0.4	Lopend	13-02-2015	Inhoud toegevoegd: ▪ Voorwoord ▪ Inleiding ▪ Projectorganisatie ▪ Hoofdvraag en deelvragen ▪ Concept data ▪ Gegevens
0.3	Lopend	12-02-2015	Verbeteringen naar aanleiding klant.
0.2	Lopend	11-02-2015	Inhoud toegevoegd: ▪ Context ▪ Aanleiding ▪ Analyse ▪ Probleem ▪ Opdracht
0.1	Lopend	10-02-2015	Opzet document.

Voorwoord

Het schrijven van een plan van aanpak is nog best lastig. Zelf zou ik natuurlijk het liefst beginnen met programmeren. Ik zie echter wel in dat het maken van een plan van aanpak een groot voordeel oplevert tijdens het uitvoeren van het afstudeerproject.

Ik wil graag Alex Jongman bedanken voor de begeleiding tijdens het maken van het plan van aanpak. De verhelderende uitleg heeft naar mijn mening meer diepgang gegeven aan het document en is een zeer leerzame ervaring geweest.

Ook wil ik Rob Gelderblom en Simon Klaver bedanken voor de hulp vanuit de organisatie. Deze hulp heeft ervoor gezorgd dat ik een beter overzicht kreeg van zowel de organisatie als het probleem en dat heeft ervoor gezorgd dat de kwaliteit van het document is toegenomen.

Eindhoven
09-03-2015
Floris Velleman

Inleiding

Als laatste jaar student zal er in de periode van 2 februari 2015 tot 2 juni 2015 een afstudeeropdracht worden uitgevoerd in opdracht van de Hogeschool Utrecht. Deze afstudeeropdracht wordt gedaan bij NP-Komplete Technologies in Eindhoven.

Er wordt tijdens deze periode gewerkt aan een import/export wizard. Waarom dit gebeurt en wat hoe dit kan gebeuren wordt beschreven in dit document.

Maar wat is import/export nou precies? Het kan op vele manieren worden gemaakt maar de meest voor de hand liggende methode is het overschrijven van bestaande bestanden. Dit is zonder twijfel de makkelijkste methode om het te regelen maar of het zo handig is blijft de vraag.

Is mergen dan altijd de oplossing? Wellicht moet er eerder gedacht worden aan het maken van nieuwe bestanden zodra conflicten worden gedetecteerd. Dit zijn echter punten die een gebruiker waarschijnlijk zelf in handen wil hebben gezien de gegevens van de gebruiker hieraan gekoppeld zijn.

Om een correcte context te verkrijgen wordt ingegaan op hoe het bedrijf eruit ziet en hoe het te werk gaat. Vervolgens wordt een achtergrond van de bedrijf software gegeven. Er wordt kort en bondig uitgelegd welke elementen in de software centraal staan.

Maar ook wordt gekeken naar hoe deze elementen samenhangen en hoe dit in verband staat tot een mogelijk probleem. Op basis van de bedrijfsdefinitie wordt vervolgens een aanleiding opgesteld waarop de student reageert met een analyse.

Met deze analyse kan vervolgens een daadwerkelijk probleem worden bepaald. Dit probleem heeft een paar doeleinden en leidt vanuit dit doel tot vragen.

Om de vragen te beantwoorden wordt elke deelvraag uitgezet tegen een methodiek en milestone. Als vervolgen de risico’s van deze aanpak in kaart zijn gebracht wordt een planning opgesteld die kan worden aangehouden gedurende het afstudeerproject.

NP-Komplete Technologies

In dit hoofdstuk wordt gekeken naar wat het bedrijf doet en hoe de student zich hier in naar voren brengt. Welke positie wordt aangenomen tijdens de afstudeeropdracht en welke andere interne bedrijfsprojecten zijn hieraan verbonden.

De organisatie

NP-Komplete is een software bedrijf binnen de branche EDA (Electronic Design Automation voor chip design) en bevindt zich in Eindhoven. Het bedrijf is opgericht in 2009, maar de meeste personeelsleden werkten al samen in voorloper Takumi Technologies (vanaf 2004).

De naam NP-Komplete Technologies komt van nondeterministic polynomial time complete. Maar gebruikt hiervoor Komplete in plaats van Complete omdat dit bedrijf al bestond.

De huidige software die ze maken staat bekend als iDRM wat staat voor integrated design rule management. Deze wordt gemarket onder de bedrijfsnaam Sage Design Automation (hoofdkantoor in Santa Clara, CA). Deze software wordt gebruikt om chip design te testen. Ook ontwikkelt NP-Komplete Technologies nog een aantal ondersteunende tools die bepaalde opdrachten makkelijker maken.

NP-Komplete Technologies heeft klanten maar die kunnen niet worden vermeld vanwege een non-disclosure agreement.

Het is een vrij klein bedrijf met ongeveer tien tot vijftien werknemers. Omdat het een vrij klein bedrijf is wordt er gezamenlijk geluncht. De werktijden zijn voor iedereen flexibel wat ervoor zorgt dat de meeste medewerkers iets later op de ochtend komen. De bedrijfscultuur is echter wel zo dat de student zijn eigen deadlines zal moeten hanteren.

Vanwege de grote is er binnen de organisatie dan ook weinig opdeling op het gebied van afdelingen. Toch is er binnen de organisatie wel een opdeling van verantwoordelijkheden. Zo houdt iemand zich bezig met de compiler die gebruikt wordt en een ander met bijvoorbeeld de distributie.

De opdeling vindt zich ook terug in de beroepen die mensen hebben. Zo zijn er programmeurs maar ook AE’s (application engineers). Deze application engineers werken nauw samen met klanten en helpen bij installatie. Soms werken ze zelfs bij klanten met de programma’s.

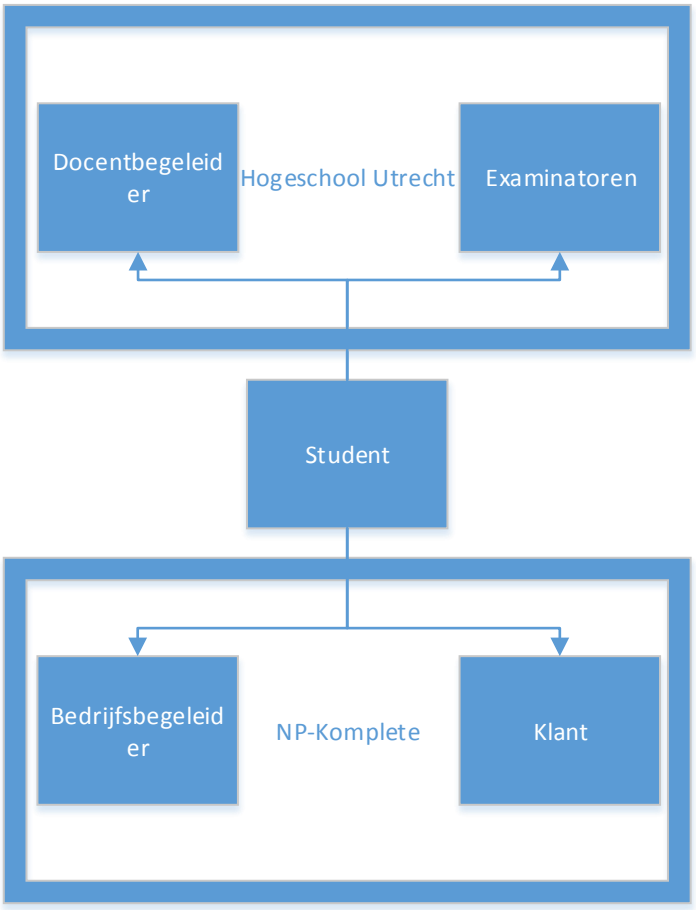
Positie binnen de organisatie

De student neemt de rol aan van software engineer binnen de organisatie. Het afstudeerproject zelf staat los van andere projecten binnen de organisatie maar is wel onderdeel van het uiteindelijke programma.

Binnen het afstudeerproject worden een aantal rollen ingenomen door verschillende betrokkenen. De volgende betrokkenen en rollen zijn bepaald binnen het bedrijf en de school:

Naam	Rol	Verantwoordelijkheid
Floris Velleman	Student	Uitvoeren van de opdracht en het maken van de documenten.
Alex Jongman	Docentbegeleider	Begeleiden bij het maken van het plan van aanpak en de scriptie.
	Eerste examinator	Beoordelen plan van aanpak en scriptie.
Rik Boss	Tweede examinator	Beoordelen plan van aanpak en scriptie.
Rob Gelderblom	Bedrijfsbegeleider	Begeleiden bij de technische aspecten van het project.
Simon Klaver	Klant	Opdracht beschrijving.

Om duidelijk te maken welke verbindingen er zijn is onderstaand diagram gemaakt:



[2] - Samenhang projectorganisatie

De student is hierbij in het midden geplaatst om aan te geven dat deze tot beiden behoord. De gegevens van het bedrijf als volgt:

Straat:	De Rozentuin 18	Tel:	+31(0)40-2110009
Adres:	5611 SW Eindhoven	Email:	info@np-komplete.com
Land:	Nederland		

[2] Afbeelding 1

Aanleiding

NP-Komplete Technologies bouwt en levert iDRM, een physical design rule check tool, aan chip fabrieken. Dit is een programma waarmee bepaalde design rules van chips kunnen worden gecontroleerd op basis van een regel. Deze regel is gebaseerd op een pattern die toepassing heeft op een bepaalde layer uit de bestaande verzameling van lagen van de chip (layermap). De lagen van deze chip geven een basis om een specifiek figuur te vinden (het pattern). Een regel kan op verschillende lagen zoeken. Het programma helpt chip fabrieken met het controleren van deze regels in chips.

De klanten en application engineers die het programma gebruiken hebben een probleem aangegeven.

Werk van klanten in iDRM is georganiseerd in projecten. Onderdelen van het ene project zijn vaak te hergebruiken in een ander project. Bij het importeren van een project in iDRM kunnen eventuele patterns niet zonder layermap worden geïmporteerd. Dit zorgt ervoor dat de layermap wordt gemerged met de bestaande layermap. Dit levert vaak een resultaat wat niet gewenst is. Ook wordt niet aangegeven wat er precies is veranderd.

Zo is het mogelijk dat een layer op naam wordt overschreven waardoor de bestaande mapping van patterns een hele andere betekenis krijgt. Ook kan het dat er meerdere layermaps zijn die eigenlijk hetzelfde doen.

De klanten willen dat er kan worden gekozen voor een import met of zonder layermap. Ook zouden ze graag extra functionaliteit willen in deze import. Zo willen ze bijvoorbeeld kunnen kiezen welke projectelementen er wel en niet wordt geïmporteerd. De klanten willen de mogelijkheid om conflicten te kunnen verminderen en ze makkelijker te kunnen oplossen.

Vanuit het bedrijf komt naar voren dat een mogelijke interface gebruikersvriendelijk moet zijn omdat klanten duidelijk hebben gemaakt dat het bestaande programma op sommige punten moeilijk is in het gebruik.

Dit houdt voornamelijk in dat een nieuw product dit standpunt zou moeten meenemen in het design van de interface en gebruikersvriendelijkheid.

Probleem

Er kan al vrij snel gesteld worden dat het hier gaat om het uitbreiden en verbeteren van een bestaand proces. Er is een tekort aan functionaliteit in de huidige import.

Dit komt naar voren door de merge conflicten die zich voordoen bij het importeren. Aangezien export hier indirect mee te maken heeft is het mogelijk het op beide punten te verbeteren. Het daadwerkelijke probleem zit dan ook bij het import/export proces.

Het probleem is eigenlijk een losse eis van de klanten en application engineers. Dit is dan ook direct het probleem in import/export. Er is te weinig functionaliteit in het huidige import/export proces om te doen wat de application engineers en klanten willen doen. Het niet aangeven van conflicten tijdens het importeren is een gebrekkige functionaliteit in het huidige proces.

Ook wordt vermeld dat er een probleem is met het mergen omdat hier geen keuze kan worden gemaakt voor welke elementen worden geïmporteerd. Dit impliceert vrijwel direct dat er ook vraag is naar de mogelijkheid om te kiezen voor andere methode van importeren/exporteren.

Zo zou het wellicht mogelijk moeten kunnen zijn een object te kopiëren of om een object te overschrijven.

De manier waarop en hoe een mogelijk nieuw product wordt gebruikt vereist ook aandacht. Zo wordt gezegd dat minstens een deel van de gemaakte functionaliteit niet gemakkelijk te gebruiken is door klanten. Dit is een probleem waarmee rekening moet worden gehouden bij het ontwikkelen van een mogelijk nieuw product.

Er is momenteel dan ook weinig flexibiliteit in het huidige proces. Daarnaast er zijn verbeteringen mogelijk die minimaal de frustraties van de twee groepen wegnemen.

Doelstelling

Dit probleem leidt tot het bepalen van doelstellingen die het probleem verhelpen. Om dit probleem te verhelpen zou het probleem met het importeren van patterns moeten worden verholpen. Om dit te doen is het nodig verschillende import opties te hebben.

Echter zal hier eerst een goede structuur voor moeten worden opgesteld. Dit kan gebeuren door een analyse te maken van de huidige projectstructuur en de huidige code base die wordt gebruikt voor import en export. Deze elementen kunnen worden omschreven als relevante projectelementen aangezien deze toepassing zullen hebben bij het import/export proces.

Ook moet het mogelijk zijn om te kiezen tussen welke onderdelen er wel en niet worden geïmporteerd. Dit kan ook direct worden gebruikt als import methode (behoud origineel). De wijzigingen aan elementen zouden expliciet vermeld moeten worden.

Er zal een dialog moeten worden gemaakt waarmee deze opties kunnen worden gekozen. Ook moet deze dialog een eenvoudig bruikbare interface krijgen waarmee zowel de klanten als application engineers kunnen omgaan. Dit is dan ook direct de doelgroep voor de applicatie.

Deze dialog zou tegelijkertijd werk uit handen van de gebruiker moeten halen. Zo zou het kunnen verwijderen van een layermap kunnen leiden tot het automatische remappen van patterns. Het doel is dan ook om te bepalen of er nog meer van deze afhankelijkheden zijn en deze op te nemen in de dialog.

Na dit project moet er een eindproduct zijn waarmee projectelementen kunnen worden geïmporteerd. Dit eindproduct moet gebruikersvriendelijk zijn en met behulp van de gebruiker in staat zijn conflicten op een flexibele wijze op te lossen.

Vragen

In dit hoofdstuk wordt de hoofdvraag gedefinieerd. Op basis van deze hoofdvraag wordt vervolgens bepaald welke deelvragen er nodig zijn om dit te beantwoorden. Omdat de hoofdvraag een proof of concept oplevert is er sprake van een productopdracht.

Hoofdvraag

Met behulp van welke architectuur kan een import/export wizard die relevante projectelementen bevat worden gemaakt in iDRM zodat conflicten kunnen worden opgelost of verminderd?

Deelvragen

1. Welke architectuur kan worden gebruikt voor het maken van een import/export wizard binnen iDRM?
    a. Hoe zit de huidige import/export architectuur in elkaar binnen iDRM?
2. Hoe kan een import/export wizard op een voor de doelgroep gebruikersvriendelijke manier worden getoond binnen iDRM?
3. Welk relevante projectelementen worden binnen de IDRM import/export bewaard?
4. Welke conflicten kunnen worden onderscheiden bij het importeren van een project binnen iDRM?

Aanpak en methodiek

De ontwikkelstraat bij NP-Komplete technologies is ad hoc. Dit houdt in dat er vrijwel geen documentatie wordt gemaakt die de software beschrijft. Er is wel een user guide en een zeer beknopte beschrijving maar deze geven maar beperkt aan wat het gedrag van de software is.

Gedurende het project zal echter gebruik worden gemaakt van agile[3]. Hier worden een aantal fasen van gebruikt:

- Onderzoek
- Realisatie
- Validatie
- Reflectie

Er is gekozen voor deze aanpak om ervoor te zorgen dat het voor alle betrokkenen duidelijk is in welke fase momenteel gewerkt wordt. Ook zorgt deze aanpak ervoor dat de kwaliteitscriteria in de onderzoeksfase kunnen worden vastgesteld en worden bevestigd in de verificatie fase.

Hierbij wordt iteratief gewerkt gedurende de realisatie en validatie. Hierbij worden de requirements die in de onderzoeksfase zijn ontdekt gebruikt om prioriteit aan te geven. De onderzoeksfase zal eerst helemaal moeten worden doorlopen voordat er in andere fases kan worden gewerkt.

Onderzoeksfase

Het afstudeerproject zal beginnen door een analyse te maken van wat een import/export wizard binnen iDRM allemaal zal moeten bevatten. Dit gebeurt door de deelvragen te beantwoorden en een architectuur te maken voor de import/export wizard.

Functioneel ontwerp

Er wordt gestart met het maken van een functioneel ontwerp om als basis te dienen voor de technische architectuur. Met behulp van de klanten en het bedrijf kan een duidelijke definitie van de vereiste functionaliteiten naar voren komt.

Taak 1	Analyse huidige situatie

Om te kunnen bepalen wat gemaakt moet worden is het belangrijk om te weten hoe de huidige import/export situatie werkt. Om dit te doen zal als eerst een analyse worden gemaakt. Door het maken van een analyse van de huidige situatie kan duidelijk worden aangegeven waar verbeteringen plaats moeten vinden in het proces. Hoewel dit niet direct een deelvraag beantwoord is het een benodigde context voor het bepalen van een correcte architectuur. Dit geeft ook een beter beeld van de werking aan de betrokkenen wat zal kunnen bijdragen aan de manier waarop problemen worden benaderd.

Dit kan gebeuren door te achterhalen op welke manier de klanten het programma momenteel gebruiken om deze taken uit te voeren. Ook kan door de user guide te lezen worden opgemaakt hoe dit proces verloopt. Het bedrijf zelf zal ook informatie hebben over wat de normale flow is van dit proces. Gezien er weinig bestaande documentatie is over dit proces is er gekozen voor deze benadering.

[3] Agile methode

Door het combineren van deze informatie zal een duidelijk overzicht kunnen worden gemaakt van hoe het nu werkt. Dit levert uiteindelijk een diagram op dat op een relatief abstracte wijze weergeeft hoe het proces verloopt.

Taak 2	Analyse huidige project structuur

Vervolgens kan worden bepaald hoe de huidige project structuur in elkaar zit. Dit geeft mogelijk een abstract basis van de huidige import/export architectuur. Deze basis zal op een later punt kunnen worden uitgewerkt naar een technischere benadering die een deelvraag beantwoord.

Door het maken van deze analyse zal er de mogelijkheid kunnen zijn om de functionele eisen aan het systeem verder te benadrukken. Dit kan door de begrippen en samenhang te gebruiken die zijn gevonden door de analyse. De tabel met import/export methode voor specifieke groepen zal er voor zorgen dat er rekening kan worden gehouden met bestaande opties.

Hoewel de technische samenhang op dit punt mogelijk nog niet helemaal duidelijk zal zijn kan dit verder worden uitgewerkt door in het technisch ontwerp deze elementen te vergelijken met de daadwerkelijke code.

Het uitvoeren van deze analyse zal gebeuren door te beschrijven hoe een projectstructuur in elkaar steekt. Dit gebeurt op basis van bestanden die het project heeft en de samenhang die hierin aanwezig is. Deze samenhang kan worden afgeleid uit het programma (iDRM) dat wordt gebruikt door het bedrijf. Ook zal hier duidelijk worden welke import en export functionaliteit er momenteel in het programma zit en voor welke projectelementen dit van toepassing is. Omdat er geen overzicht is van hoe een projectstructuur in elkaar zit is gekozen voor deze aanpak.

Uiteindelijk zal door middel van deze analyse een diagram worden gemaakt dat weergeeft hoe elementen aan elkaar verbonden zijn. Ook zal dit een tabel opleveren waarin per project element groep is weergegeven welke import/export methode er voor zijn. Deze taak zal de derde deelvraag beantwoorden.

Taak 3	Functioneel ontwerp

Zodra taak 1 en taak 2 zijn verricht kan worden gestart met het beschrijven van de functionele eisen aan de nieuwe architectuur. Omdat de voorgaande taken wel bijdragen aan het duidelijk in kaart krijgen van de begrippen en context worden deze opgenomen als eerste onderdeel van het functioneel ontwerp. Vervolgens is het van belang te bepalen welke eisen er zijn (requirements). Op basis van deze eisen kan worden bepaald welke actoren er zijn binnen dit proces wat zal leiden tot het ontwikkelen van use cases op basis van de eisen die er zijn.

Het in kaart krijgen van de functionele eisen kan gebeuren door te overleggen met de opdrachtgever en klant. De opdrachtgever zal uiteindelijk bepalen welke onderdelen opgenomen moeten worden en welke lagere prioriteit hebben. Dit gaat ook op voor het use case diagram indien er meerdere actors blijken te zijn. Vanuit deze requirements kan vervolgens worden bepaald welke use cases er zijn.

Zodra er duidelijkheid is over wat er gemaakt moet worden kunnen er use cases worden gemaakt. Deze bevatten scenario’s, activity diagrammen en een schets van hoe de user interface er minimaal uit moet zien om dit te kunnen uitvoeren. Door middel van deze aanpak kan er relatief eenvoudig worden getest en daarom is er ook besloten het op deze manier te behandelen.

Er zal uiteindelijk een document worden opgeleverd waarin de volgende punten zijn opgenomen:

- Analyse van de huidige situatie
- Analyse van de huidige project structuur
- Requirements
- Functionele eisen
- Actors
- Use case diagram
- Use cases

Dit functionele ontwerp zal dan ook het antwoord bevatten op de derde deelvraag van het project. Daarnaast zal het een basis bieden voor het technisch ontwerp. Ook zal een basis worden geboden voor het beantwoorden van de sub vraag van de eerste deelvraag.

Technisch ontwerp

Zodra een functioneel ontwerp is gemaakt en er een overeenkomst is gemaakt ten opzichte van de requirements en project afbakening kan er een technische ontwerp worden opgesteld. Dit gebeurt op basis van de informatie die is verkregen bij het maken van het functioneel ontwerp.

Taak 4	Analyse van de huidige project import/export

Om te begrijpen waar verbeteringen moeten worden aangebracht is het essentieel om te begrijpen waar moet worden ingehaakt in de bestaande code. Om dit te doen zal een analyse moeten worden gemaakt van hoe de huidige project import/export werkt. Dit zal er uiteindelijk voor kunnen zorgen dat er een klassendiagram kan worden opgesteld en dat hierin verbeteringen kunnen worden gemaakt.

Deze analyse kan worden uitgevoerd door in de huidige code te zoeken vanuit alle gevonden import/export opties. Deze opties zijn bekend vanuit de tabel van taak 2. Door alle in dit proces gevonden geassocieerde elementen in een klassendiagram te zetten kan worden weergegeven van waaruit gewerkt wordt. Vervolgens kan de gemaakte analyse uit het functioneel ontwerp worden gebruikt om te kijken hoe de code zich verhoudt tot de project structuur. Er is voor deze aanpak gekozen omdat er geen documentatie is over hoe de code werkt maar wel dat er import/export functionaliteit is. Deze documentatie kan gebruikt worden om deze punten te vinden.

Dit zal uiteindelijk een klassendiagram opleveren van waaruit een duidelijk beeld wordt gegeven van welke elementen er worden gebruikt in het programma. Ook zal het diagram dat de samenhang van de elementen beschrijft worden vertaald naar de technische implementatie. Dit zorgt uiteindelijk voor een nieuw diagram dat aangeeft hoe de huidige import/export architectuur in elkaar zit binnen iDRM. Dit beantwoordt dan de sub vraag van de eerste deelvraag.

Taak 5	Analyse van de mogelijke conflicten

Om te bepalen welke conflicten er mogelijk zijn bij het import/export proces moet een analyse worden gemaakt die aangeeft op welke onderdelen conflicten kunnen ontstaan. Deze conflicten kunnen dan gebruikt worden voor het verbeteren van het huidige import/export proces.

Een analyse van de mogelijke conflicten zal gebeuren door te beschrijven welke vergelijkingen er plaats vinden in de code. Er zal bepaald moeten worden welke projectelementen een conflict kunnen vormen en op welke manier dit gebeurt. Dit kan gebeuren door in de functies te kijken van de projectelementen (waar vergelijkingen voorkomen).

Dit zal uiteindelijk per project element een tabel opleveren met daarin de velden en of situaties dat een conflict kan optreden en hoe deze momenteel wordt opgelost. Dit overzicht geeft dan een antwoord op de vierde deelvraag.

Taak 6	Technisch ontwerp

Zodra taak 4 en taak 5 uitgevoerd zijn kan er worden gewerkt aan het technisch ontwerp. Op basis van de voorgaande taken is er op dit punt genoeg duidelijkheid om een architectuur te maken voor een import/export wizard binnen iDRM. Ook kan er een design worden opgesteld waarin alle functionele eisen terugkomen.

Door in eerste instantie de resultaten van taak 4 en taak 5 in het technisch ontwerp te stoppen is een duidelijke context aanwezig. Vervolgens kan op basis van de gestelde requirements een klassendiagram worden gemaakt. Zodra dit is gebeurd, kan er een design worden gemaakt dat aansluit op de actors die zijn bepaald in het functionele ontwerp.

Dit kan in eerste instantie gebeuren aan de hand van onderzoek op het gebied van usability. Echter staat onderzoek hierbij niet centraal maar gaat het om het gemak van de gebruiker. Het is dan ook daarom dat dit onderdeel in een latere fase kan worden verbeterd op basis van de input die de gebruikers leveren.

Dit vormt gezamenlijk het technisch ontwerp en zal antwoord geven de eerste en tweede deelvraag. De kwaliteit van deze benadering wordt in de validatiefase bepaald.

Zodra het bedrijf en de klant akkoord gaan met de afgesproken scope en requirements kan de realisatiefase beginnen.

Realisatiefase

Gedurende de realisatiefase zal worden gewerkt aan de import/export wizard zoals die is beschreven in het functioneel- en technisch ontwerp.

Taak 7	Import/export wizard

De import/export wizard wordt gemaakt zodat deze kan worden bevestigd en omdat dit een vraag is vanuit de opdrachtgever. Dit gebeurt door in eerste instantie het klassendiagram te implementeren.

Op basis van deze structuur kan vervolgens gewerkt worden naar de requirements die zijn gesteld (dit gebeurt op basis van de use cases). Er wordt in deze fase dus een meetbaar product gemaakt waarmee de eerste deelvraag mogelijk kan worden beantwoord.

Uiteindelijk zal er een import/export wizard worden opgeleverd die is opgenomen in iDRM. Zodra de import/export widget is gemaakt kan de validatiefase starten.

Validatiefase

Taak 8	Testplan en testrapport

Tijdens de validatiefase zal in eerste instantie worden getest of het product dat in de realisatiefase is ontwikkeld voldoet aan de requirements. Dit gebeurt door een testplan te maken op basis van de use cases.

Met behulp van dit testplan kan vervolgens bepaald worden of de gemaakte import/export wizard van voldoende kwaliteit is. Dit kan worden getest aan de hand van een automatische testing tool of desnoods handmatig. Op basis van de bevindingen die worden gemaakt kan een testrapport worden ingevuld. Het testrapport zal uiteindelijk aan kunnen geven welke functionaliteit succesvol is opgenomen in de import/export wizard.

Op basis hiervan zal een dergelijk testrapport kunnen aangeven of de gemaakte architectuur correct is voor het implementeren van een import/export wizard in iDRM.

Taak 9	Testscript

Zodra dit is bevestigd kan een testscript worden opgesteld. Dit gebeurt op basis van een template en wordt met behulp van de gebruikers uitgevoerd. Dit testscript wordt aangepast om een betere context te geven voor deze specifieke opdracht.

De bevindingen die dit oplevert kunnen, indien de tijd dit toestaat, worden terug gevoerd naar het technische ontwerp waarin vervolgens verandering kunnen worden aangebracht. Dit kan de tweede deelvraag valideren door bevindingen bij het testen die uitwijzen dat de import/export wizard gemakkelijk te gebruiken is.

Reflectie fase

Gedurende de reflectie fase wordt de tijd voornamelijk gebruikt voor het schrijven van de scriptie.

Taak 10	Scriptie

Het schrijven van de scriptie gebeurt eigenlijk al tijdens de voorgaande fases. Het schrijven van conclusies op de deelvragen gebeurt echter niet en zal dan ook in deze fase worden gedaan.

Zodra de conclusies op de deelvragen zijn opgesteld wordt een conclusie op de hoofdvraag geschreven door de hoofdvraag te valideren op basis van de manier waarop de architectuur de conflicten kan verhelpen.

De scriptie zal uiteindelijk worden gemaakt op basis van het plan van aanpak en de antwoorden op de deelvragen. De conclusies die hieruit gehaald kunnen worden beantwoorden de vragen binnen het project en leiden tot de scriptie.

Taak 11	Presentatie

Voor het presenteren van de bevindingen tijdens de afstudeerzitting wordt als laatst een presentatie gemaakt. Deze presentatie wordt gemaakt met behulp van foto’s van het bedrijf en objecten. Ook zal er in deze presentatie voor elke slide een aantal detail slides zijn waarin naderhand nog extra toelichting kan worden gegeven op specifieke onderdelen.

Theoretisch kader

Om ervoor te zorgen dat er geen tekort is aan kennis voor het behandelen van deze vragen is er in de onderstaande tabel literatuur opgenomen die verdere verdieping biedt voor specifieke taken.

Referentie	Taak
User interface evaluation in the real world	6, 8
Wizard	6, 7, 8
QT Ui design	6, 7
Integrated circuit design	1, 2, 4, 5
Testscript template	9

Er is voor deze specifieke bronnen gekozen omdat ze dicht bij de taken staan die moeten worden uitgevoerd. Hoewel er veel alternatieve literatuur is rondom het ontwikkelen van Qt User interface’s is er gekozen voor deze referentie omdat dit is voortgekomen vanuit de standaard die Qt hanteert waardoor het waarschijnlijk stabieler zal zijn dan een andere aanpak.

Risico

In dit onderdeel worden de projectrisico's besproken die directe invloed kunnen hebben op de voortgang en kwaliteit van het afstuderen.

Er wordt hierbij een kans en impact toegewezen. Deze zijn beiden opgedeeld op een schaal van 1 tot 5 waarbij 5 een erg grote impact/kans is en 1 een erg kleine. Vervolgens worden deze vermenigvuldigd en kan op basis hiervan een inschatting worden gemaakt van de tijdsbesteding.

Er wordt bij deze risico's gesorteerd op Kans*Impact van hoog naar laag.

Methodiek

Risico	Maatregelen	Kans*Impact	Taak
Te veel documentatie voor klein project.	Documentatie volledig richten op oplossen deelvragen.	15 (3*5)	10
	Alleen minimale eisen voor documentatie opnemen.	12 (4*3)	1, 2, 3, 4, 5, 6, 8, 9
Er wordt duidelijk gemaakt dat er nog functionaliteit bij moet.	Terugvallen op gemaakte afspraken rondom requirements. Mogelijk toevoegen bij begin nieuwe iteratie.	10 (5*2)	7, 8, 9
Betrokkenen vanuit het bedrijf zijn niet beschikbaar gedurende gewenste fase.	Mogelijkheid tot vervanging overleggen met bedrijfsbegeleider.	4 (2*2)	1, 2, 3, 9
Geen tijdige toestemming tot afsluiten van een fase.	Alvast vooruitwerken aan elementen met weinig samenhang tot voorgaande fase.	4 (4*1)	3, 6, 7

Kwaliteit

Risico	Maatregelen	Kans*Impact	Taak
Betrokkenen hebben een tekort aan inhoudelijke kennis om het project goed uit te kunnen voeren.	Bronnen opzoeken voor verdere verduidelijkingen.	8 (2*4)	Allemaal
Er is niet genoeg tijd om de minimale eisen uit te voeren.	Mogelijkheden tot verandering functionaliteit/uitstel van deadlines bespreken met begeleider.	8 (2*4)	Allemaal
Er worden andere richtlijnen gehanteerd door betrokkenen.	Richtlijnen vaststellen tussen alle betrokkenen.	6 (3*2)	1, 2, 3, 4, 5, 6, 8, 9
De middelen die gebruikt worden om het project uit te voeren zijn van onvoldoende kwaliteit om dit te doen.	Overleggen met bedrijf voor alternatieve ontwikkelingsopties.	4 (1*4)	7, 8, 9
Kwaliteit eisen worden veranderd vanuit de school/organisatie.	Terugvallen op gemaakte afspraken en dit voorleggen aan docentbegeleider.	1 (1*5)	Allemaal

Technisch

Risico	Maatregelen	Kans*Impact	Taak
Integriteit problemen met bestaande data.	Herstellen naar eerder gemaakte backup zonder deze problemen.	6 (2*3)	Allemaal
Verlies van data (stroomuitval, defecte hardware, etc.).	Triple backups: 1. Externe backup (USB, CD) 2. Bewaren op disk 3. Bewaren in cloud (dropbox)	3 (3*1)	Allemaal
Geen toegang tot bestaande programma code.	Lokale versie van de code bij houden waarmee gewerkt kan worden.	1 (1*3)	4, 5, 7

Planning

Er kan op basis van de producten en risico's een inschatting worden gemaakt van hoeveel tijd er nodig is om dit te kunnen doen. Dit geeft ook direct een indicatie van hoelang bepaalde fases zullen duren.

Product	Geschatte tijd (in uren)	Taak
Plan van aanpak	**120**	**-**
- Aanleiding, probleem, doelstelling, vragen (60)	60	
- Context, algemene onderdelen (20)	20	
- Aanpak en methodiek (20)	20	
- Risico, tijdinschatting, planning (20)	20	
Functioneel ontwerp	**100**	**3**
- Analyse huidige situatie	20	**1**
- Analyse huidige project structuur	20	**2**
- Algemene onderdelen, requirements, afbakening	20	**3**
- Actors, use case diagram, use cases	40	**3**
Technisch ontwerp	**100**	**6**
- Analyse van de huidige project import/export	50	**4**
- Analyse van de mogelijke conflicten	50	**5**
- Initiële design	20	**6**
Eindproduct	**120**	**7**
- Uitwerken klassendiagram	20	**7**
- Use case uitwerkingen	100	**7**
Testplan	**40**	**8**
Testrapport	**40**	**8**
- Opstellen testify	20	**8**
- Uitvoeren testen en documentatie	20	**8**
Design	**60**	**9**
- Testscript en verbeteringen	60	**9**
Scriptie	**120**	**10**
Presentatie	**20**	**11**

Dit vult echter niet alle tijd op die er staat voor het afstuderen en geeft daardoor dus ook de mogelijkheid tot flexibiliteit (mocht een inschatting verkeerd zijn). Indien de inschattingen lager uitvallen is het mogelijk extra functionaliteiten te implementeren door het proces van bovenaf te herhalen maar dan voor andere functionaliteit.

Het maken van het functioneel en technisch ontwerp is hier een hoog aantal uren toegekend om ervoor te zorgen dat dit goed kan worden overlegd. Het design vereist een hoop interactie met de klanten en het testen van dit proces zal ook tijd kosten.

Vanuit de afstudeerleidraad komen een aantal deadlines voor producten. De deadlines voor de specifieke producten op basis van de tijdsinschatting zijn als volgt:

Product	Deadline
Plan van Aanpak	20 maart 2015
Functioneel ontwerp	6 maart 2015
Technisch ontwerp	20 maart 2015
Eindproduct	22 mei 2015
Testplan	22 mei 2015
Testrapport	22 mei 2015

Design	22 mei 2015
Concept scriptie	20 april 2015
Concept scriptie	4 mei 2015
Scriptie	2 juni 2015
Presentatie	5 juni 2015

Omdat er iteratief gewerkt wordt kunnen groen en rood in een daadwerkelijk planning worden geïnterpreteerd als iteratie ruimte (waarbij minimaal een enkele iteratie als basis is geplant). Voor de planning wordt de onderstaande legenda gebruikt:

Product	Kleur
Plan van aanpak	PvA
Functioneel ontwerp	FO
Technisch ontwerp	TO
Design	Design
Testplan	Test Plan
Testrapport	Test Rap
Eindproduct	App
Scriptie	Scriptie
Presentatie	Presentatie

Hieronder is een overzicht van alle maanden opgenomen die staan voor het afstuderen inclusief de bijbehorende activiteiten. Gedurende de eerste week van het project is een introductie project uitgevoerd.

Februari

Ma	Di	Wo	Do	Vr	Za	Zo
9	10	11	12	13	14	15
PvA	PvA	PvA	PvA	PvA		

Ma	Di	Wo	Do	Vr	Za	Zo
16	17	18	19	20	21	22
PvA	PvA	PvA	PvA	PvA		
FO	FO	FO	FO	FO		

Ma	Di	Wo	Do	Vr	Za	Zo
23	24	25	26	27	28	
PvA	PvA	PvA	PvA	PvA		
FO	FO	FO	FO	FO		

Maart

Ma	Di	Wo	Do	Vr	Za	Zo
2	3	4	5	6	7	8
PvA	PvA	PvA	PvA	PvA		
FO	FO	FO	FO	FO		

Ma	Di	Wo	Do	Vr	Za	Zo
9	10	11	12	13	14	15
PvA	PvA	PvA	PvA	PvA		
TO	TO	TO	TO	TO		

Ma	Di	Wo	Do	Vr	Za	Zo
16	17	18	19	20	21	22
PvA	PvA	TO	TO	TO		
TO	TO	Design	Design	Design		

Ma	Di	Wo	Do	Vr	Za	Zo
23	24	25	26	27	28	29
Design	Design	Design	Design	Design		
App	App	App	App	App		

Ma	Di
30	31
App	App
Scriptie	Scriptie

April

Wo	Do	Vr	Za	Zo
1	2	3	4	5
App	App	App		
Scriptie	Scriptie	Scriptie		

Ma	Di	Wo	Do	Vr	Za	Zo
6	7	8	9	10	11	12
App	App	App	App	App		
Scriptie	Scriptie	Scriptie	Scriptie	Scriptie		

Ma	Di	Wo	Do	Vr	Za	Zo
13	14	15	16	17	18	19
App	App	App	App	App		
Scriptie	Scriptie	Scriptie	Scriptie	Scriptie		

Ma	Di	Wo	Do	Vr	Za	Zo
20	21	22	23	24	25	26
App	App	App	App	App		
Scriptie	Scriptie	Scriptie	Scriptie	Scriptie		

Ma	Di	Wo	Do
27	28	29	30
App	App	App	App
Scriptie	Scriptie	Scriptie	Scriptie

Mei

Vr	Za	Zo
1	2	3
Test Plan		
Design		

Ma	Di	Wo	Do	Vr	Za	Zo
4	5	6	7	8	9	10
Test Plan	Test Plan	Test Plan	Test Plan	Test Plan	4	
Design	Design	Design	Design	Design		

Ma	Di	Wo	Do	Vr	Za	Zo
11	12	13	14	15	16	17
Design	Design	Design	Design	Design		
Scriptie	Scriptie	Scriptie	Scriptie	Scriptie		

Ma	Di	Wo	Do	Vr	Za	Zo
18	19	20	21	22	23	24
Test rap	Test rap	Test rap	Test rap	Test rap		
Scriptie	Scriptie	Scriptie	Scriptie	Scriptie		

Ma	Di	Wo	Do	Vr	Za	Zo
25	26	27	28	29	30	31
Scriptie	Scriptie	Scriptie	Scriptie	Scriptie		

Juni

Ma	Di	Wo	Do	Vr	Za	Zo
1	2	3	4	5		
Scriptie	Scriptie	Presentatie	Presentatie	Presentatie		

Bronnenlijst

Agile methode. (n.d.). Retrieved from
http://en.wikipedia.org/wiki/Agile_software_development

Steve Krug (2010) Usability test script. In Rocket Surgery Made Easy. Retrieved from
http://sensible.com/downloads/test-script.pdf

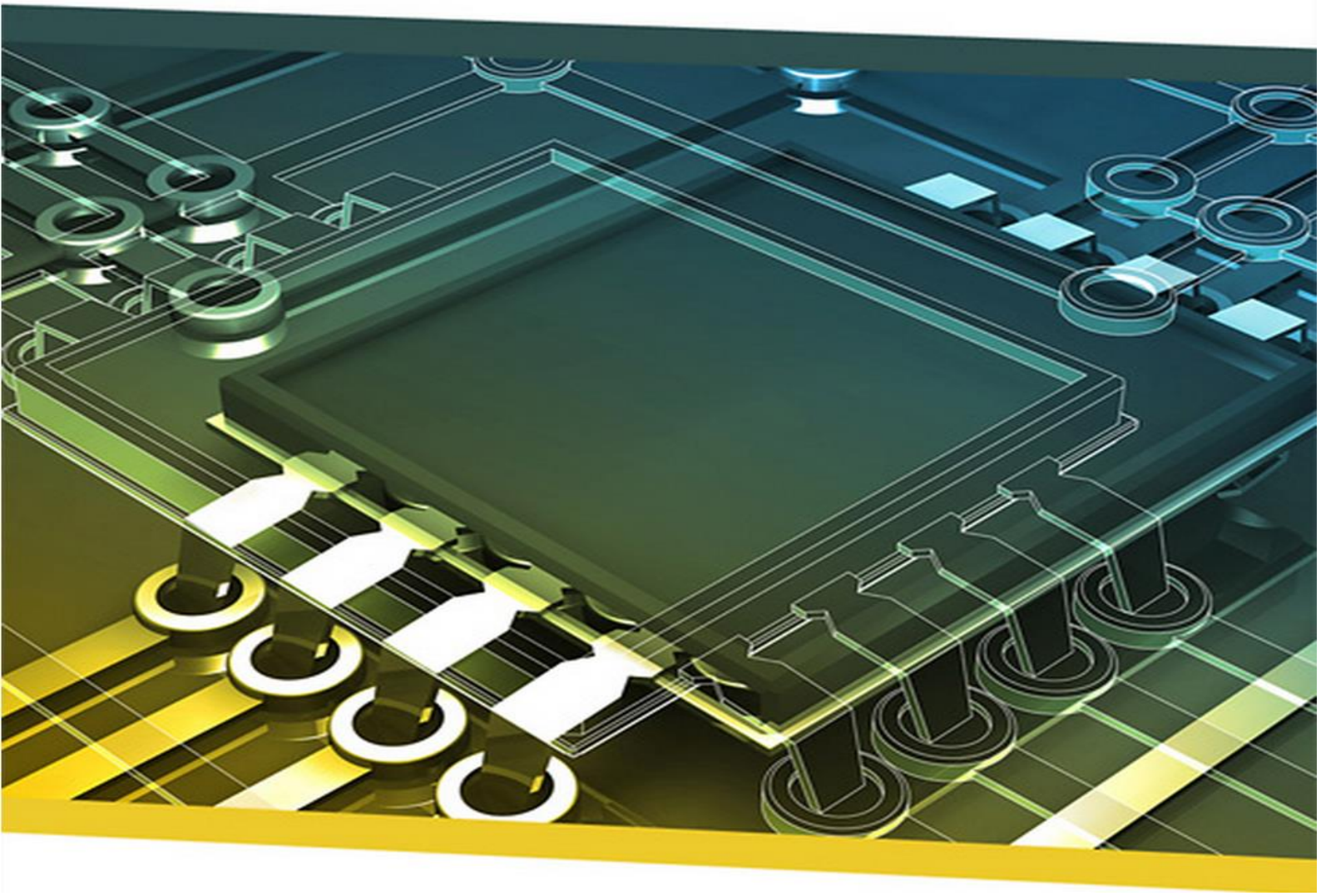
Robin Jeffries, James R. Miller, Cathleen Wharton, Kathy Uyeda (1991) User interface evaluation in the real world. In ACM Digital library. Retrieved from
http://dl.acm.org/citation.cfm?id=108862

Wizard. (n.d.). In Designing interfaces. Retrieved from
http://designinginterfaces.com/patterns/wizard/

QT Ui design (n.d.). In Qt Documentation. Retrieved from
http://doc.qt.io/qt-5/qt-gui-concepts.html

Integrated circuit design. (n.d.). In Wikipedia. Retrieved from
http://en.wikipedia.org/wiki/Integrated_circuit_design

12.2. Bijlage 2 – Functioneel ontwerp



Project import/export

Functioneel ontwerp

Student

Naam:	Velleman
Voorletters:	F.M.V.
Roepnaam:	Floris
Studentnummer:	1602376
Docentbegeleider:	Alex Jongman

Bedrijf

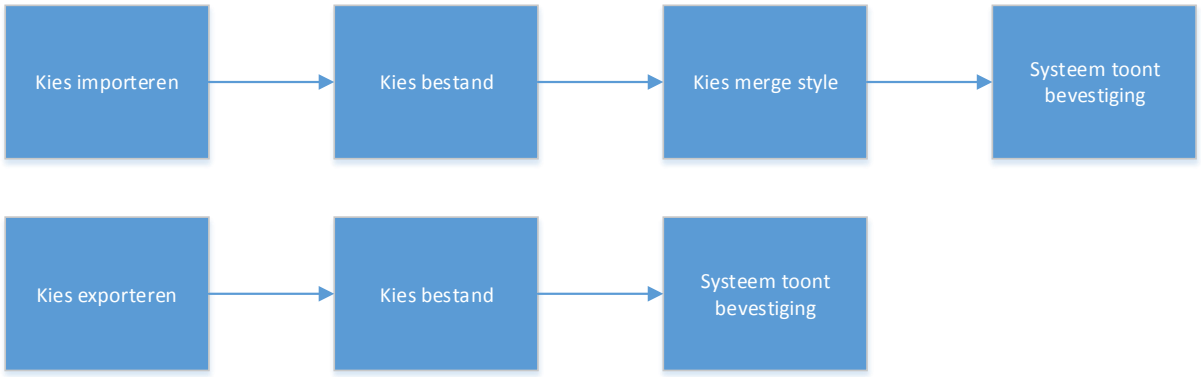
Naam:	NP-Komplete technologies
Adres:	De Rozentuin 18
Postcode:	5611 SW
Plaats:	Eindhoven
Inhoudelijk bedrijfsbegeleider:	Maarten Berkens
Technisch bedrijfsbegeleider:	Rob Gelderblom

Huidige situatie

Binnen iDRM zijn er verschillende mogelijkheden als het aankomt op importeren en exporteren. Zo kan er simpelweg in het menu op importeren/exporteren worden gedrukt maar kan dit ook gebeuren vanuit de lijsten met objecten. Zo kan daar momenteel een enkel element worden ge-export.

Op deze manier zijn er dan ook meerdere manieren waarop export bestanden kunnen worden gemaakt en daarom is er andere inhoud. Er bestaat momenteel dus de mogelijkheid om onderscheid te maken tussen de verschillende inhoud van de files.

Als iemand een export of import wil doen gebeurt dat momenteel door deze opties. Gezien dit een handig systeem blijkt te zijn zal er op dit gebied weinig hoeven veranderen. De uitzondering hier is dat er wel moet worden uitgebreid op het aantal opties. Het huidige proces kan dan ook als volgt worden weergegeven:



Er kan in dit diagram al snel worden opgemaakt dat een hoop opties ontbreken. Zo is er enkel de optie om te kiezen voor het automatisch renamen of mergen. Dit lijkt binnen het programma altijd goed te gaan maar wat er precies goed is gegaan wordt niet vermeld.

Los hiervan is meerdere elementen importeren enkel een optie binnen het menu (door middel van project import). Zo is er de mogelijkheid tot exporten van vrijwel elk element binnen het programma. Er is echter geen mogelijkheid om op die plekken ook weer te kunnen importen.

Voor het importeren van layermaps is er een optie onder CAD Layers. Er is hier echter geen export knop terwijl dit hier best een optie had kunnen zijn. Deze knop is onder het file menu geplaatst en export de layermap.

Business use cases

Gezien de vraag komt vanuit de gebruikers van het programma is het van belang om te bepalen welke handelingen zij willen uitvoeren binnen het programma.

Om dit duidelijk te kunnen weergeven is de volgende tabel gemaakt waarin de project staat is uitgezet tegen de actie die wordt uitgevoerd door de gebruiker.

Project	Actie
Leeg project	Import bestaand project.
Leeg project	Hergebruik elementen uit ander project.
Bestaand project	Verbeter op basis van ander project.
Bestaand project	Alles importeren dat geen conflicten geeft.
Bestaand project	Alles hernoemen dat conflicten geeft.
Bestaand project	Kies welke elementen (als elementen samenhangen dan worden die andere ook meegenomen).
Bestaand project	Importeren van alle patterns (zonder de layers die moeten op unspecified voor patterns).

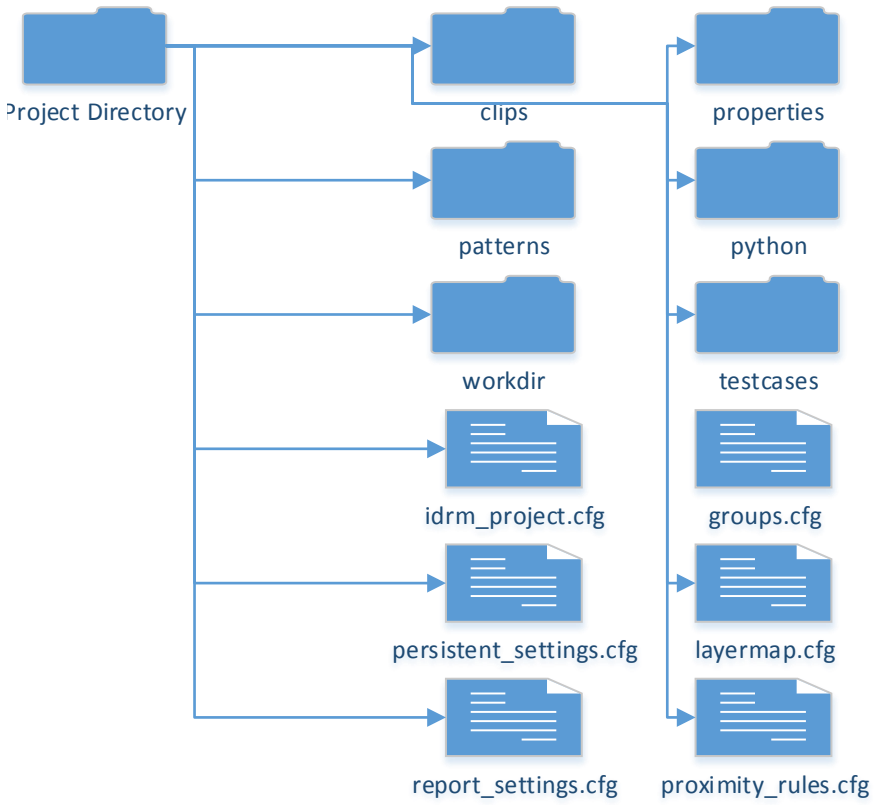
Het is opvallend dat de gebruiker dus niet erg vaak daadwerkelijk alles zal willen aanpassen. Toch wil de gebruiker als het er op aan komt wel opties hebben voor het veranderen van deze gegevens.

Analyse huidige structuur

In dit onderdeel wordt weergegeven welke bestanden en samenhang er momenteeel is in het programma. Dit gebeurt door het programma en de bijbehorende bestanden te analyseren en te overleggen met de klant/opdrachtgever.

Om te begrijpen hoe de import en export momenteel werken is het van belang te weten welke bestanden er worden gebruikt door het project.

Om dit te doen is een project opgesteld waarin de volledige project structuur wordt gebruikt. In dit voorbeeld project is de volgende structuur op te maken:



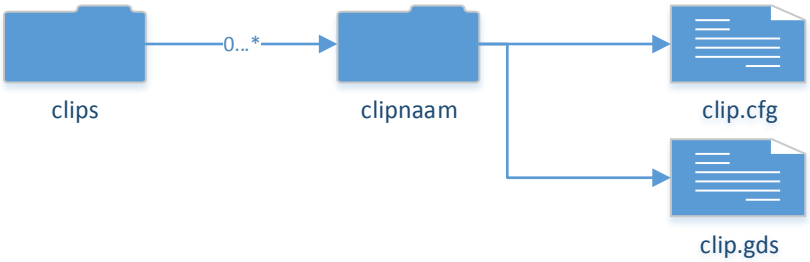
Hierbij is aangegeven dat cfg staat voor configuration. De cfg bestanden zouden moeten worden meegenomen tijdens het import/export proces. Om verdere duidelijkheid te krijgen over de structuur die hier is beschreven kan voor elk van de subfolders een nieuwe diagram worden opgesteld.

Om aan te geven dat het over bijvoorbeeld meerdere patterns gaat is er gebruik gemaakt van een asterisk (*). De GDS files worden gebruikt om een stukje van een chip layout in te bewaren.

Er kan per folder worden gekeken naar waar dit in het programma voorkomt en hoe het momenteel werkt. Dit geeft een beter beeld van hoe er kan worden uitbereid en wat er beter kan.

Clips

De elementen in deze folder worden momenteel niet meer gebruikt. Ze hoeven dan ook nog niet in de import/export te komen. Het is wel belangrijk om hier rekening mee te houden. Dit zou op een vergelijkbare manier kunnen worden gedaan als patterns gezien het een vergelijkbare structuur heeft.



Conflicten zouden hier kunnen voorkomen bij het importeren van een clipnaam die al bestaat. Aangezien de elementen in de folder een vaste naam hebben zal een kopie maar beperkt nut hebben ten opzichte van andere elementen.

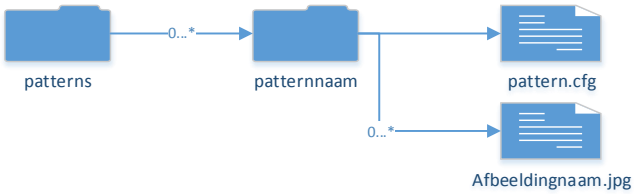
Een kopie zou wel kunnen maar dan zou de gebruiker handmatig een juiste versie moeten kiezen (door middel van renamen bijvoorbeeld). Dit zou ook handig kunnen zijn indien de gebruiker een kopie wil maken van het origineel.

Binnen het programma zelf komen clips voor in een apparte tab. Deze tab is echter nog niet af en daarom kan er nog niet heel veel worden gezegd over de daadwerkelijke toepassing van deze onderdelen. Los van het feit dat de structuur van deze clips nog kan veranderen.

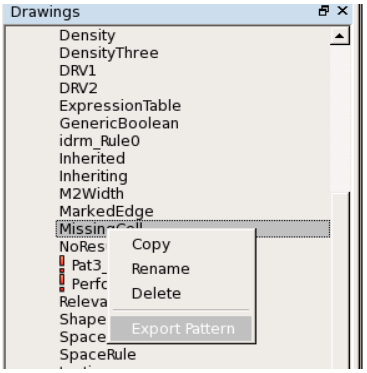
Patterns

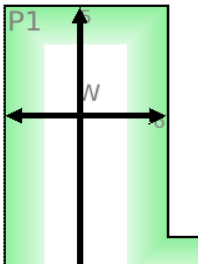
Deze map wordt veel gebruikt binnen het programma aangezien een pattern de basis vormt voor een rule instance. Het is van groot belang deze in de import/export te hebben. De folder zelf bevat voor elk pattern een mapje met daarin de details van het object in een cfg file.

Er kan op basis van deze informatie binnen het programma een pattern worden gemaakt.



Patterns hebben binnen het programma een appart tabje. De rules zijn volledig gebasseerd op een pattern en dit zorgt er ook voor dat dit de hoogste prioriteit heeft. Er is voor patterns de mogelijkheid om deze te kunnen exporten binnen de lijst.





[4] - Export pattern

Er is echter niet de mogelijkheid om meerdere elementen te selecteren (en deze te exporten). Ook is er niet de mogelijkheid om te importeren in de UI (uitsluitend via het hele project of handmatig).

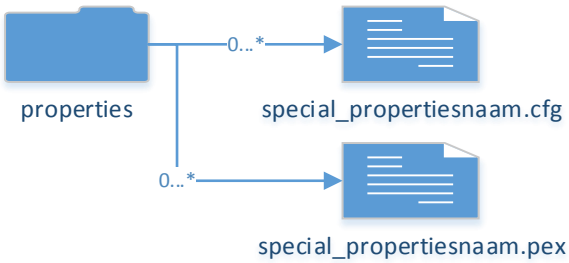
Er zijn meerdere conflicten die zich kunnen voordoen binnen deze folder. Zodra een pattern al voorkomt in het huidige project moet worden gekeken naar de pattern configuratie en eventuele afbeeldingen.

Dit komt vooral aan op de layermap koppeling die een pattern maakt. Indien er al bestaande elementen zijn in deze lijst kan dit problemen veroorzaken. De meeste conflicten komen hier dan ook voort uit de samenhang van een pattern met andere elementen.

Properties

Bij properties wordt een bestand onthouden waarmee extra syntax kan worden toegevoegd aan het programma. Deze syntax is bedoeld voor conditionele expressies en biedt een optie voor klanten om het verder te kunnen uitbreiden.

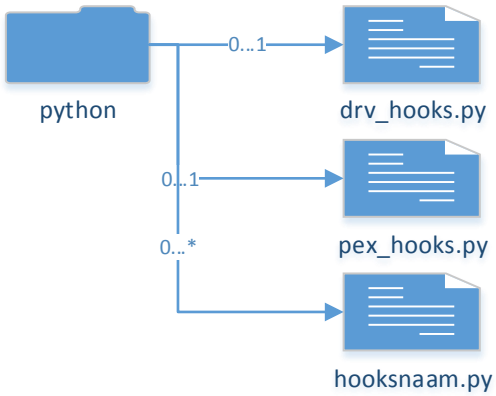
Binnen het programma zijn deze special properties niet veranderbaar maar ze kunnen wel worden gebruikt door ze te slepen vanuit het proximity rule widget.



Bij properties kunnen conflicten ontstaan doordat een special property al bestaat. Dit kan zowel door de inhoud als door de naam. Het beste dat kan worden gedaan voor de pex file is het vergelijken van de size van de files.

Python

De python folder bevat twee bestanden die verantwoordelijk zijn voor het pre en post processen van het vinden van een rule.



[4] Afbeelding

De drv_hooks en pex_hooks zijn de standaard bestanden die kunnen verwijzen naar andere .py bestanden. Dit zijn ook de enige bestanden waar het programma in eerste instantie naar zal zoeken.

Er kunnen hier conflicten ontstaan doordat een file al bestaat. De inhoud van deze files zou kunnen worden vergeleken.

Testcases

De testcases zijn de binnen het programma bepaalde testarea's. Deze vallen onder een teststructure (folder) die vervolgens alle testarea's bevat.

Een structure kan nul of meerdere testcases bevatten en kan conflicten opleveren als er al een zelfde structure is. Tevens kan een testcase een conflict opleveren. Er kan in dit geval het beste worden vergeleken op inhoud.

Binnen het programma is het mogelijk zowel een test structure als een test area te exporteren. Door het exporteren van een test structure is het dus eigenlijk al mogelijk om meerdere test area's tegelijk te exporteren. Het exporteren van meerdere test structures is nog geen optie.

Projectdirectory

In de projectdirectory zelf zijn nog een aantal files opgenomen die allemaal een vaste naam hebben en met uitzondering van proximity rules altijd standaard aanwezig zijn.

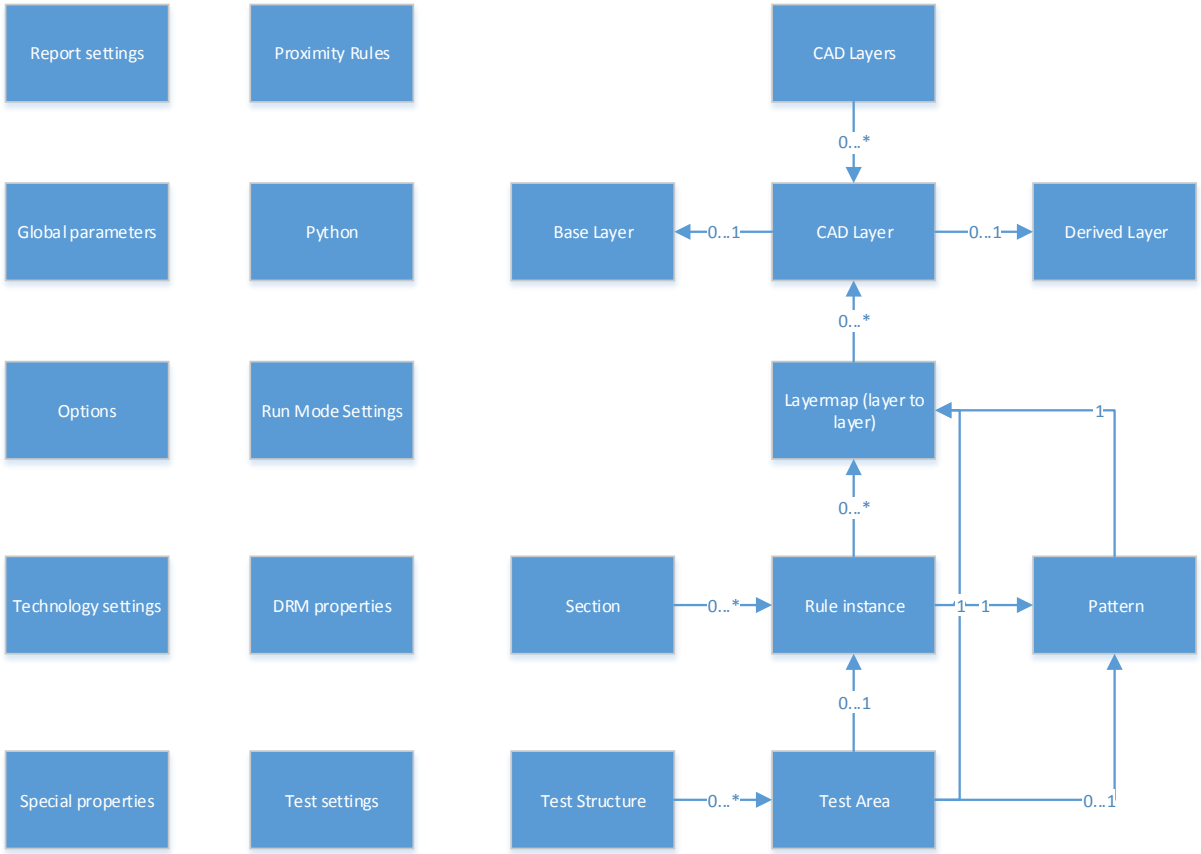
In het configuratie bestand voor layermap worden meerdere layermaps gedefinieerd. Dit kan zorgen voor conflicten die nog niet eerder zijn vermeld.

De layermap en de persistent settings hebben beiden binnen het programma een eigen widget waarop ze worden getoond en kunnen worden geimporteerd en geexporteerd.

Werkelijke samenhang van elementen

Op basis van de gesprekken die gevoerd zijn met zowel de klant als het bedrijf is een diagram opgesteld waarin alle afhankelijkheden staan die te maken hebben met import/export.

Dit diagram is door de analyse van de inhoud van de bestanden verbeterd en tot een uiteindelijk overzicht van de projectstructuuur gebouwt.



Momenteel worden deze elementen allemaal gebruikt binnen het programma. De import en export functionaliteit van deze onderdelen verschilt. Om duidelijkheid te krijgen in wat er momenteel mogelijk is met de bestaande elementen is een tabel opgesteld.

Hierbij is gekeken of het element binnen het programma import/export functionaliteit heeft of dat het enkel bij het gehele project gebeurt. Ook is gekeken hoe het importeren van de bestanden werkt. Zo is het mogelijk dat bepaalde instellingen alleen worden ingeladen bij een leegproject.

Indien dit als nee is opgegeven werkt de import/export bij elk project (leeg of met elementen). Een belangrijk punt in dit diagram is het pattern. Het pattern is eigenlijk een DRM rule instance die een pattern heeft. Dit kan echter samen worden gezien als een pattern voor het importeren/exporteren.

Onderwerp	Import in UI	Export in UI	Import/Export door project	Leeg project
CAD Layers (Layers algemeen)	Ja	Ja	Ja	Nee
Layermap	Ja	Nee	Ja	Nee
Section	Nee	Ja	Ja	Nee
Rule instance	Nee	Ja	Ja	Nee
Pattern	Nee	Ja	Ja	Nee
Test structure	Nee	Ja	Ja	Nee
Test area	Nee	Ja	Ja	Nee
Global parameters	Ja	Ja	Ja	Nee
Options	Nee	Nee	Ja	Ja
Technology settings	Nee	Nee	Nee	Nee
Special properties	Ja	Nee	Ja	Ja
Run Mode Settings	Ja	Ja	Nee	Ja
DRM properties	Nee	Nee	Ja	Ja
Test settings	Nee	Nee	Ja	Ja
Python	Nee	Nee	Ja	Nee
Report settings	Nee	Nee	Ja	Ja
Proximity Rules	Nee	Nee	Nee/Ja	Nee

Een opvallend punt in deze tabel is ‘Technology settings’ aangezien hier helemaal niets mee kan gebeuren. Dit is blijkbaar een bug omdat dit wel zou moeten worden ingeladen bij een leeg project.

Requirements

In dit hoofdstuk worden de functionele eisen aan het systeem beschreven. Hierbij wordt onderscheid gemaakt tussen functionele eisen en niet functionele eisen.

Functional requirements

- Er moet een gebruikersinterface zijn voor het importeren, exporteren en linken van projectelementen.
- Er moet per project element een keuze zijn voor de manier waarop het wordt geïmporteerd in het geval er een conflict is (origineel behouden, overschrijven, hernoemen, toevoegen aan het project).
- Per project element moet de optie bestaan deze wel of niet te importeren/exporteren.
- Bij het importeren van patterns moet het mogelijk zijn zonder layers te importeren.
- Bij het importeren van rule instances moet het mogelijk zijn zonder layers te importeren.
- Bij het importeren van test area's moet het mogelijk zijn te importeren zonder rule instance(s).
- Bij het uitschakelen van een element tijdens het importeren moet het mogelijk zijn de hieraan verbonden elementen snel te kunnen veranderen door een andere link te maken voor deze objecten.
- Groeperingsmechanisme voor de projectelementen.
- Zoekfunctie voor het filteren van projectelementen.

Non-functional requirements

- Het programma mag niet crashen bij normaal gebruik (uitgaande dat er hier altijd een redelijke hoeveelheid geheugen is).
- De gebruikersinterface moet fout ontwijkend zijn.
- Binnen de interface moet voor de gebruiker snel duidelijk zijn hoe er gewerkt moet worden met de interface.
- De interface mag niet gebruik maken van elementen die de testbaarheid van de interface in gevaar brengen.
- De code moet robuust zijn en berekend op fouten vanuit de gebruiker.

Functionaliteit

Er zijn binnen deze opdracht een aantal functionele opdrachten opgesteld. Deze komen voornamelijk vanuit de klant en de bedrijfsbegeleider. Binnen de opdrachten valt een opdeling te maken met wat er prioriteit heeft.

Zo is er een basis gedeelte dat minimaal moet worden gemaakt tijdens de afstudeerstage maar is er ook extra functionaliteit gevraagd. Dit komt voornamelijk uit wensen van de klant.

Must have

- Er moet een gebruikersinterface zijn voor het importeren van projectelementen (Rule instance(s), Layer(s) en Pattern(s)).
- Er moet per project element een keuze zijn voor de manier waarop het wordt geïmporteerd in het geval er een conflict is (origineel behouden, overschrijven en hernoemen).
- Per project element moet de optie bestaan deze wel of niet te importeren.
- Bij het importeren van patterns moet het mogelijk zijn zonder layers te importeren.
- Bij het importeren van rule instances moet het mogelijk zijn zonder layers te importeren.
- Bij het uitschakelen van een element tijdens het importeren moet het mogelijk zijn de hieraan verbonden elementen snel te kunnen veranderen door een andere link te maken voor deze objecten.
- Het programma mag niet crashen bij normaal gebruik (uitgaande dat er hier altijd een redelijke hoeveelheid geheugen is).
- De interface mag niet gebruik maken van elementen die de testbaarheid van de interface in gevaar brengen (Elk interface object moet een object naam hebben).

Should have

- Er moet een gebruikersinterface zijn voor het importeren van projectelementen (Rule instance(s), Layer(s), Pattern(s) en Test area(s)).
- Bij het importeren van test area’s moet het mogelijk zijn te importeren zonder rule instance(s).
- De gebruikersinterface moet fout ontwijkend zijn.

Could have

- Groeperingsmechanisme voor de projectelementen.
- Binnen de interface moet voor de gebruiker snel duidelijk zijn hoe er gewerkt moet worden met de interface.
- Zoekfunctie voor het filteren van projectelementen.
- De code moet robuust zijn en berekend op fouten vanuit de gebruiker.

Won’t have

- Exporteren/Linken van een project.
- Import/Export/Linken van Clips.

Actors

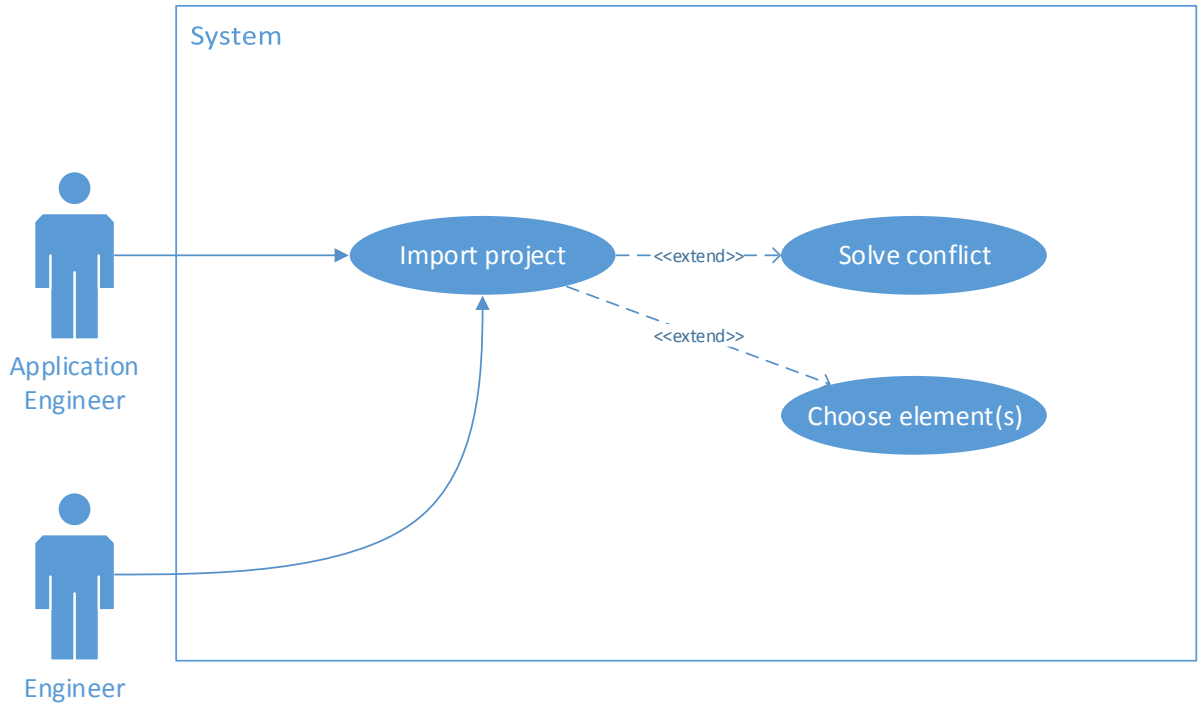
Er zijn een aantal actors binnen dit project. Zo is de klant een actor maar ook de medewerkers van NP-Komplete Technologies (AE's).

Naam:	Application engineer
Rol:	Gebruiker
Beschrijving:	Een application engineer is een medewerker van NP-Komplete technologies. Deze application engineer gaat om met de klanten en geeft demo's. Tevens geven deze gebruikers cursussen en werken soms zelfs bij de klanten. Ze zijn de best voorgelichte gebruikers op het gebied van het programma.

Naam:	Engineer
Rol:	Gebruiker
Beschrijving:	Een engineer is iemand met een technische opleiding en enige ervaring met het programma. Deze gebruikers zijn meestal verantwoordelijk voor het gebruik van het programma binnen de organisatie van de klant en werken hier ook.

Use case diagram

Op basis van de actoren en gevraagde functionaliteit kan een use case diagram worden opgesteld. Hierin komen alle onderdelen terug die leiden tot de functionaliteit zoals beschreven in de requirements.

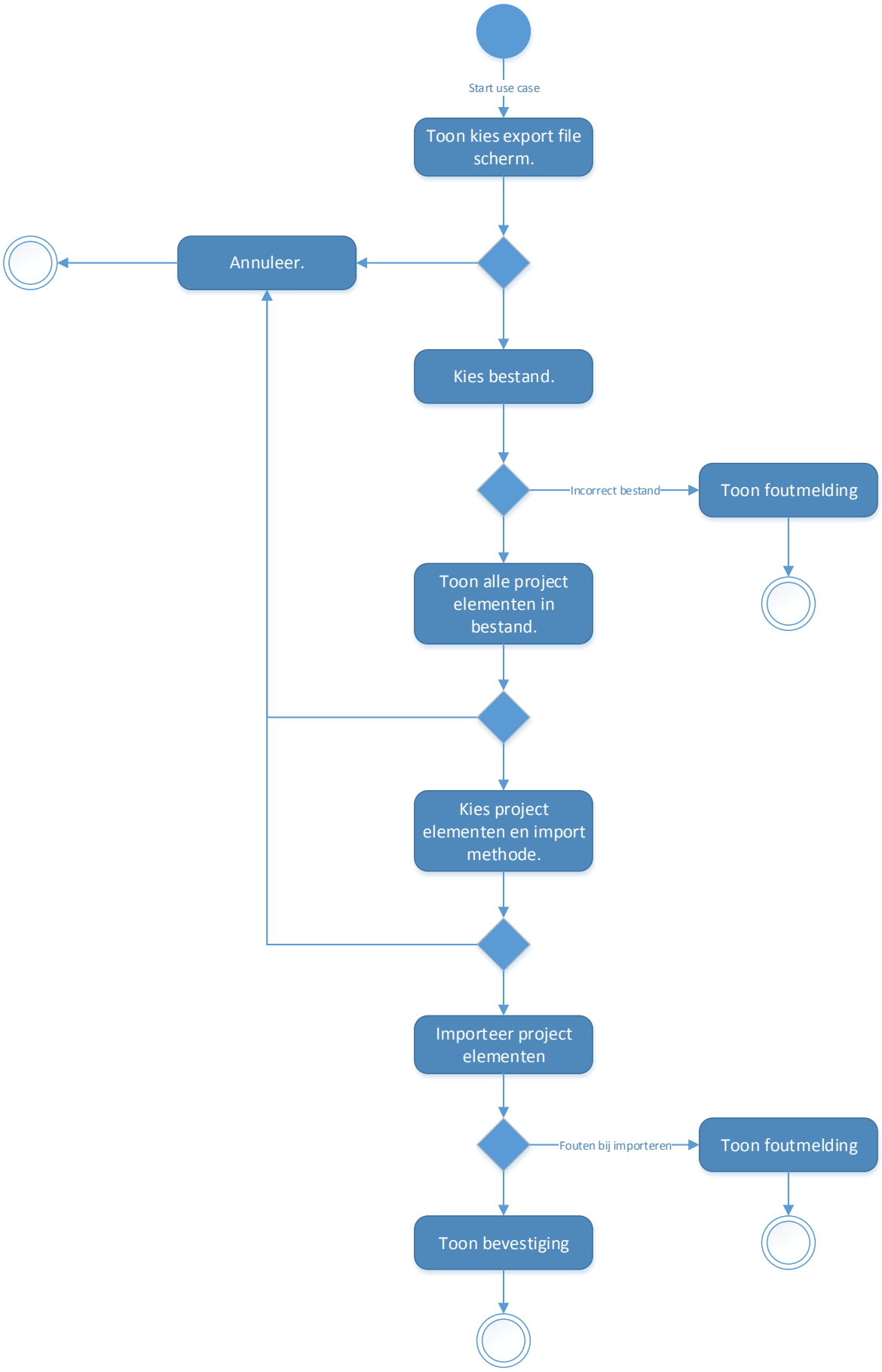


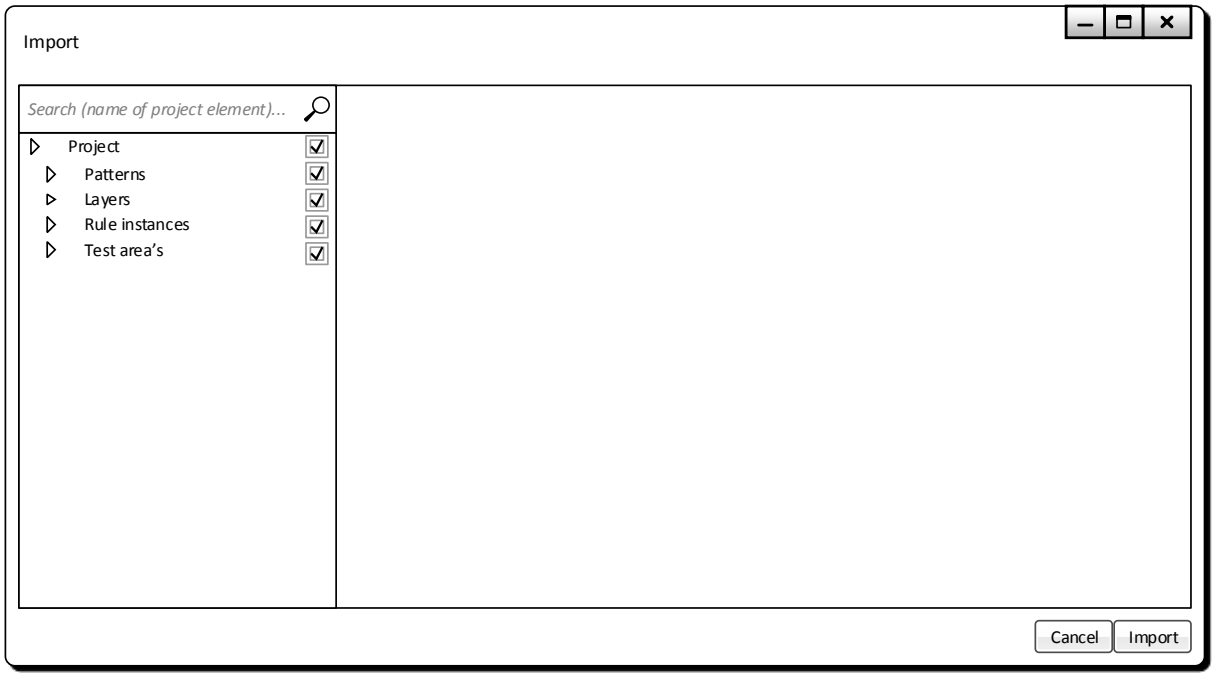
Use cases

Vanuit het use case diagram kan de specificaties van de functionaliteit worden uitgewerkt in drie use cases.

Import project

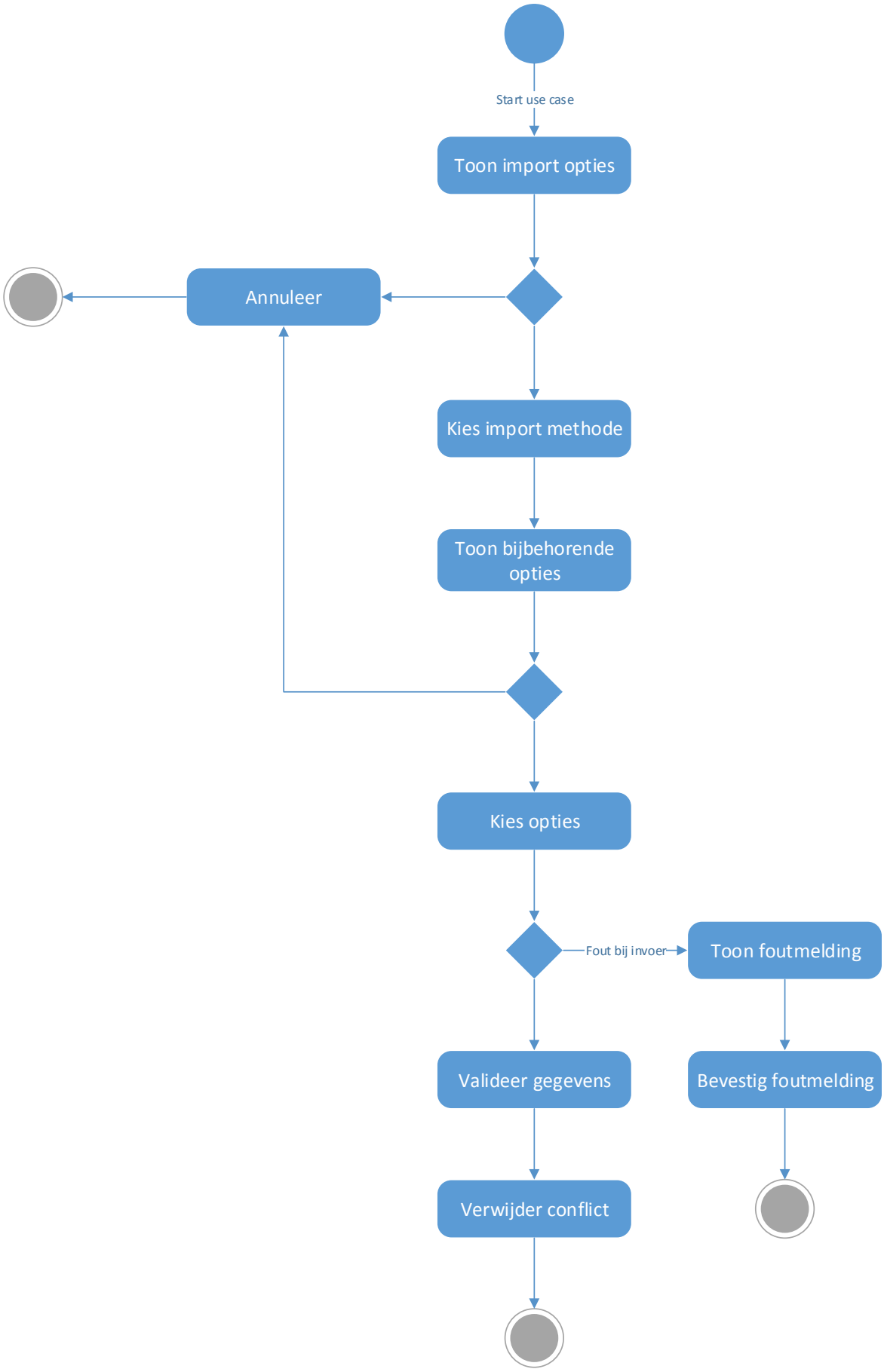
Use case naam	Import project	
Auteur	Floris Velleman	
Versie	1.0	
Doelstelling	Het doel van deze use case is het importeren van nieuwe elementen in het project dat open staat.	
Actor(s)	Application Engineer, Engineer	
Preconditie	Er is een nieuw project gemaakt of een bestaand project geladen. Er is vervolgens op importeren gedrukt in het File menu.	
Hoofdscenario	1.1 2.1 3.1 4.1 5.1 5.2 6.1 6.2	Actor start use case. Systeem toont scherm voor het kiezen van een export file. Actor kiest een bestand. Systeem toont alle projectelementen in het bestand. Actor kiest projecten elementen en import wijze (Start use case(s)). Actor bevestigd. Systeem importeerd projectelementen. Systeem toont melding.
Postconditie	De projectelementen uit het bestand zijn op de door de actor beschreven manier geïmporteerd in het huidige project.	
Alternatiefscenario 1 (Bij stap 3.1, stap 5.1 en stap 5.2 van HS)	3.1	Actor annuleert.
Postconditie	Er zijn geen bestanden geïmporteerd.	
Alternatiefscenario 2 (Bij stap 4.1 van HS) – Incorrect bestand (of leeg)	4.1 5.1	Systeem toont foutmelding. Actor bevestigd.
Postconditie	Er zijn geen bestanden geïmporteerd.	
Alternatiefscenario 3 (Bij stap 6.1) – Fouten bij importeren	6.2 7.1	Systeem toont foutmelding. Actor bevestigd.
Postcondinitie	Er zijn geen bestanden geïmporteerd.	

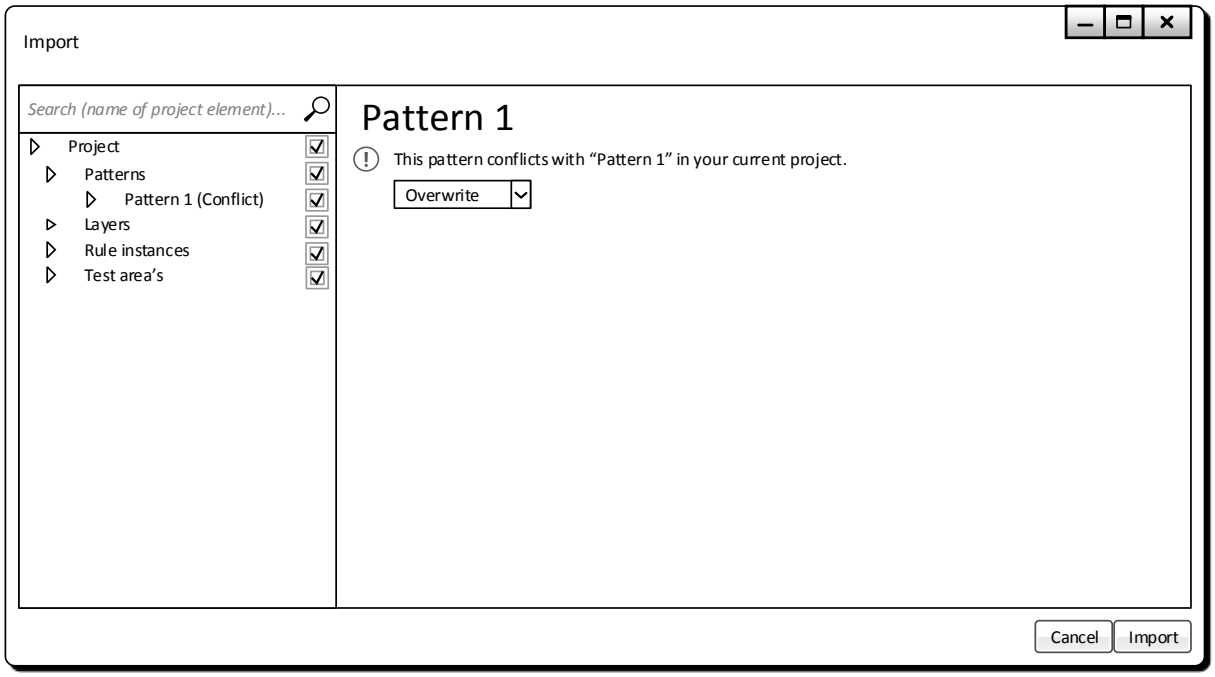

Start use case
Toon kies export file scherm.
Annuleer.
Kies bestand.
Incorrect bestand
Toon foutmelding
Toon alle project elementen in bestand.
Kies project elementen en import methode.
Importeer project elementen
Fouten bij importeren
Toon foutmelding
Toon bevestiging


Import
Search (name of project element)...
Project
Patterns
Layers
Rule instances
Test area's
Cancel
Import

Solve conflict

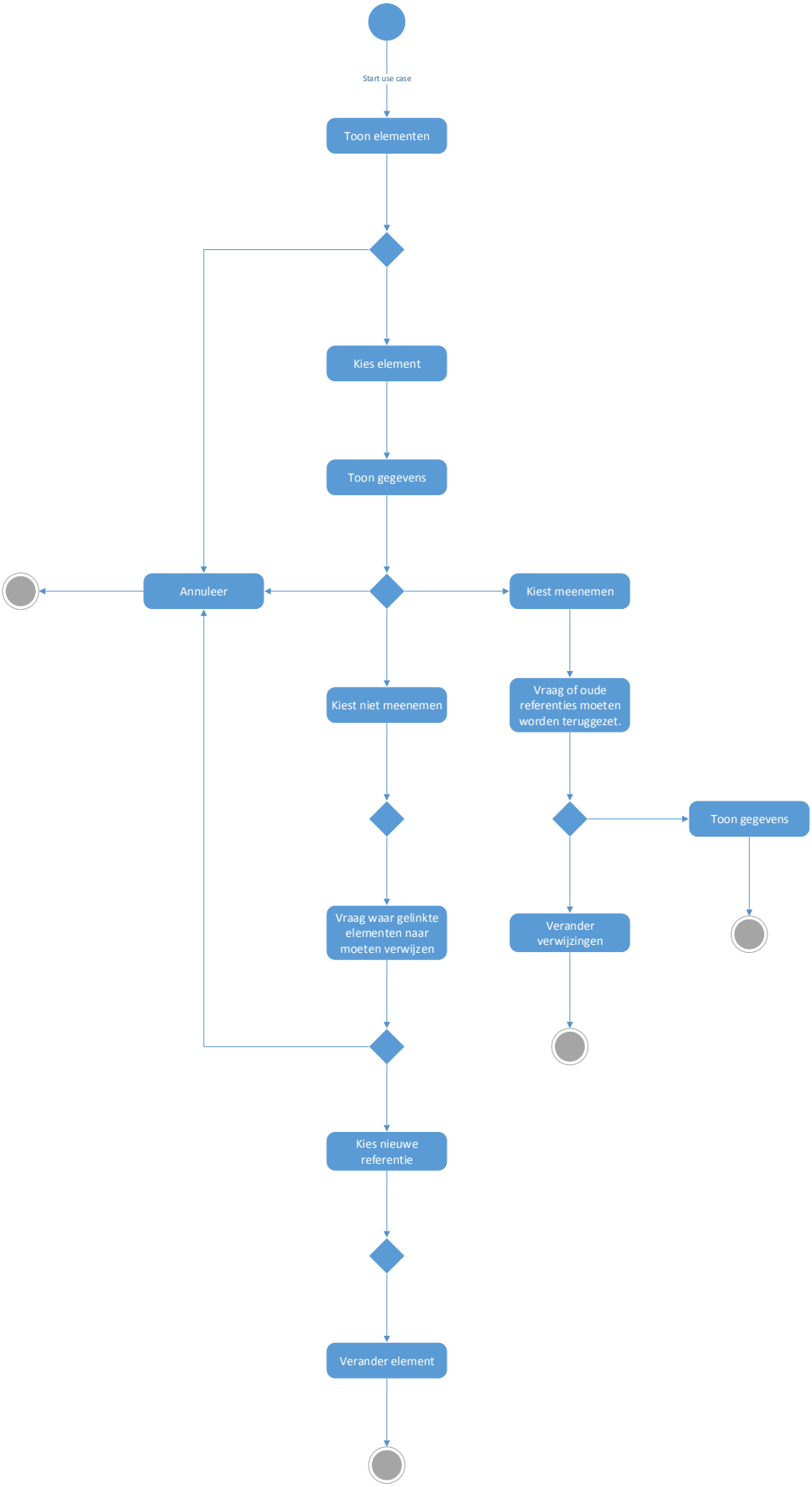
Use case naam	Solve conflict	
Auteur	Floris Velleman	
Versie	1.0	
Doelstelling	Het doel van deze use case is het oplossen van een conflict dat een element veroorzaakt.	
Actor(s)	Application Engineer, Engineer	
Preconditie	Er is gekozen voor importeren en er zijn hierbij conflicten gevonden. De gebruiker heeft een element met een conflict geselecteerd.	
Hoofdscenario	1.1 2.1 3.1 4.1 5.1 6.1 6.2	Actor start use case. Systeem toont import opties. Actor kiest import methode. Systeem toont bijbehorende import opties (overwrite, merge en keep original). Actor kiest bijbehorende opties. Systeem valideert ingevulde gegevens. Systeem verwijdert conflict.
Postconditie	Het conflict is opgelost.	
Alternatiefscenario 1 (Bij stap 3.1 en stap 5.1 van HS)	X.1	Actor annuleert.
Postconditie	Het conflict is niet opgelost.	
Alternatiefscenario 2 (Bij stap 6.1 van HS) – Fout bij ingevulde gegevens	7.1 8.1	Systeem toont foutmelding. Actor bevestigd.
Postconditie	Het conflict is niet opgelost.	


Start use case
Toon import opties
Annuleer
Kies import methode
Toon bijbehorende opties
Kies opties
Fout bij invoer
Toon foutmelding
Valideer gegevens
Bevestig foutmelding
Verwijder conflict

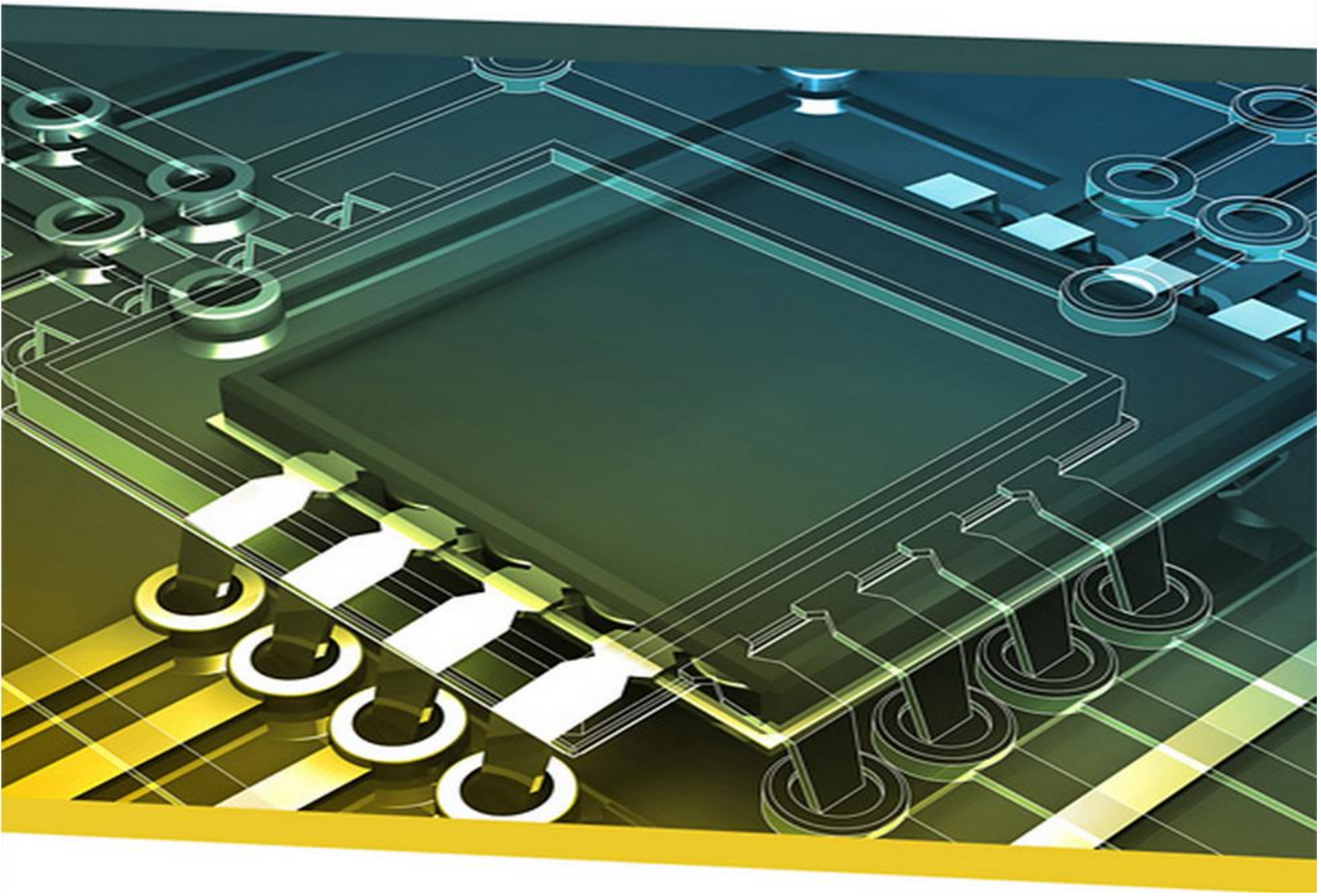

Import
Search (name of project element)...
Project
Patterns
Pattern 1 (Conflict)
Layers
Rule instances
Test area's
Pattern 1
This pattern conflicts with "Pattern 1" in your current project.
Overwrite
Cancel
Import

Choose element(s)

Use case naam	Choose element(s)	
Auteur	Floris Velleman	
Versie	1.0	
Doelstelling	Het doel van deze use case is het kiezen van welke element(s) er worden geïmporteerd/geëxporteerd. Standaard worden alle elementen meegenomen.	
Actor(s)	Application Engineer, Engineer	
Preconditie	Er is een bestaande structuur geladen in het geheugen.	
Hoofdscenario	1.1 2.1 3.1 4.1 5.1 6.1 7.1 8.1	Actor start use case. Systeem toont elementen. Actor kiest element. Systeem toont gegevens. Actor kiest niet meenemen (keep original). Systeem vraagt waar gelinkte elementen naar moeten verwijzen (binnen huidige import/export). Actor kiest referentie. Systeem veranderd verwijzingen.
Postconditie	Het geselecteerde element wordt niet meegenomen bij de import/export.	
Alternatiefscenario 1 (Bij stap 3.1, stap 5.1 en stap 7.1 van HS)	X.1	Actor annuleert.
Postconditie	Het geselecteerde element is niet veranderd.	
Alternatiefscenario 2 (Bij stap 5.1 van HS) – Actor kiest meenemen.	6.1 7.1 8.1	Systeem vraagt of oude referenties moeten worden teruggezet. Actor bevestigd. Systeem veranderd verwijzingen.
Postconditie	De originele staat van de verwijzingen naar het element is terug gezet. Het element wordt meegenomen in de import/export.	
Alternatiefscenario 3 (Bij AS2 stap 7.1) – Actor wil referenties niet terugzetten.	8.1	Systeem toont gegevens.
Postcondinitie	Het geselecteerde element is niet veranderd. Het element wordt meegenomen in de import/export.	


Start use case
Toon elementen
Kies element
Toon gegevens
Annuleer
Kiest meenemen
Kiest niet meenemen
Vraag of oude referenties moeten worden teruggezet.
Toon gegevens
Vraag waar gelinkte elementen naar moeten verwijzen
Verander verwijzingen
Kies nieuwe referentie
Verander element

12.3. Bijlage 3 – Technisch ontwerp



Project import/export

Technisch ontwerp

Student

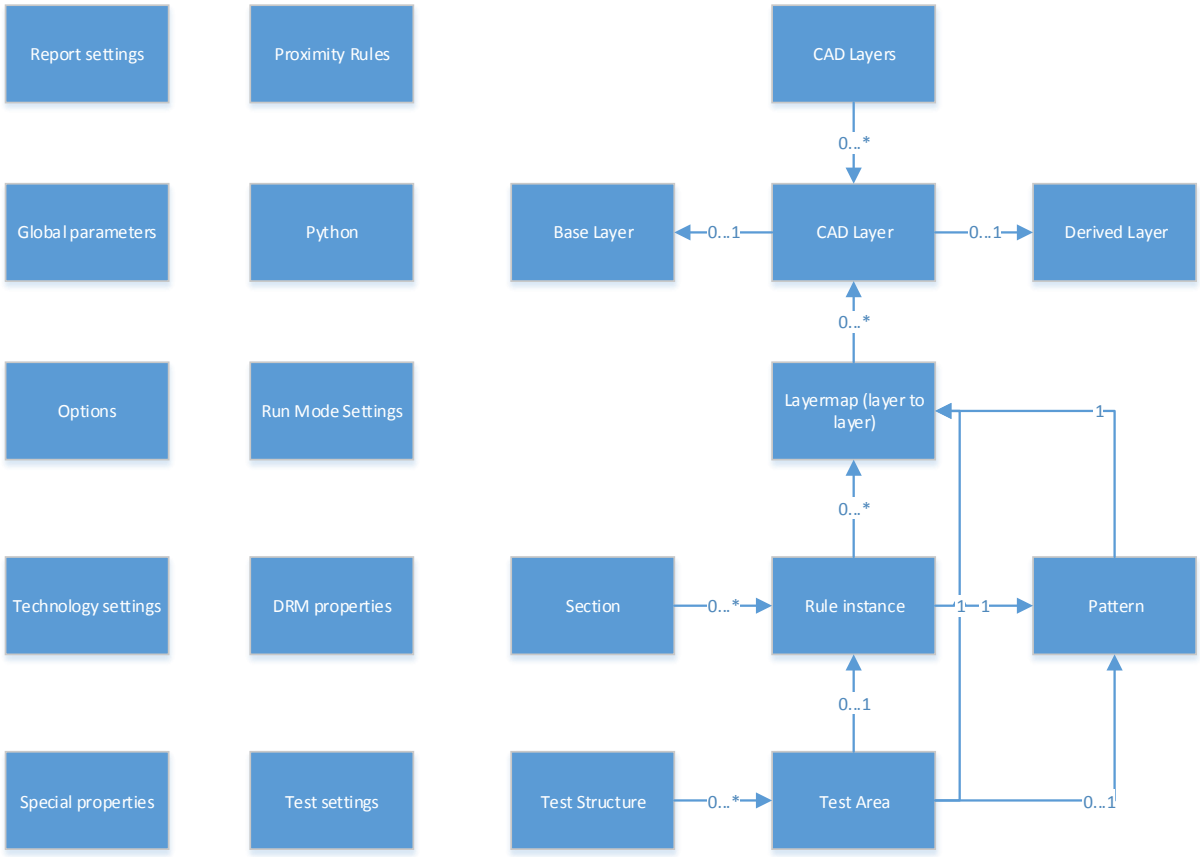
Naam:	Velleman
Voorletters:	F.M.V.
Roepnaam:	Floris
Studentnummer:	1602376
Docentbegeleider:	Alex Jongman

Bedrijf

Naam:	NP-Komplete technologies
Adres:	De Rozentuin 18
Postcode:	5611 SW
Plaats:	Eindhoven
Inhoudelijk bedrijfsbegeleider:	Maarten Berkens
Technisch bedrijfsbegeleider:	Rob Gelderblom

Technische analyse huidige structuur

In dit hoofdstuk wordt gekeken naar hoe de in het functioneel ontwerp gevonden groepen zich verhouden tot de daadwerkelijke implementatie in de source code. Dit wordt gedaan door naar het huidige import/export proces te kijken en hierbij de bijbehorende klassen en attributen te identificeren aan de hand van de in de file structuur gevonden groepen. Dit zal uiteindelijk een klassendiagram opleveren van de huidige structuur in het programma.



Vanuit dit diagram moet duidelijk worden welke relatie er bestaat met de code. Zodra dit verband duidelijk is kan er met deze begrippen worden gewerkt naar het uitwerken van de requirements.

Door het huidige proces in stappen te beschrijven kan een basis worden gebruikt voor de volgorde waarin de import/export moeten verlopen.

Om te achterhalen waar dit proces plaats vindt is er in de code gekeken (vanuit de import en export knop). Dit geeft vervolgens aan dat de main interface een data member heeft die de import en export functies heeft.

Import analyse

Binnen deze functie kan de volgende opdeling gemaakt worden:

Stap in import proces
Maak temporary directory aan
Kies temporary directory als huidige project
Import global parameters
Import pexhooks en drvhooks

Import layermap
Import persistent settings
Import properties
Import patterns
Import rules
Import groups
Import report settings
Import test structures
Import test areas
Import proximity rules
Bewaar en werk vanuit temporary directory totdat er een save wordt gedaan

Het is belangerijk om te zien dat niet alle groepen hierbij terugkomen in het originele diagram. Sommige zijn er zelfs uitgelaten en zullen mogelijk tot een verzameling behoren.

Huidige proces en elementen

Als de onderdelen in dit proces worden uitgezet tegen de groepen van het functioneel ontwerp kan de volgende verdeling worden gemaakt. Hierbij wordt de naam gebruikt die ook in de source code aan de specifieke elementen wordt toegekend.

Technische groep (naam in code)	Groep(en) uit originele diagram
Global parameters	Global parameters
Pexhooks en drvhooks	Python
Layermap	Layermap
	CADlayers
	CADLayer
	Derived layer
	Base layer
Persistent settings	Options
	DRM Properties
	Run mode settings
	Test settings
	Technology settings
Properties	Special properties
Patterns	Pattern
Rules	Rule instance
Groups	Section
Report settings	Report settings
Test structures	Test structure
Test areas	Test area
Proximity rules	Proximity rules

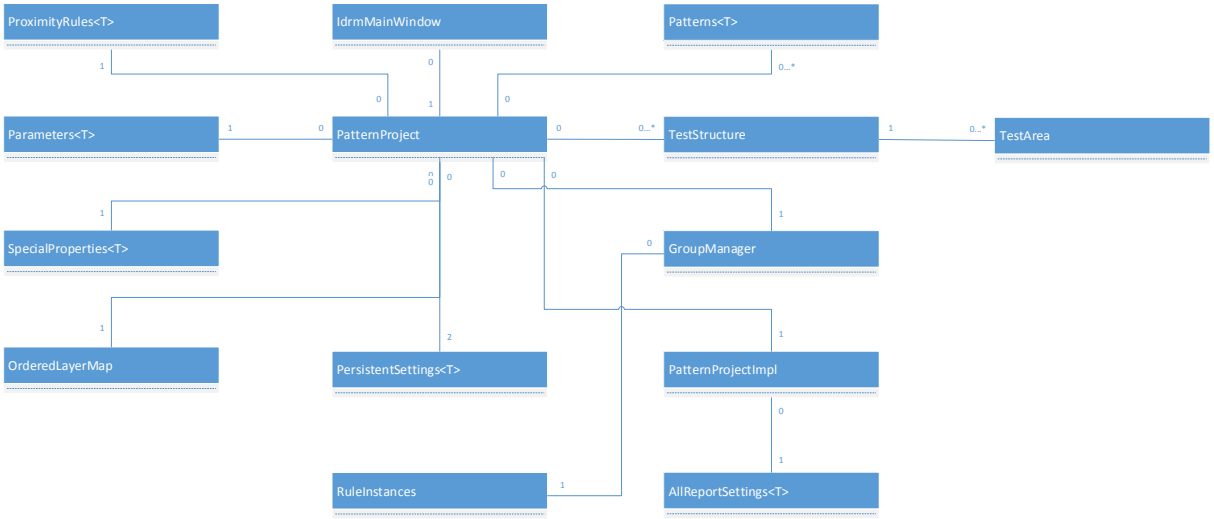
Op basis van deze opdeling is het mogelijk om een technische analyse te maken van de elementen die gebruikt worden en welke klassen hieraan vastzitten. Dit is gedaan door te kijken naar hoe deze groepen zich vertalen tot elementen in de code.

De volgende klassen worden in de code geassocieerd met deze groepen:

Technische groep	Naam klasse
Global parameters	Parameters

Pexhooks en drvhooks	-
Layermap	OrderedLayerMap
Persistent settings	PersistentSettings
Properties	SpecialProperties
Patterns	Pattern
Rules	RuleInstances
Groups	GroupManager
Report settings	AllReportSettings
Test structures	TestStructure
Test areas	TestArea
Proximity rules	ProximityRules

Door te kijken hoe deze elementen vastzitten aan de verschillende andere klassen en de hoofdinterface kan een klassendiagram worden gemaakt. Dit is gedaan door te kijken welke members de klasse (PatternProject) heeft.



Het is opvallend dat er veel klassen tussen zitten die in meervoud staan. Dit is bewust omdat dit meestal wrappers zijn voor een bepaald lijst type. In de ProximityRules klasse kunnen bijvoorbeeld meerdere instanties van ProximityRule voorkomen.

Door te bepalen welke comparison plaats vindt in deze sub elementen kan bepaald worden wat zal leiden tot een conflict.

Technische analyse conflicten

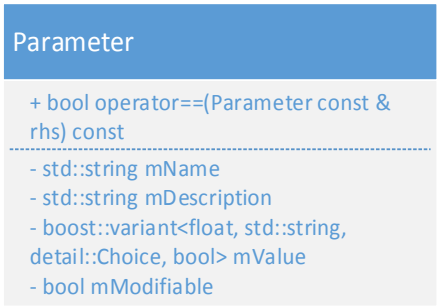
Met behulp van de gevonden klassen kan worden gekeken naar hoe deze worden vergeleken. Dit gebeurt in de huidige software door een evaluate functie en of een operator overload. Voor een aantal elementen gebeurt het helemaal niet of op een relatief vreemde plek.

Door te werken vanuit de eerder opgestelde tabel kunnen de daadwerkelijke onderliggende klassen worden opgezocht (waarin deze evaluaties worden gebruikt).

Technische groep	Naam klasse
Global parameters	Parameters
Pexhooks en drvhooks	-
Layermap	OrderedLayerMap
Persistent settings	PersistentSettings
Properties	SpecialProperties
Patterns	Pattern
Rules	RuleInstances
Groups	GroupManager
Report settings	AllReportSettings
Test structures	TestStructure
Test areas	Test
Proximity rules	ProximityRules

Parameters

Dit is eigenlijk geen klasse maar een type alias voor een lijst van Parameter instanties. Deze Parameter klasse heeft een aantal variabelen en maakt gebruik van een operator overload om vergelijkingen te maken met andere instanties. De aan dit project relevante onderdelen van de Parameter klasse worden in het onderstaande UML Klassen diagram beschreven:



Het gaat hier om een globale parameter binnen het programma. Dit betekent dat deze een aantal mogelijk types heeft:

- Number
- Text
- Choice
- Checkbox

De daadwerkelijke waarde hiervan wordt bewaard in mValue. Er kan met dit object ook worden bepaald van welke type mValue is. Gezien deze klasse geen relaties heeft met andere klasse kunnen er geen conflicten optreden rondom relaties tussen klassen.

In de comparison overload functie valt op te maken dat er conflicten kunnen ontstaan op basis van de volgende velden:

Veld	Conflict
mName	Naam is gelijk aan die van het andere object.
mDescription	Beschrijving is niet gelijk aan die van het andere object.
mModifiable	Veld is niet gelijk aan die van het andere object.
mValue	Type is niet gelijk aan die van het andere object.
	Waarde is niet gelijk aan die van het andere object.

Tijdens het importeren wordt voor elk parameter (in het import bestand) gekeken of er al een parameter is met dezelfde naam. Dit is dan ook het veld wat een conflict zal kunnen geven.

Indien een vergelijkbare wordt gevonden kan een foutmelding optreden indien een ander conflict zich ook voordoet (zie tabel). Als er dus een parameter gevonden is met dezelfde naam maar met een andere waarde ontstaat er een conflict en wordt dit momenteel afgehandeld door het niet te importeren en een foutmelding te tonen.

Als er geen conflicten optreden na het vinden van een parameter met dezelfde naam wordt er niets gedaan met de parameter uit het import bestand.

Pexhooks en Drvhooks

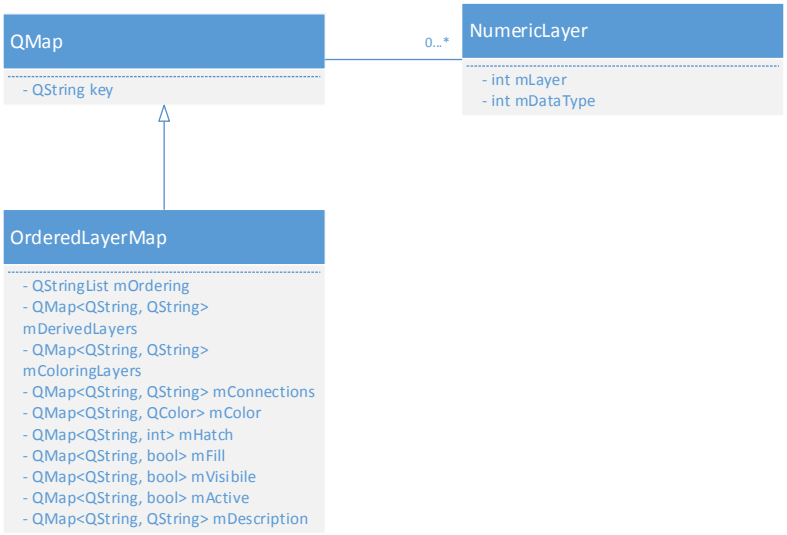
Dit zijn python bestanden waar geen daadwerkelijke klasse voor bestaat. Deze bestanden worden wel gebruikt in het programma maar zijn scripts die worden uitgevoerd voordat naar een patroon wordt gezogd. Er wordt in dit geval niet gekeken of de inhoud van deze scripts gelijk is aan een andere maar er wordt wel gekeken of het script al bestaat.

Indien een script al voorkomt in het huidige project wordt een foutmelding getoond. De overige scripts moeten door de gebruiker worden verplaatst. Er kan dus enkel een conflict optreden op basis van de naam van het bestand

Veld	Conflict
Naam	Bestand bestaat al.

OrderedLayerMap

Dit is een wrapper klasse voor alle mogelijke soorten layers. Deze worden echter gelinkt doormiddel van strings (op basis van naam). Deze klasse is gebasseerd op een map en gebruikt een onderliggende structure om deze te vullen. De aan dit project relevante onderdelen van de OrderedLayerMap klasse worden in het onderstaande UML Klassen diagram beschreven:



Hierbij geldt dat de Qmap een QString bevat als key (de naam van de layer). Als value heeft de map een NumericLayer structure. De member variabelen zijn in dit geval uitbereidingen op de huidige data structuur. De QStringList die gebruikt wordt is voor het bijhouden van de volgorde van de layers binnen de lijsten van het programma (omdat een QMap altijd gesorteerd is).

Hoewel er een aantal overloaded operators zijn binnen de klasse worden alle conflicten buiten de klasse afgehandeld. Toch gebeurt dit op basis van de velden uit de klasse en kan een tabel worden opgesteld waarin alle conflicten worden weergegeven:

Veld	Conflict
Name	Naam is gelijk aan die van het andere object.
NumericLayer	NumericLayer is gelijk aan die van het andere object.
Hatch	Hatch is niet gelijk aan die van het andere object.
Color	Color is niet gelijk aan die van het andere object.
Fill	Fill is niet gelijk aan die van het andere object.
Visibility	Visibility is niet gelijk aan die van het andere object.
Active	Active is niet gelijk aan die van het andere object.
Ordering	Ordering is niet gelijk aan de andere import map.
Description	Description is niet gelijk aan die van het andere object.
ColoringLayer	ColoringLayer is niet gelijk aan die van het andere object.
	Ander object is geen coloring layer.
ConnectionLayer	ConnectionLayer is niet gelijk aan die van het andere object.
	Ander object is geen connectivity layer.
DerivedLayer	DervivedLayer is niet gelijk aan die van het andere object.

	Ander object is geen derived layer.

Er wordt bij het importeren van deze klasse naar twee dingen gekeken. Zo moet de naam en het nummer van de layer (mLayer en mDataType samen) uniek zijn. Indien dit niet zo is wordt momenteel enkel gekeken welke type layer er hier gebruikt moet worden. Er is wel een optie om te kunnen kiezen tussen de namen en nummers. Het is mogelijk dat een nummer leeg is.

Omdat het niet mogelijk is om andere velden te negeren zijn er twee mogelijke conversies voor de andere velden:

- NumericLayer naar DerivedLayer
- DerivedLayer naar NumericLayer

Bij de derived layer gaat de value over een operatie die de layer moet controleren. Hoewel deze als string niet letterlijk hetzelfde zouden hoeven te zijn kunnen ze toch gelijk zijn (AND(a,b) en AND(b,a)).

Elementen uit deze lijst worden gebruikt door een hoop verschillende projectelementen. Dit zorgt ervoor dat er de mogelijkheid zou moeten zijn om te bepalen waar deze naartoe verwijzen indien een layer niet wordt meegenomen.

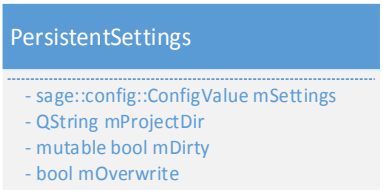
Dit komt ook voor als een layer zou worden overschreven. Aangezien de naam kan worden overschreven is het van belang de projectelementen in het huidige project ook aan te passen.

PersistentSettings

Deze klasse zal enkel geïmporteerd worden als het project leeg is. Dit zorgt er voor dat er nooit een conflict kan zijn omdat alle gegevens automatisch worden overschreven.

Bij het exporteren wordt de data echter wel meegenomen zodat het beschikbaar is mocht er een import worden gedaan in een nieuw project.

De PersistentSettings klasse ziet er als volgt uit:



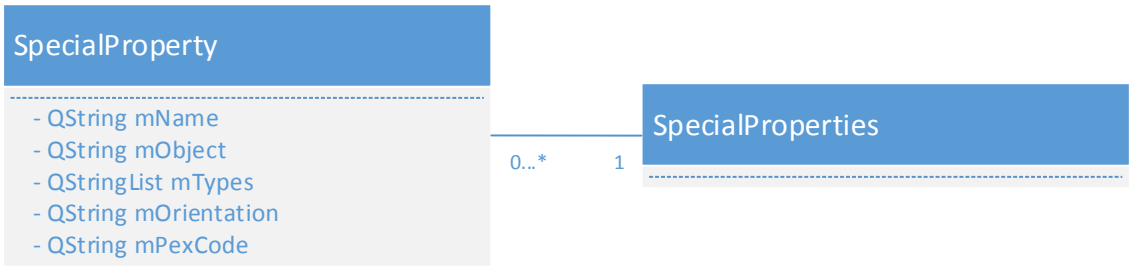
Gezien deze klasse verder nergens aan vast zit is er geen mogelijkheid om een conflict te veroorzaken met dit object. Indien wordt besloten deze ook te importeren als het project niet leeg is kan een conflict ontstaan op basis van de naam van de settings die worden bijgehouden.

Veld	Conflict
mSettings[index].mName	Naam bestaat al in huidige project.

SpecialProperties

Deze klasse is een wrapper voor een lijst van SpecialProperty instanties. Er is geen functie in de SpecialProperties klasse waarmee expliciet wordt gecontroleerd of er zich een conflict voordoet. Binnen de insert functie wordt echter gekeken of een SpecialProperty instantie een bestaande naam heeft. Indien dit zo is wordt de oude SpecialProperty instantie overschreven door de nieuwe.

De aan dit project relevante onderdelen van de SpecialProperties klasse worden in het onderstaande UML Klassendiagram beschreven:



De conflicten zitten voornamelijk in de data van de SpecialProperty klasse gezien deze geen verdere bindingen heeft. De volgende tabel toont de conflicten die mogelijk zijn:

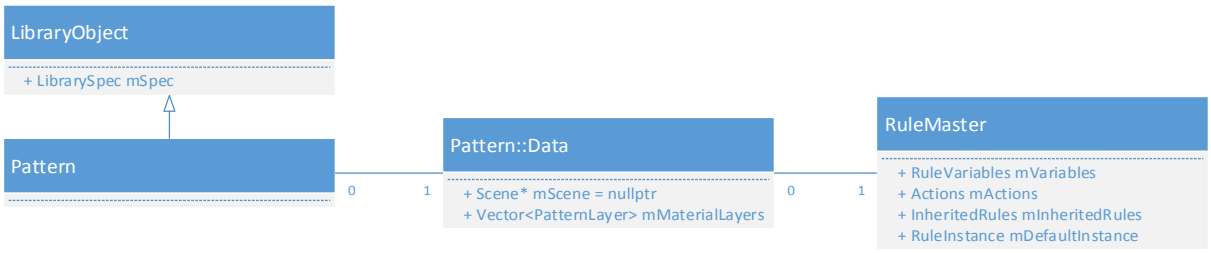
Veld	Conflict
Name	Naam is gelijk aan die van het andere object.
Object	Object is niet gelijk aan die van het andere object.
Types	Types zijn niet gelijk aan die van het andere object.
Orientation	Orientation is niet gelijk aan die van het andere object.
PexCode	PexCode is niet gelijk aan die van het andere object.

Het niet meenemen van een SpecialProperty heeft geen effect op andere klassen en kan zo geen conflict veroorzaken.

Pattern

In tegenstelling tot de voorgaande klassen is dit geen wrapper maar is er in PatternProject een lijst van instanties van deze klasse opgenomen. Bij het importeren wordt deze klasse maar weinig gebruikt omdat vrijwel alles dat hier mee te maken heeft met een parser gebeurt. Er is wel een equivalent functie kijkt of de twee patterns uiteindelijk gelijk zijn aan elkaar.

In de code zit alle informatie over het pattern echter in een structure die als member is opgenomen. De aan dit project relevante onderdelen van de Pattern klasse worden in het onderstaande UML Klassendiagram beschreven:



Een Scene is in dit geval de tekeningen die worden gemaakt bij een pattern. De RuleMaster klasse heeft voornamelijk globale informatie over het pattern. De mSpec variabele in LibraryObject bewaard de library en pattern naam. Een PatternLayer is een verwijzing naar een layer.

In de import methode valt op te maken dat er ook nog optionele attachments zijn voor elk pattern (bestanden).

Er zijn een hoop conflicten die zich kunnen voordoen bij het importeren van een pattern. Momenteel wordt een pattern hernoemt zodra de naam niet uniek is en overschrijft deze de huidige. Dit geeft echter een segmentation fault.

Op basis van de velden en afhankelijkheid binnen het programma kan echter wel een tabel worden opgesteld met de mogelijke conflicten die zich kunnen voordoen.

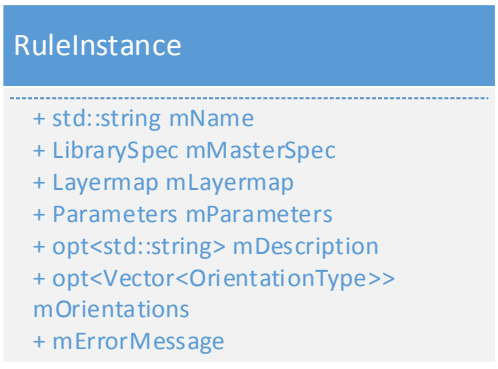
Veld	Conflict
Name	Name is gelijk aan die van het andere object.
MaterialLayers	Material layers zijn niet gelijk aan die van het andere object.
Scene	Scene is niet gelijk aan die van het andere object.
RuleVariables	RuleVariables zijn niet gelijk aan die van het andere object.
Actions	Actions zijn niet gelijk aan die van het andere object.
InheritedRules	InheritedRules zijn niet gelijk aan die van het andere object.
RuleInstance	RuleInstance (default) is niet gelijk aan die van het andere object.
Attachments	Attachments zijn niet gelijk aan die van het andere object.
Images	Images zijn niet gelijk aan die van het andere object.

Het vergelijken van de afbeelding en attachments gebeurt op basis van een compare van de file namen. Dit zou mogelijk kunnen worden gedaan door te vergelijken op basis van de inhoud.

Een pattern kan gebruikt worden door een rule instance en/of een test area wat inhoud dat er een mogelijkheid moet zijn om aan te geven waar deze naar verwijzen indien een pattern niet wordt meegenomen.

RuleInstance

De RuleInstance klasse is een data structure. De bovenliggende lijst voert de controle voor conflicten uit en kijkt hierbij of de naam gelijk is aan die van een bestaand element. Er is net zoals bij een aantal andere klassen de optie om te kiezen of de originele of de nieuwe moet worden gebruikt. Zodra er een conflict is met de naam wordt gekeken of de andere onderdelen van een rule instance voor een conflict zorgen.



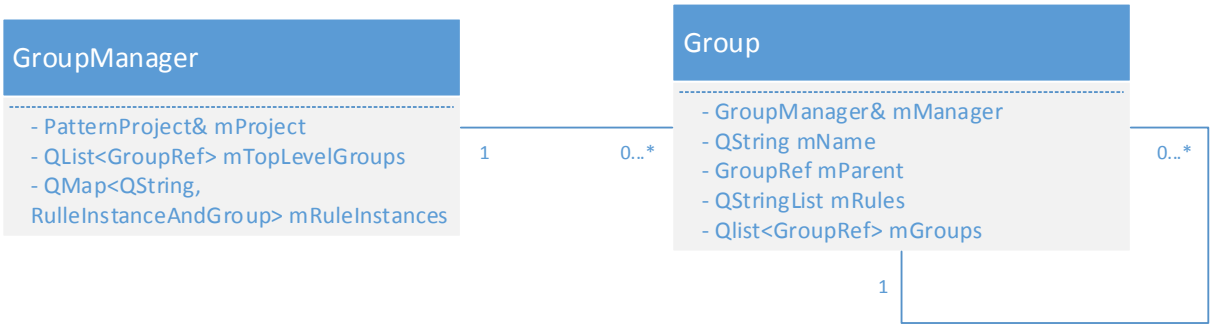
Het is van belang om in te zien dat hier gebruik wordt gemaakt van een layermap per rule instance. Deze layermap is echter een type alias van een Map<string, string> die verwijst naar een layer door middel van naam.

Veld	Conflict
Name	Name is gelijk aan die van het andere object.
MasterSpec	MasterSpec is niet gelijk aan die van het andere object.
LayerMap	LayerMap is niet gelijk aan die van het andere object.
Parameters	Parameters is niet gelijk aan die van het andere object.
Description	Description is niet gelijk aan die van het andere object.
Orientations	Orientations is niet gelijk aan die van het andere object.
ErrorMessage	ErrorMessage is niet gelijk aan die van het andere object.

Deze klasse wordt gebruikt door test area en zal als deze niet wordt meegenomen moeten verwijzen naar een andere instantie voor de test area (gezien deze niet naar niks kan verwijzen).

GroupManager

Een GroupManager is een klasse die referenties naar rule instances bewaart. Ook wordt door deze klasse een lijst van Group instanties bijgehouden. Deze group instanties verwijzen naar de daadwerkelijke instanties van de RuleInstance klasse die worden bijgehouden in de GroupManager klasse. Dit gebeurt door middel van een lijst met strings in de Group klasse.



De GroupRef die hierbij wordt gebruikt is een type alias voor een scoped pointer naar een group instance. Momenteel worden conflicten met de group klasse niet behandeld en geeft elk conflict een segmentation fault.

Deze rule instances zijn opgedeeld in groepen en deze groepen hebben een naam. Conflicten kunnen voorkomen als de groepen dezelfde namen hebben. Indien dit zo is wordt gekeken of ze ook dezelfde elementen hebben met dezelfde waarden. Als dit zo is wordt er momenteel niet geïmporteerd.

Veld	Conflict
Name	Name is gelijk aan die van het andere object.
Structure	Structuur van de groepen is niet gelijk aan die van het andere project.

TopLevelGroups	TopLevelGrouups zijn niet gelijk aan die van het andere object.
RuleInstances	RuleInstances zijn niet gelijk aan die van het andere object.
Manager	GroupManager is niet gelijk aan die van het andere object.
Parent	Parent van een group is niet gelijk aan die van het andere object.
Rules (referenties)	Rules zijn niet gelijk aan die van het andere object
Groups	Onderliggende groepen zijn niet gelijk aan die van het andere object.

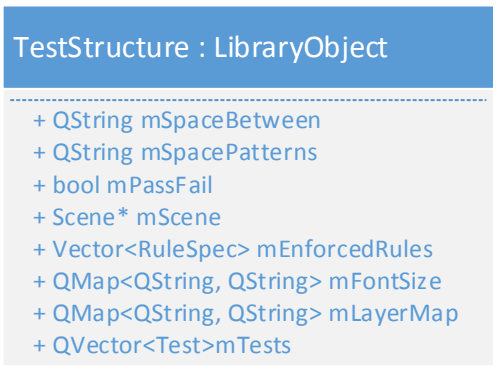
Gezien deze groepen direct verbonden zijn met rule instances en geen andere toepassing vinden in het programma is het logisch om deze in de import van rule instances op te nemen. Er kan echter per group enkel een conflict optreden als deze in hetzelfde pad ligt als een andere group.

ReportSettings

De ReportSettings klasse is vrijwel identiek aan de special properties klasse. Er wordt de exact zelfde structuur aangehouden maar de data wordt op een ander punt in het programma gebruikt en heeft een andere naam. Voor de conflicten die hierdoor kunnen optreden kan worden gekeken bij special properties. Er wordt enkel geïmporteerd als het project helemaal leeg is.

TestStructure

In deze klasse wordt een lijst bijgehouden van alle tests in deze specifieke groep. Deze klasse heeft een naam wat een conflict kan geven. Deze naam zit in het LibraryObject waarop het gebasseerd is.



In de huidige import wordt indien er een conflict is gevraagd welke voorkeur heeft (nieuwe of oude) en wordt op basis van de keuze geïmporteerd. Een daadwerkelijke compare wordt gedaan in de filereader klasse aangezien teststructure zelf geen vergelijking functies heeft.

Conflicten kunnen optreden op basis van een vergelijkbare naam maar andere inhoud (dit is met uitzondering van de tests die erin zitten).

Veld	Conflict
Name	Name is gelijk aan die van het andere object.
SpaceBetween	SpaceBetween van de teststructure is niet gelijk aan die van het andere project.
SpacePatterns	SpacePatterns van de teststructure is niet gelijk aan die van het andere project.

PassFail	PassFail van de teststructure is niet gelijk aan die van het andere project.
Scene	Scene van de teststructure is niet gelijk aan die van het andere project.
EnforcedRules	De EnforcedRules van de teststructure zijn niet gelijk aan die van het andere project.
FontSize	FontSize van de teststructure is niet gelijk aan die van het andere project.
LayerMap	LayerMap van de teststructure is niet gelijk aan die van het andere project.

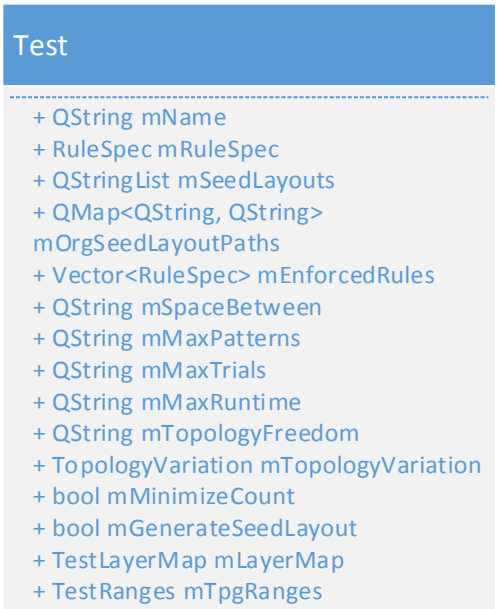
Aangezien er geen afhankelijkheden vastzitten aan teststructure kan dit element zonder problemen worden weggelaten (uitgaande dat de tests dan ook niet worden geïmporteerd).

Test

In de klasse test zelf wordt ook geen mogelijkheid gegeven voor het vergelijken van objecten maar en kan in de interface worden opgemaakt dat er voornamelijk wordt gekeken naar welke namen specifieke testen hebben. Indien deze hetzelfde zijn wordt er gevraagd welke voorkeur heeft.

Er is echter wel een functie om te kijken of een test gelijk is aan een andere test. Dit gebeurt niet als de inhoud hetzelfde het is (in welk geval er momenteel gewoon niet wordt geïmporteerd).

Deze functie is er wel buiten de klasse in de vorm van een comparison operator overload (bool operator==(Test const & lhs, Test const & rhs)).



Conflicten kunnen optreden op basis van een vergelijkbare naam maar andere inhoud.

Veld	Conflict
Name	Name is gelijk aan die van het andere object.
RuleSpec	RuleSpec van de test is niet gelijk aan die van het andere project.
SeedLayouts	SeedLayouts van de test is niet gelijk aan die van het andere project.

OrgSeedLayoutPaths	OrgSeedLayoutPaths van de test is niet gelijk aan die van het andere project.
EnforcedRules	EnforcedRules van de test is niet gelijk aan die van het andere project.
SpaceBetween	SpaceBetween van de test is niet gelijk aan die van het andere project.
MaxPatterns	MaxPatterns van de test is niet gelijk aan die van het andere project.
MaxTrials	MaxTrials van de test is niet gelijk aan die van het andere project.
MaxRuntime	MaxRuntime van de test is niet gelijk aan die van het andere project.
TopologyFreedom	TopologyFreedom van de test is niet gelijk aan die van het andere project.
TopologyVariation	TopologyVariation van de test is niet gelijk aan die van het andere project.
MinimizeCount	MinimizeCount van de test is niet gelijk aan die van het andere project.
GenerateSeedLayout	GenerateSeedLayout van de test is niet gelijk aan die van het andere project.
LayerMap	LayerMap van de test is niet gelijk aan die van het andere project.
TpgRanges	De TpgRanges van de test zijn niet gelijk aan die van het andere project.

De tests zitten vast aan de test structures en moeten uit deze lijst verwijderd worden indien ze niet worden geïmporteerd. Gezien deze erg nauw zijn verbonden is het een mogelijkheid om gedurende de import de test structures te combineren met de tests.

ProximityRules

In deze klasse wordt gebruik gemaakt van een structure die de daadwerkelijk proximity rule bevat. De proximity rules worden momenteel alleen bij het laden en opslaan van een project verwerkt. Dit zorgt ervoor dat bij het exporteren wel een file wordt gemaakt waarin deze settings staan maar dat er nog geen mogelijkheid is tot importen. Dit komt doordat export gewoon de save functie aanroept op een temporary directory.

Er zit in deze klasse nog geen functie waardoor proximity rules kunnen worden geïmporteerd. Ook ontbreekt een functie waarmee vergelijking kunnen worden gedaan. Dit houdt in dat indien deze groep wordt toegevoegd aan de import/export wizard er een dergelijke functie gemaakt zal moeten worden.

Om te kijken of een proximity rule gelijk is aan een andere kan worden gekeken naar de naam. Dit zal voor conflicten zorgen in de interface als er meerdere van zijn. Om te kijken of er niet dezelfde waarde in zit kan worden gecontroleerd op de volgende velden:

- Name
- Type
- Settings
- Visibility

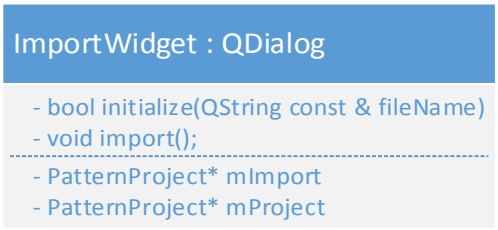
Een proximity rule heeft geen relatie met andere projectelementen.

Klassendiagram

Op basis van de gevonden architectuur en de wensen van de gebruikers kan worden gesteld dat er behoefte is aan een interface voor het importeren van projectelementen. Deze projectelementen moeten bereikbaar zijn vanuit deze interface. Om ervoor te zorgen dat er niet nog een godklasse ontstaat kunnen instanties worden gebruikt van PatternProject (project klasse).

Dit zorgt ervoor dat er eenvoudig onderscheid kan worden gemaakt tussen het import project en het huidige project. Er is echter tijdens de analyse duidelijk geworden dat de huidige code in deze klasse referenties bevat naar de interface die niet beschikbaar zullen zijn voor de import data. Dit heeft dan ook als nadeel dat er sommige stukken uit de bestaande code zullen moeten worden veranderd (soms misschien zelfs redundant).

Hoewel dit ervoor kan zorgen dat er meer code bij PatternProject zal komen zorgt het er ook voor dat er een correcte opdeling is als wordt uitgegaan van het MVC pattern.



Er wordt hier gebruik gemaakt van pointers om te voorkomen dat de gigantische hoeveelheid data in de klassen wordt gekopieërd. Er kan in de daadwerkelijke implementatie echter beter gebruik worden gemaakt van smart pointers om ervoor te zorgen dat er geen memory leaks zijn.

Design

Voor de import/export wizard is het van belang dat er gebruik wordt gemaakt van een gebruikersvriendelijke interface. Om dit te berijken wordt iteratief verbeteringen aangebracht aan de gebruikersinterface.

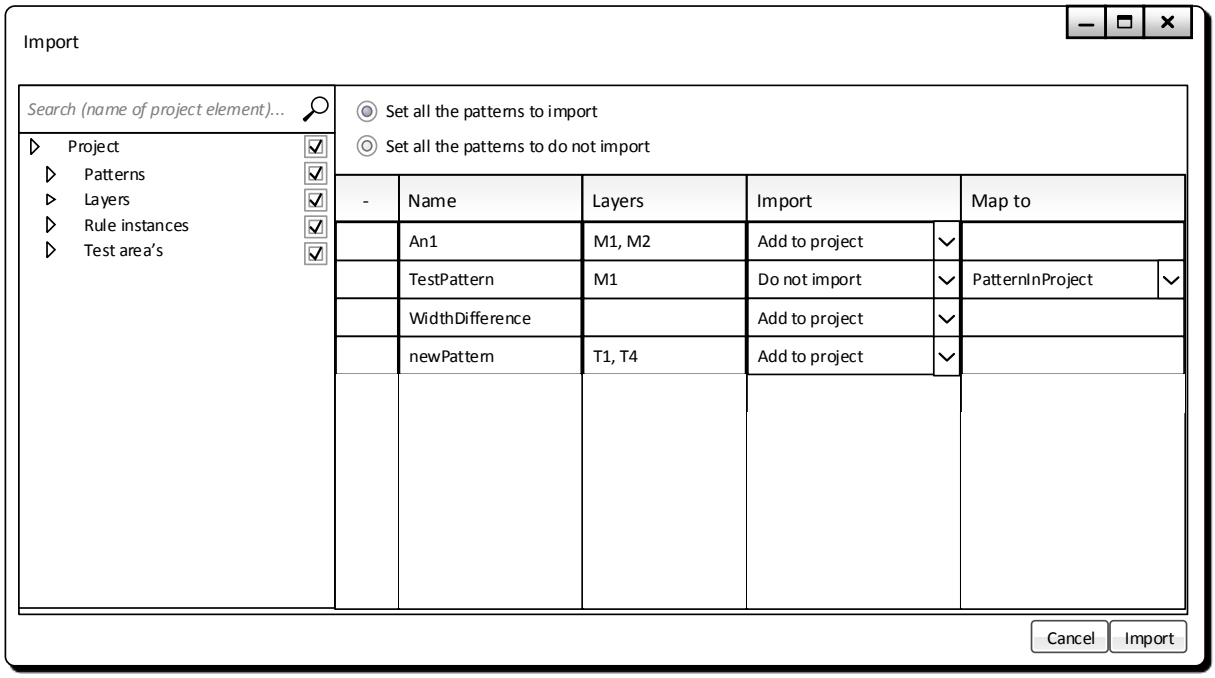
Omdat het hier gaat om een vrij technisch programma en de doelgroep iets flexibeler is op technisch gebied zal de mate van bruikbaarheid mogelijk lager uitvallen dan een product dat is gericht op de gemiddelde computer gebruiker.

Er staan in dit hoofdstuk meerdere designs die zijn voorgekomen uit feedback van de gebruikers. Dit is vervolgens teruggekoppeld en opgenomen als een hogere versie.

Versie 1

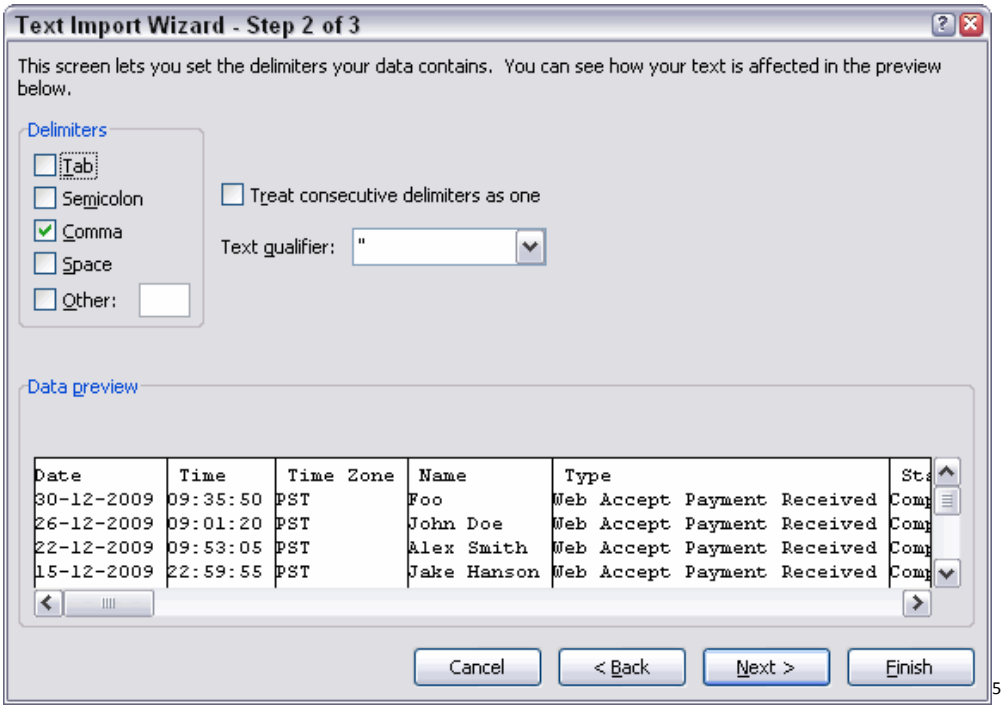
Tijdens het maken van deze versie is vooral vastgehouden aan literatuur en overeenkomsten in import widgets. Door vanuit dit punt te werken is er een design wat een goede onderbouwing heeft.

Het eerste design is:

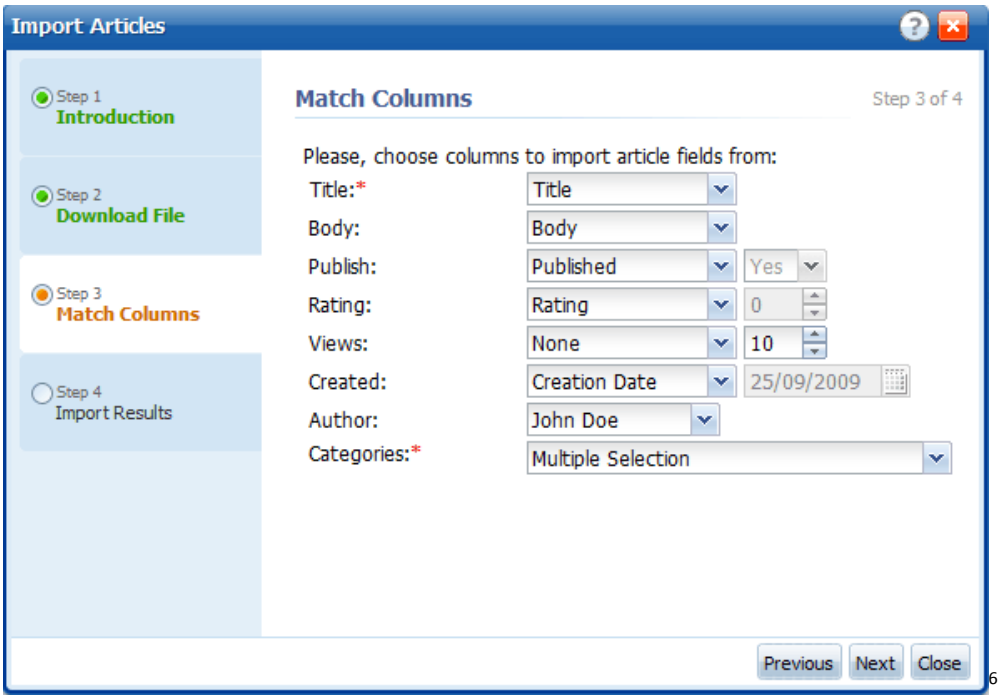


Hierbij zijn in de tree view aan de linker kant van het scherm de specifieke projectelementen opgenomen als sub onderdelen van de bijbehorende header. Zodra hier op geklikt wordt zal een scherm worden getoont dat alle informatie bevat rondom dit element. Dit zorgt ervoor dat alle data kan worden ingezien vanuit het import project.

Voor inspiratie is gekeken naar al bestaande import widgets en de overeenkomsten hierin. De volgende import widgets zijn gebruikt om tot een basis te komen voor het design van deze interface.

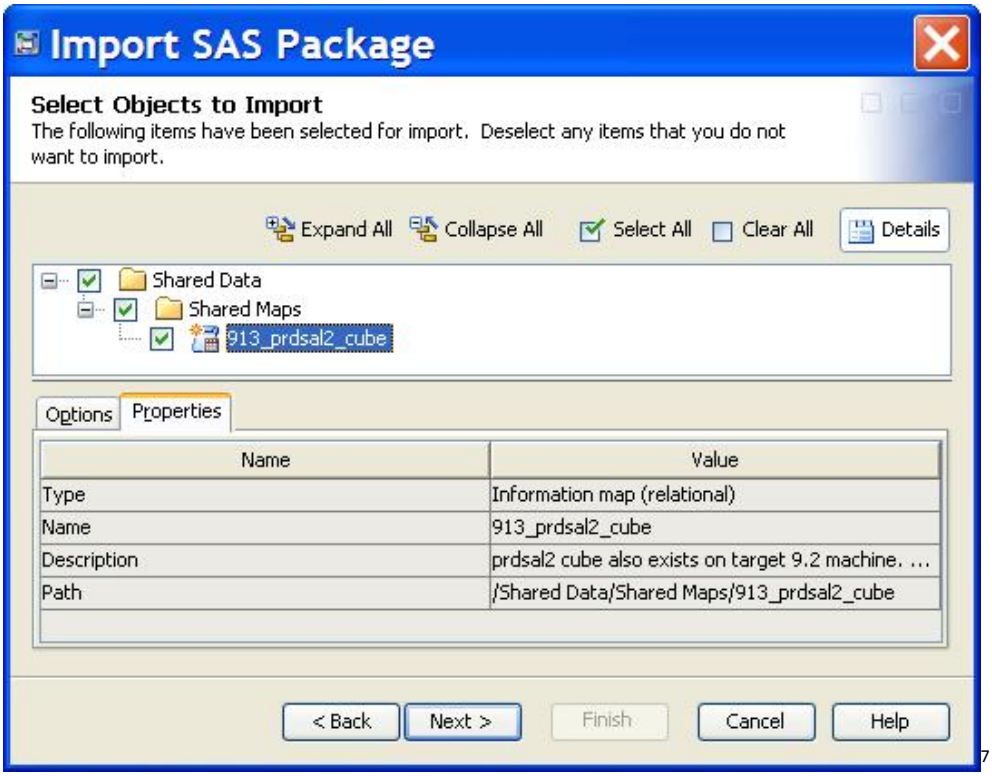


[5]

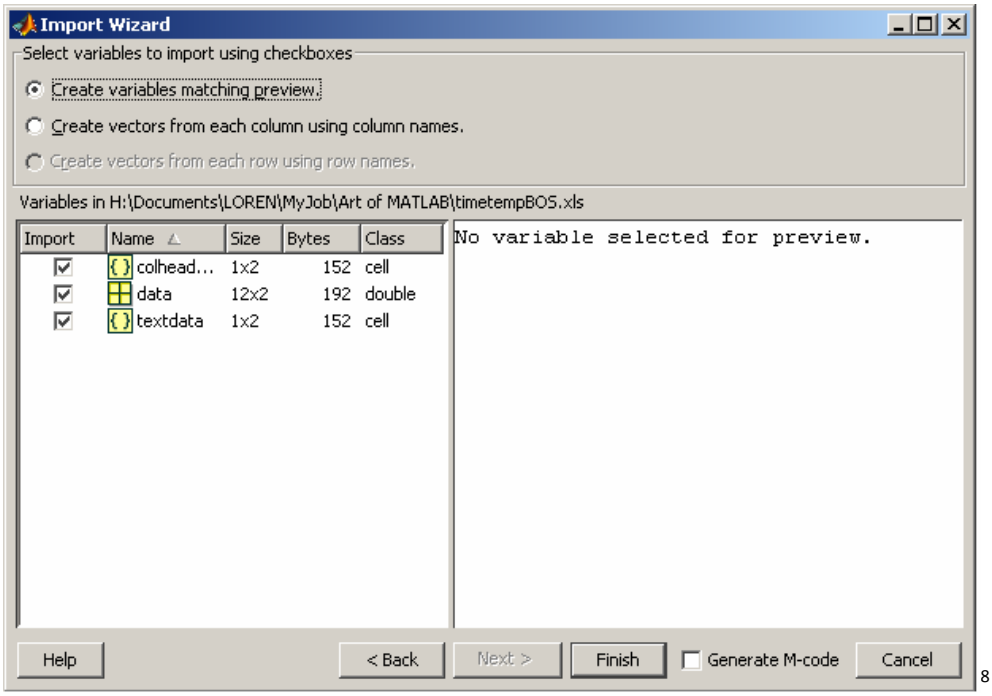


[6]

[5] http://www.jkp-ads.com/images/importtext06.gif
[6] http://www.web-site-scripts.com/images/kmp/feature-tour/screenshots/articles-import.png

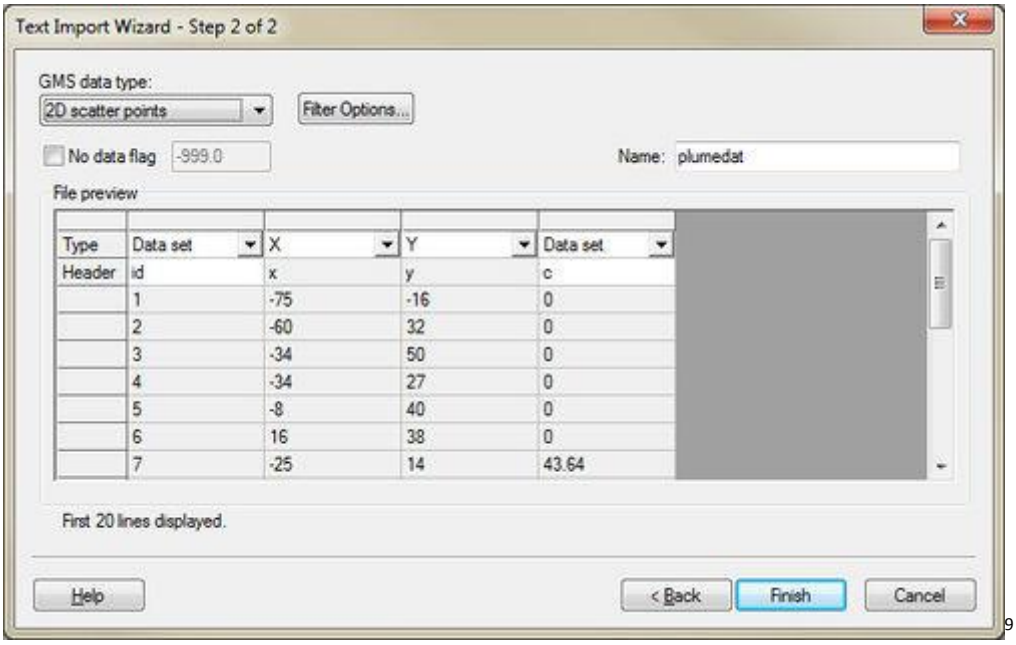


[7]



[8]

[7] http://support.sas.com/kb/35/addl/fusion_35212_1_import_wizard_icon.jpg
[8] http://blogs.mathworks.com/images/loren/57/uiimport1.png



[9]

Door te kiezen voor vijf verschillende import widgets kunnen zekere overeenkomsten worden gevonden en kan minimaal een basis worden bepaald. De overeenkomsten die opvielen zijn als volgt:

- Tabel weergave voor data
- Subgroup selectie mogelijkheid (combo box, treeview)
- Optie tot annuleren van proces
- Optie voor importeren
- Optie voor het zien van alle data

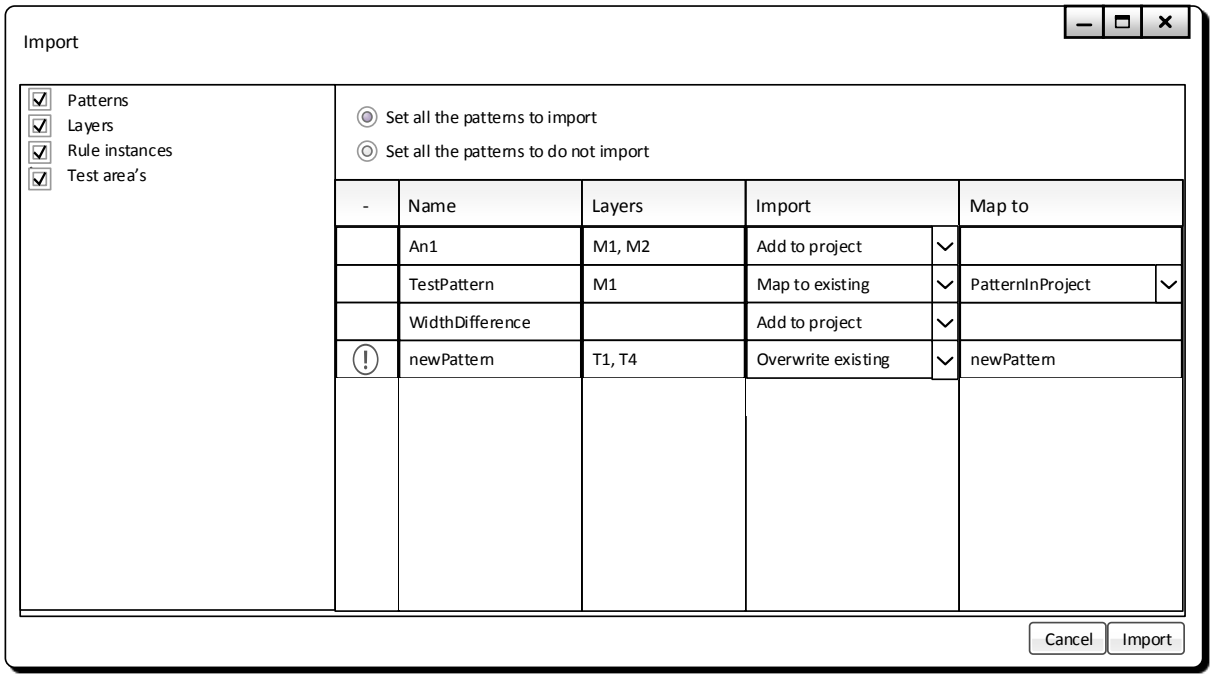
Met deze standpunten is gewerkt aan het initiële design. Er is gekozen voor een treeview omdat dit een vergelijkbare weergave geeft van het project zoals het programma dat al doet. Gezien er de mogelijkheid is om veel projectelementen te hebben zijn er knoppen toegevoegd waarmee snel alle elementen van een soort wel of niet kunnen worden geïmporteerd. Dit sluit aan op de use cases die zijn opgesteld door de gebruikers.

Het zoeken naar projectelementen zorgt ervoor dat een gebruiker snel data kan vinden over een bepaald object.

Versie 2

Op basis van de feedback en testen is gewerkt aan een verbeterd design. Het verbeterde design na iteratie 1 ziet er als volgt uit:

[9] http://www.xmswiki.com/w/images/thumb/7/70/ImportWizardStep2.jpg/500px-ImportWizardStep2.jpg

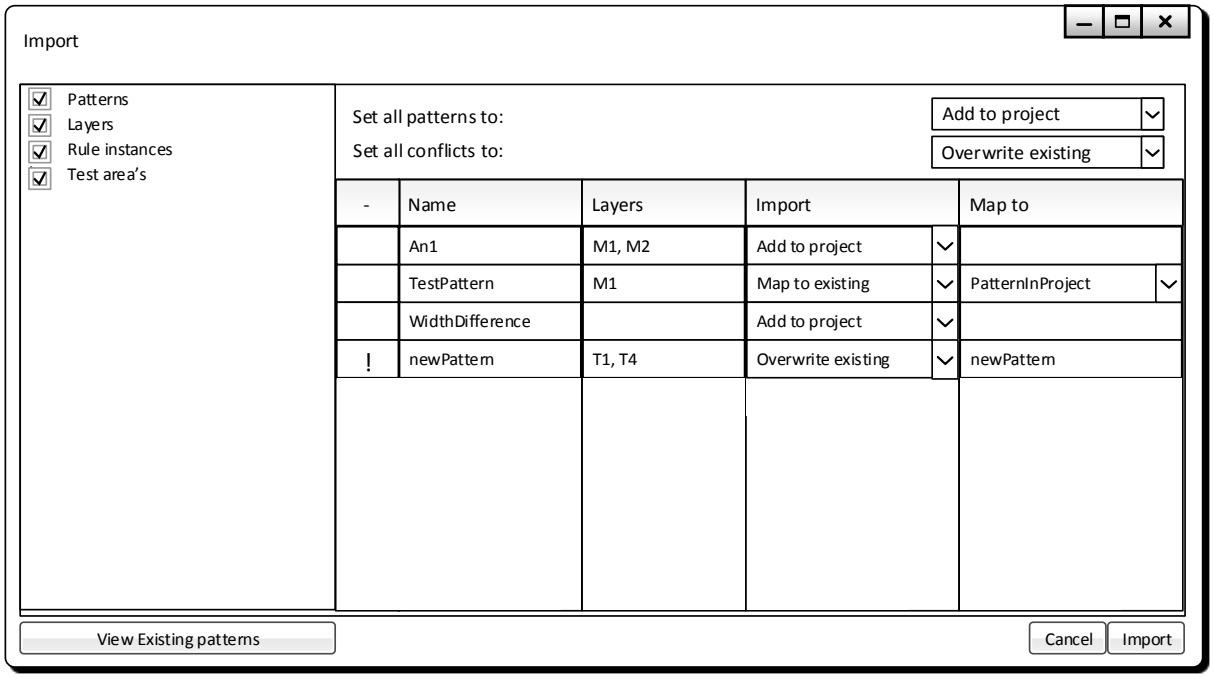


Hierbij hebben de gebruikers aangegeven dat een subgroep en zoeken niet handig is en dat ze liever in de tabel de data kunnen zien. Aangezien dat hier niet past is er een dubbelklik optie op een rij toegevoegd die ervoor zorgt dat het geselecteerde element en alle bijbehorende data wordt getoont.

Ook is een icoontje opgenomen om aan te geven of er een conflict is met een element uit het al bestaande project.

Versie 3

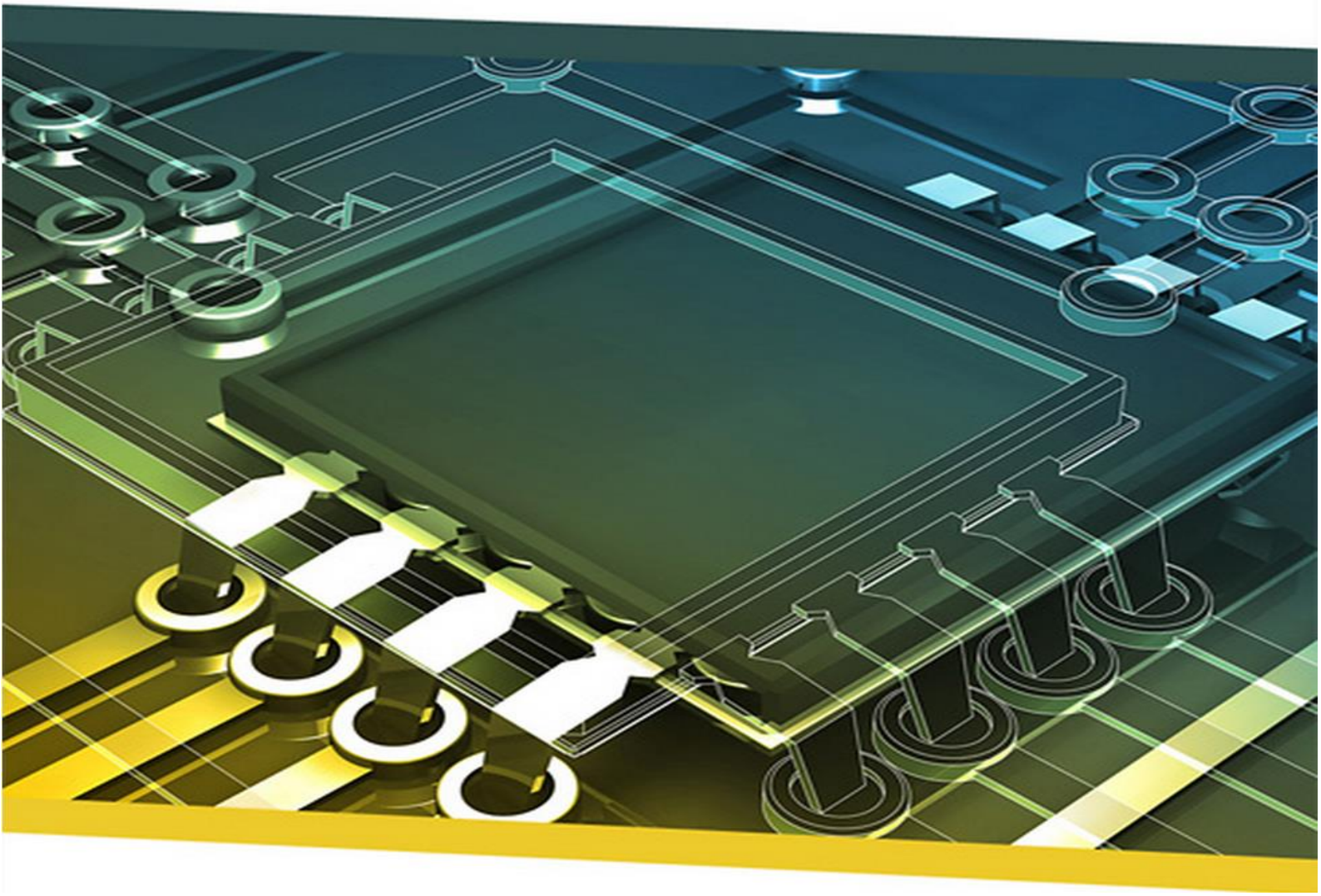
Na de tweede iteratie is nogmaals een test gedaan om te zien welke mogelijke verbeteringen er kunnen worden gemaakt aan de interface. Op basis van de feedback en testresultaten is het volgende design gemaakt:



Ten opzichte van de voorgaande versie zijn er weinig grote veranderingen. Er is aangegeven dat de radio buttons die gebruikt werden niet handig waren en dat het geen goede opties gaf voor conflicten. Daarom is besloten om snelle opties op te nemen voor specifiek geen of wel een conflict. Dit gebeurt nu op basis van een combobox.

Ook is gevraagd om meer functionaliteit. Zo vinden gebruikers het handig als er een optie is om de huidige projectelementen te kunnen bekijken. Dit is opgenomen in de vorm van een knop linksonderin die een tabel weergave geeft met informatie over de bestaande objecten van dat type.

12.4. Bijlage 4 - Testplan



Project import/export binnen iDRM

Testplan

Student

Naam:	Velleman
Voorletters:	F.M.V.
Roepnaam:	Floris
Studentnummer:	1602376
Docentbegeleider:	Alex Jongman

Bedrijf

Naam:	NP-Komplete Technologies
Adres:	De Rozentuin 18
Postcode:	5611 SW
Plaats:	Eindhoven
Inhoudelijk bedrijfsbegeleider:	Maarten Berkens
Technisch bedrijfsbegeleider:	Rob Gelderblom

Business use cases

Om te bepalen of de functionaliteit goed werkt kan een groot aantal test methodieken worden gebruikt. Er is besloten om de use cases die vanuit de gebruikers komen te testen in de interface. De use cases die hieronder vallen zijn als volgt (uit het functioneel ontwerp):

Project	Actie
Leeg project	Import bestaand project.
Leeg project	Hergebruik elementen uit ander project.
Bestaand project	Verbeter op basis van ander project.
Bestaand project	Alles importeren dat geen conflicten geeft.
Bestaand project	Alles hernoemen dat conflicten geeft.
Bestaand project	Kies welke elementen (als elementen samenhangen dan worden die andere ook meegenomen).
Bestaand project	Importeren van alle patterns (zonder de layers die moeten op iets anders voor patterns).

Per use case kan een testcase worden opgesteld. Op basis van de resultaten van deze testcases kan worden bepaald of conflicten worden opgelost/verminderd door het gebruik van de import widget.

Gezien het zesde onderdeel automatisch ingesteld wordt is deze niet opgenomen als test case.

Voor het testen zelf wordt gebruik gemaakt van een interface testing tool die binnen het bedrijf onderhouden wordt. Dit programma heet testify en biedt macro achtige functionaliteit voor het afspelen van scenario’s.

Testcase 1

Test case #: 1	Test case naam: Import bestaand project
Systeem: iDRM	Datum: 11-05-2015
Auteur: Floris Velleman	Beschrijving: Importeren van projectelementen in een leeg project.

Pre-conditie De gebruiker heeft het programma gestart. Het programma toont een leeg project.

Stap	Actie	Verwachte reactie	Pass/ Fail	Commentaar
1	Kies optie import project.	Systeem toont optie voor het kiezen van import bestand.		
2	Gebruiker kiest import bestand (Test data).	Systeem toont import widget met data.		
3	Gebruiker drukt op ok.	Systeem importeerd data en sluit scherm.		
4	Check post conditie 1	Correct/Fout		

Post-conditie 1. Alle data is opgenomen in het project.

Test data

De volgende test data zal in het import bestand aanwezig moeten zijn. Zodra er is geïmporteerd en er wordt opgeslagen moet de inhoud van de project folder deze elementen bevatten.

Parameters

Name	Description	Value	mModifiable
GP1	The first param	20 (Number)	True
GP2		Text (Text)	False
GP3	The third param	0 : { Yes, No } (Choice)	True
GP4		Enabled (Checkbox)	False

Python hooks

Name
drv_hooks.py
pex_hooks.docx

Layers

Name	Number	Type	Color : Hatch	Fill	Description
L1	1:0		255, 0, 255 : 1	True	Normal L1
L2	2:0		0, 0, 0 : 0	False	
L3		Derived	255, 0, 255 : 1	True	Derived L3
L4		Derived	0, 0, 0 : 0	False	
L5		Coloring	255, 0, 255 : 1	True	Coloring L5
L6		Coloring	0, 0, 0 : 0	False	
L7		Connection	255, 0, 255 : 1	True	Connection L7
L8		Connection	0, 0, 0 : 0	False	

Patterns

Name	MaterialLayers	Scene	Variables	Attachments	Description
P1	None	Square	None	None	P1
P2	L1	None	A : 7	File.txt	
P3	L1, L2	Square	None	None	P3
P4	L3	None	B : GP1	File.txt	

Rule instances

Name	Master	Parameters	Layermap	Path (groups)
RI1	P1	GP1	L1	Test/Test1
RI2	P2	GP2	L1	Test/
RI3	P3	GP3	None	Test/Test2
RI4	P4	GP4	L3	Test/Test3

Tests

Name	Spec	Layermap	Minimize count	Parent
T1	RI1	L1	True	S1
T2	RI2	L2	False	S2
T3	P1	L3	True	S1
T4	P2	L4	False	S2

Testcase 2

Test case #: 2	Test case naam: Import deel bestaand project
Systeem: iDRM	Datum: 11-05-2015
Auteur: Floris Velleman	Beschrijving: Importeren van projectelementen in een leeg project.

Pre-conditie De gebruiker heeft het programma gestart. Het programma toont een leeg project.

Stap	Actie	Verwachte reactie	Pass/ Fail	Commentaar
1	Kies optie import project.	Systeem toont optie voor het kiezen van import bestand.		
2	Gebruiker kiest import bestand (Test data).	Systeem toont import widget met data.		
3	Gebruiker kiest voor 1 van elk project element niet importeren.	Systeem toont remap optie.		
4	Gebruiker drukt op ok.	Systeem importeerd data en sluit scherm.		
5	Check post conditie 1	Correct/Fout		

Post-conditie 1. De gespecificeerde data is opgenomen in het project.

Test data

De volgende test data zal in het import bestand aanwezig moeten zijn. Zodra er is geïmporteerd en er wordt opgeslagen moet de inhoud van de project folder deze elementen bevatten. Hierbij moet voor elke categorie 1 element ontbreken.

Parameters

Name	Description	Value	mModifiable
GP1	The first param	20 (Number)	True
GP2		Text (Text)	False
GP3	The third param	0 : { Yes, No } (Choice)	True
GP4		Enabled (Checkbox)	False

Python hooks

Name
drv_hooks.py
pex_hooks.docx

Layers

Name	Number	Type	Color : Hatch	Fill	Description
L1	1:0		255, 0, 255 : 1	True	Normal L1
L2	2:0		0, 0, 0 : 0	False	
L3		Derived	255, 0, 255 : 1	True	Derived L3
L4		Derived	0, 0, 0 : 0	False	

L5		Coloring	255, 0, 255 : 1	True	Coloring L5
L6		Coloring	0, 0, 0 : 0	False	
L7		Connection	255, 0, 255 : 1	True	Connection L7
L8		Connection	0, 0, 0 : 0	False	

Patterns

Name	MaterialLayers	Scene	Variables	Attachments	Description
P1	None	Square	None	None	P1
P2	L1	None	A : 7	File.txt	
P3	L1, L2	Square	None	None	P3
P4	L3	None	B : GP1	File.txt	

Rule instances

Name	Master	Parameters	Layermap	Path (groups)
RI1	P1	GP1	L1	Test/Test1
RI2	P2	GP2	L1	Test/
RI3	P3	GP3	None	Test/Test2
RI4	P4	GP4	L3	Test/Test3

Tests

Name	Spec	Layermap	Minimize count	Parent
T1	RI1	L1	True	S1
T2	RI2	L2	False	S2
T3	P1	L3	True	S1
T4	P2	L4	False	S2

Bestaand project

Voor de use cases waarin wordt uitgegaan van een bestaand project is test data nodig. De test data die aanwezig zal zijn in het project als gesproken wordt over een bestaand project is als volgt:

Parameters

Name	Description	Value	mModifiable
GP1	The first param	20 (Number)	True
GP2		Text (Text)	False
GP3	The third param	0 : { Yes, No } (Choice)	True
GP4		Enabled (Checkbox)	False

Python hooks

Name
drv_hooks.py
pex_hooks.docx

Layers

Name	Number	Type	Color : Hatch	Fill	Description
L1	1:0		255, 0, 255 : 1	True	Normal L1
L2	2:0		0, 0, 0 : 0	False	
L3		Derived	255, 0, 255 : 1	True	Derived L3
L4		Derived	0, 0, 0 : 0	False	
L5		Coloring	255, 0, 255 : 1	True	Coloring L5
L6		Coloring	0, 0, 0 : 0	False	
L7		Connection	255, 0, 255 : 1	True	Connection L7
L8		Connection	0, 0, 0 : 0	False	

Patterns

Name	MaterialLayers	Scene	Variables	Attachments	Description
P1	None	Square	None	None	P1
P2	L1	None	A : 7	File.txt	
P3	L1, L2	Square	None	None	P3
P4	L3	None	B : GP1	File.txt	

Rule instances

Name	Master	Parameters	Layermap	Path (groups)
RI1	P1	GP1	None	Test/Test1
RI2	P2	GP2	L1	Test/
RI3	P3	GP3	None	Test/Test2
RI4	P4	GP4	L3	Test/Test3

Tests

Name	Spec	Layermap	Minimize count	Parent
T1	RI1	None	True	S1
T2	RI2	None	False	S2
T3	P1	None	True	S1
T4	P2	L1 (van P2)	False	S2

Testcase 3

Test case #: 3	Test case naam: Verbeter bestaand project
Systeem: iDRM	Datum: 11-05-2015
Auteur: Floris Velleman	Beschrijving: Importeren van projectelementen in een bestaand project.

Pre-conditie De gebruiker heeft het programma gestart en het bestaande project geladen.

Stap	Actie	Verwachte reactie	Pass/ Fail	Commentaar
1	Kies optie import project.	Systeem toont optie voor het kiezen van import bestand.		
2	Gebruiker kiest import bestand (Test data).	Systeem toont import widget met data.		
3	Gebruiker drukt op ok.	Systeem importeerd data en sluit scherm.		
4	Check post conditie 1	Correct/Fout		

Post-conditie 1. Alle data is opgenomen in het project. 2. Elementen die al bestonden zijn overschreven. 3. Relaties zijn nog correct.

Test data

De volgende test data zal in het import bestand aanwezig moeten zijn. Zodra er is geïmporteerd en er wordt opgeslagen moet de inhoud van de project folder minimaal deze elementen bevatten.

Parameters

Name	Description	Value	mModifiable
GP1	The overwritten param	Text (Text)	False
GP5	The fifth param	10 (Number)	True

Python hooks

Name
pex_hooks.py
test_hooks.py

Layers

Name	Number	Type	Color : Hatch	Fill	Description
L2	2:0		255, 0, 255 : 1	True	Normal L1
L1	1:0		0, 0, 0 : 0	False	
L4		Derived	255, 0, 255 : 1	True	Derived L4
L6		Coloring	255, 0, 255 : 1	True	Coloring L6
L8		Connection	255, 0, 255 : 1	True	Connection L8
L9	12:0		0, 0, 255 : 1	True	New L9

Patterns

Name	MaterialLayers	Scene	Variables	Attachments	Description

P2	None	None	None	None	P2
P1	L1	None	A : 7	File.txt	
P5	L9	Square	None	None	P5

Rule instances

Name	Master	Parameters	Layermap	Path (groups)
RI2	P1	GP1	L1	Test/Test1
RI1	P2	GP5	L1	Test/
RI5	P5	GP5	None	Test/Test2

Tests

Name	Spec	Layermap	Minimize count	Parent
T2	RI1	L1	True	S1
T1	RI2	L2	False	S2
T5	P5	L9	True	S2

Testcase 4

Test case #: 4	Test case naam: Import zonder conflicten
Systeem: iDRM	Datum: 11-05-2015
Auteur: Floris Velleman	Beschrijving: Importeren van projectelementen die geen conflicten geven in een bestaand project.

Pre-conditie
De gebruiker heeft het programma gestart en het bestaande project geladen.

Stap	Actie	Verwachte reactie	Pass/ Fail	Commentaar
1	Kies optie import project.	Systeem toont optie voor het kiezen van import bestand.		
2	Gebruiker kiest import bestand (Test data).	Systeem toont import widget met data.		
3	Gebruiker kiest voor elk conflict niet importeren en drukt vervolgens op ok.	Systeem importeerd data en sluit scherm.		
4	Check post conditie 1	Correct/Fout		

Post-conditie
1. Alle data behalve de conflicten zijn opgenomen in het project. 2. Relaties zijn nog correct.

Test data

De volgende test data zal in het import bestand aanwezig moeten zijn. Zodra er is geïmporteerd en er wordt opgeslagen moet de inhoud van de project folder deze elementen bevatten.

Parameters

Name	Description	Value	mModifiable
GP1	The overwritten param	Text (Text)	False
GP5	The fifth param	10 (Number)	True

Python hooks

Name
pex_hooks.py
test_hooks.py

Layers

Name	Number	Type	Color : Hatch	Fill	Description
L2	2:0		255, 0, 255 : 1	True	Normal L1
L1	1:0		0, 0, 0 : 0	False	
L4		Derived	255, 0, 255 : 1	True	Derived L3
L6		Coloring	255, 0, 255 : 1	True	Coloring L5
L8		Connection	255, 0, 255 : 1	True	Connection L7
L9	12:0		0, 0, 255 : 1	True	New L9

Patterns

Name	MaterialLayers	Scene	Variables	Attachments	Description

P2	None	Square	None	None	P1
P1	L1	None	A : 7	File.txt	
P5	L9	Square	None	None	P3

Rule instances

Name	Master	Parameters	Layermap	Path (groups)
RI2	P1	GP1	L1	Test/Test1
RI1	P2	GP5	L1	Test/
RI5	P5	GP5	None	Test/Test2

Tests

Name	Spec	Layermap	Minimize count	Parent
T2	RI1	L1	True	S1
T1	RI2	L2	False	S2
T5	P5	L9	True	S2

Testcase 5

Alles hernoemen dat conflicten geeft.

Test case #: 5	Test case naam: Import door hernoemen
Systeem: iDRM	Datum: 11-05-2015
Auteur: Floris Velleman	Beschrijving: Importeren van projectelementen die geen conflicten geven in een bestaand project. Alle conflicten worden hernoemd.

Pre-conditie
De gebruiker heeft het programma gestart en het bestaande project geladen.

Stap	Actie	Verwachte reactie	Pass/ Fail	Commentaar
1	Kies optie import project.	Systeem toont optie voor het kiezen van import bestand.		
2	Gebruiker kiest import bestand (Test data).	Systeem toont import widget met data.		
3	Gebruiker kiest voor elk conflict hernoemen en drukt vervolgens op ok.	Systeem importeerd data en sluit scherm.		
4	Check post conditie 1	Correct/Fout		

Post-conditie

1. Alle data behalve de conflicten zijn opgenomen in het project.
2. De conflicten zijn onder een nieuwe naam toegevoegd.
3. Relaties zijn nog correct.

Test data

De volgende test data zal in het import bestand aanwezig moeten zijn. Zodra er is geïmporteerd en er wordt opgeslagen moet de inhoud van de project folder deze elementen bevatten. De uitzondering hierop zijn de conflicten die onder een andere naam moeten zijn opgenomen.

Parameters

Name	Description	Value	mModifiable
GP1	The overwritten param	Text (Text)	False
GP5	The fifth param	10 (Number)	True

Python hooks

Name
pex_hooks.py
test_hooks.py

Layers

Name	Number	Type	Color : Hatch	Fill	Description
L2	2:0		255, 0, 255 : 1	True	Normal L1
L1	1:0		0, 0, 0 : 0	False	
L4		Derived	255, 0, 255 : 1	True	Derived L3
L6		Coloring	255, 0, 255 : 1	True	Coloring L5
L8		Connection	255, 0, 255 : 1	True	Connection L7

L9	12:0		0, 0, 255 : 1	True	New L9

Patterns

Name	MaterialLayers	Scene	Variables	Attachments	Description
P2	None	Square	None	None	P1
P1	L1	None	A : 7	File.txt	
P5	L9	Square	None	None	P3

Rule instances

Name	Master	Parameters	Layermap	Path (groups)
RI2	P1	GP1	L1	Test/Test1
RI1	P2	GP5	L1	Test/
RI5	P5	GP5	None	Test/Test2

Tests

Name	Spec	Layermap	Minimize count	Parent
T2	RI1	L1	True	S1
T1	RI2	L2	False	S2
T5	P5	L9	True	S2

Testcase 6

Importeren van alle patterns (zonder de layers die moeten op iets anders voor patterns).

Test case #: 6	Test case naam: Import zonder layers
Systeem: iDRM	Datum: 11-05-2015
Auteur: Floris Velleman	Beschrijving: Importeren van patterns zonder layers mee te nemen.

Pre-conditie De gebruiker heeft het programma gestart en het bestaande project geladen.

Stap	Actie	Verwachte reactie	Pass/ Fail	Commentaar
1	Kies optie import project.	Systeem toont optie voor het kiezen van import bestand.		
2	Gebruiker kiest import bestand (Test data).	Systeem toont import widget met data.		
3	Gebruiker kiest niet meenemen layers en drukt vervolgens op ok.	Systeem importeerd data en sluit scherm.		
4	Check post conditie 1	Correct/Fout		

Post-conditie 1. Alle patterns zijn opgenomen in het project. 2. De import layers zijn niet opgenomen in het project. 3. Relaties zijn nog correct.

Testdata

De volgende test data zal in het import bestand aanwezig moeten zijn. Zodra er is geïmporteerd en er wordt opgeslagen moet de inhoud van de project folder de patterns bevatten.

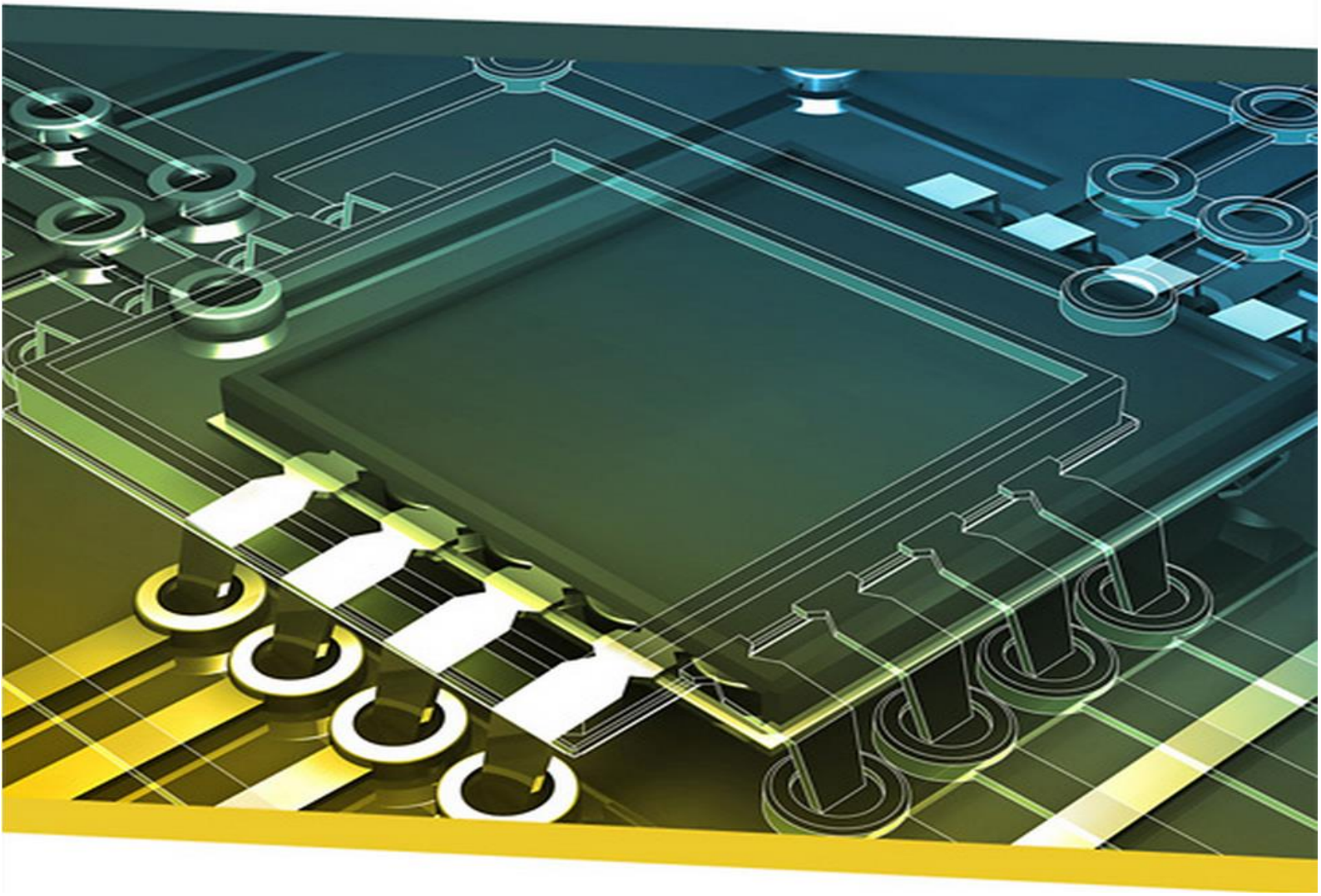
Patterns

Name	MaterialLayers	Scene	Variables	Attachments	Description
P5	L9	Square	None	None	P5
P6	L10	None	A : 7	None	P6
P7	L11	Square	B : 0	File.txt	

Layers

Name	Number	Type	Color : Hatch	Fill	Description
L9	10:0		255, 0, 255 : 1	True	Normal L9
L10	20:0		0, 255, 0 : 0	False	Normal L10
L11	30:0		0, 0, 0 : 1	True	Normal L11

12.5. Bijlage 5 – Testrapport



Project import/export binnen iDRM

Testrapport

Student

Naam:	Velleman
Voorletters:	F.M.V.
Roepnaam:	Floris
Studentnummer:	1602376
Docentbegeleider:	Alex Jongman

Bedrijf

Naam:	NP-Komplete Technologies
Adres:	De Rozentuin 18
Postcode:	5611 SW
Plaats:	Eindhoven
Inhoudelijk bedrijfsbegeleider:	Maarten Berkens
Technisch bedrijfsbegeleider:	Rob Gelderblom

Criteria voor goedkeuring test

Tijdens het testen van de architectuur van de import widget zijn er een aantal opties voor de mogelijk resultaten van de tests. Deze zijn aangegeven op basis van kleur en geven een weergave van welke functionaliteit werkt/niet werkt.

Groen

De testen die met groen zijn aangegeven zijn testen die slagen en aan de verwachte resultaten voldoen. Deze kunnen voorkomen bij zowel het proces dat wordt doorlopen als de data die uiteindelijk wordt verwacht.

Indien er bij de meest recente onderdelen van de tests enkel groen (of oranje) staat is de test geslaagd en voldoet de functionaliteit aan de verwachtingen die gesteld zijn.

Oranje

Testen die een oranje status hebben zijn testen die een fout bevatten. Deze fout is tijdens het testen duidelijk geworden en heeft geen directe invloed op de testcase. De test case zal dus slagen ondanks deze fout.

Hierbij kan gedacht worden aan fouten in de interface. Dit zorgt er voor dat er mogelijk een incorrecte weergave wordt gegeven van wat er gebeurd. Dit zijn aandachtspunten die in latere iteraties kunnen worden behandeld.

Rood

Als een test met rood is aangegeven is er een fout in de uitgewerkte functionaliteit en kan het programma niet functioneren zoals het zou moeten.

Indien er in een test iets fout gaat dat als rood wordt aangemerkt zal de test gefaald zijn.

Testcase 1

Test case #: 1	Test case naam: Import bestaand project
Systeem: iDRM	Test datum: 13-05-2015
Auteur: Floris Velleman	Beschrijving: Importeren van projectelementen in een leeg project.

Pre-conditie De gebruiker heeft het programma gestart. Het programma toont een leeg project.

Stap	Actie	Verwachte reactie	Pass/ Fail	Commentaar
1	Kies optie import project.	Systeem toont optie voor het kiezen van import bestand.		
2	Gebruiker kiest import bestand (Test data).	Systeem toont import widget met data.		
3	Gebruiker drukt op ok.	Systeem importeerd data en sluit scherm.		
4	Check post conditie	Correct/Fout		

Post-conditie 2. Alle data is opgenomen in het project.

Iteratie 1

Onderdeel	**Status/Opmerking**
Parameters	
Python hooks	
Layers	
Patterns	
Rule instances	
Tests	Geen interface/optie voor importeren van tests.

Iteratie 2

Onderdeel	**Status/Opmerking**
Parameters	
Python hooks	
Layers	
Patterns	
Rule instances	
Tests	

Testcase 2

Test case #: 2	Test case naam: Import deel bestaand project
Systeem: iDRM	Test datum: 13-05-2015
Auteur: Floris Velleman	Beschrijving: Importeren van projectelementen in een leeg project.

Pre-conditie De gebruiker heeft het programma gestart. Het programma toont een leeg project.

Stap	Actie	Verwachte reactie	Pass/ Fail	Commentaar
1	Kies optie import project.	Systeem toont optie voor het kiezen van import bestand.		
2	Gebruiker kiest import bestand (Test data).	Systeem toont import widget met data.		
3	Gebruiker kiest voor 1 van elk project element niet importeren.	Systeem toont remap optie.		
4	Gebruiker drukt op ok.	Systeem importeerd data en sluit scherm.		
5	Check post conditie	Correct/Fout		

Post-conditie 2. De gespecificeerde data is opgenomen in het project.

Iteratie 1

Onderdeel	**Status/Opmerking**
Parameters	
Python hooks	
Layers	
Patterns	
Rule instances	Remappen van patterns gaat fout (segmentation fault)
Tests	Geen interface/optie voor importeren van tests.

Iteratie 2

Onderdeel	**Status/Opmerking**
Parameters	
Python hooks	
Layers	
Patterns	
Rule instances	
Tests	

Testcase 3

Test case #: 3	Test case naam: Verbeter bestaand project
Systeem: iDRM	Test datum: 12-05-2015
Auteur: Floris Velleman	Beschrijving: Importeren van projectelementen in een bestaand project.

Pre-conditie De gebruiker heeft het programma gestart en het bestaande project geladen.

Stap	Actie	Verwachte reactie	Pass/ Fail	Commentaar
1	Kies optie import project.	Systeem toont optie voor het kiezen van import bestand.		
2	Gebruiker kiest import bestand (Test data).	Systeem toont import widget met data.		
3	Gebruiker drukt op ok.	Systeem importeerd data en sluit scherm.		
4	Check post conditie	Correct/Fout		

Post-conditie 4. Alle data is opgenomen in het project. 5. Elementen die al bestonden zijn overschreven. 6. Relaties zijn nog correct.

Iteratie 1

Onderdeel	**Status/Opmerking**
Parameters	
Python hooks	
Layers	Segmentation fault bij het importeren van een layer die al bestaat zonder nummer.
Patterns	
Rule instances	
Tests	Geen interface/optie voor importeren van tests.

Iteratie 2

Onderdeel	**Status/Opmerking**
Parameters	
Python hooks	
Layers	
Patterns	
Rule instances	
Tests	

Testcase 4

Test case #: 4	Test case naam: Import zonder conflicten
Systeem: iDRM	Test datum: 12-05-2015
Auteur: Floris Velleman	Beschrijving: Importeren van projectelementen die geen conflicten geven in een bestaand project.

Pre-conditie De gebruiker heeft het programma gestart en het bestaande project geladen.

Stap	Actie	Verwachte reactie	Pass/ Fail	Commentaar
1	Kies optie import project.	Systeem toont optie voor het kiezen van import bestand.		
2	Gebruiker kiest import bestand (Test data).	Systeem toont import widget met data.		
3	Gebruiker kiest voor elk conflict niet importeren en drukt vervolgens op ok.	Systeem importeerd data en sluit scherm.		
4	Check post conditie	Correct/Fout		

Post-conditie 3. Alle data behalve de conflicten zijn opgenomen in het project. 4. Relaties zijn nog correct.

Iteratie 1

Onderdeel	**Status/Opmerking**
Parameters	
Python hooks	
Layers	
Patterns	
Rule instances	De rule instances worden niet correct geupdate in de interface (zo kunnen er dubbele groepen zijn (twee maal groep Test in de hoofdstructuur). Zodra er wordt opgeslagen en opnieuw wordt gestart zijn deze groepen weer 1 groep.
Tests	Geen interface/optie voor importeren van tests.

Iteratie 2

Onderdeel	**Status/Opmerking**
Parameters	
Python hooks	
Layers	
Patterns	
Rule instances	De rule instances worden niet correct geupdate in de interface (zo kunnen er dubbele groepen zijn (twee maal groep Test in de hoofdstructuur). Zodra er wordt opgeslagen en

	opnieuw wordt gestart zijn deze groepen weer 1 groep.
Tests	

Om deze test uit te voeren is er gebruik gemaakt van het niet meenemen van objecten (in de interface). Dit zorgde ervoor dat conflicten niet werden meegenomen in tegenstelling tot de andere opties die er waren. Als er direct op ok wordt gedrukt wordt automatisch overschreven.

Testcase 5

Test case #: 5	Test case naam: Import door hernoemen
Systeem: iDRM	Test datum: 12-05-2015
Auteur: Floris Velleman	Beschrijving: Importeren van projectelementen die geen conflicten geven in een bestaand project. Alle conflicten worden hernoemd.

Pre-conditie
De gebruiker heeft het programma gestart en het bestaande project geladen.

Stap	Actie	Verwachte reactie	Pass/ Fail	Commentaar
1	Kies optie import project.	Systeem toont optie voor het kiezen van import bestand.		
2	Gebruiker kiest import bestand (Test data).	Systeem toont import widget met data.		
3	Gebruiker kiest voor elk conflict hernoemen en drukt vervolgens op ok.	Systeem importeerd data en sluit scherm.		
4	Check post conditie	Correct/Fout		

Post-conditie
4. Alle data behalve de conflicten zijn opgenomen in het project. 5. De conflicten zijn onder een nieuwe naam toegevoegd. 6. Relaties zijn nog correct.

Iteratie 1

Onderdeel	**Status/Opmerking**
Parameters	
Python hooks	
Layers	Vanwege dubbele keys is het niet mogelijk elementen te renamen die meerdere elementen kunnen overschrijven. Import: L1 (1:0), L10 (2:0) Huidig: L1 (2:0), L10 (1:0) Er zijn hier meerdere mogelijkheden tot overschrijvne en het renamen van L1 zou mogelijk zijn. Deze optie wordt echter niet gegeven.
Patterns	Segmentation fault bij het importeren van pattern met optie rename.
Rule instances	
Tests	Geen interface/optie voor importeren van tests.

Iteratie 2

Onderdeel	**Status/Opmerking**
Parameters	
Python hooks	

Layers	Vanwege dubbele keys is het niet mogelijk elementen te renamen die meerdere elementen kunnen overschrijven. Import: L1 (1:0), L10 (2:0) Huidig: L1 (2:0), L10 (1:0) Er zijn hier meerdere mogelijkheden tot overschrijvne en het renamen van L1 zou mogelijk zijn. Deze optie wordt echter niet gegeven.
Patterns	
Rule instances	
Tests	

Testcase 6

Test case #: 6	Test case naam: Import zonder layers
Systeem: iDRM	Test datum: 12-05-2015
Auteur: Floris Velleman	Beschrijving: Importeren van patterns zonder layers mee te nemen.

Pre-conditie De gebruiker heeft het programma gestart en het bestaande project geladen.

Stap	Actie	Verwachte reactie	Pass/ Fail	Commentaar
1	Kies optie import project.	Systeem toont optie voor het kiezen van import bestand.		
2	Gebruiker kiest import bestand (Test data).	Systeem toont import widget met data.		
3	Gebruiker kiest niet meenemen layers en drukt vervolgens op ok.	Systeem importeerd data en sluit scherm.		
4	Check post conditie	Correct/Fout		

Post-conditie 4. Alle patterns zijn opgenomen in het project. 5. De import layers zijn niet opgenomen in het project. 6. Relaties zijn nog correct.

Iteratie 1

Onderdeel	**Status/Opmerking**
Layers	Niet meegenomen
Patterns	Meegenomen, referentie naar layer is op <unspecified> gezet.

Iteratie 2

Onderdeel	**Status/Opmerking**
Layers	Niet meegenomen
Patterns	Meegenomen, referentie naar layer is op <unspecified> gezet.

Resultaten Iteratie 1

Op basis van de bevonden resultaten kan de volgende tabel worden opgesteld:

Test case	Status
1	
2	
3	
4	
5	
6	

Resultaten Iteratie 2

Op basis van de bevonden resultaten kan de volgende tabel worden opgesteld:

Test case	Status
1	
2	
3	
4	
5	
6	

12.6. Bijlage 6 – Test scenario en bevindingen

Start iDRM op en laad project p of een andere project waar alle soorten elementen in zitten. Zodra het programma is gestart wordt de import optie gekozen en wordt gekozen voor een import van het bestand import1 (tgz). Zodra dit bestand is gekozen en er op okee is gedrukt kan de gebruiker starten met de test.

Test template

Vraag 1: Wat is het eerste dat je opvalt in dit scherm en waarom?
Vraag 2: Hoe verwacht je dat dit scherm ongeveer werkt? Hoe zou je navigeren in dit scherm? En waarom?

Laat de gebruiker door de subschermen navigeren.

Vraag vervolgens of de gebruiker naar het Layers scherm kan gaan.
Geef aan dat de layers in het project niet mee moeten worden genomen en vraag of de gebruiker dit kan doen.

Vraag 3: Wat verwacht je dat er wordt geïmporteerd als je nu op okee drukt? En waarom?

Vraag de gebruiker om naar Global parameters te gaan.

Vraag 4: Wat valt je op in dit scherm ten opzichte van het vorige scherm? (Laat de gebruiker niet wisselen van tab)
Vraag 5: Hoe denk je meer informatie te kunnen krijgen over dit object? En waarom?

Vraag de gebruiker op okee te drukken en te importeren.

Vraag 6: Wat is het eerste dat opvalt nu er is geïmporteerd?

Antwoorden Simon (18-05-2015)

1: Er valt me niet direct iets op behalve het grote witte vlak.
2: Ik verwacht dat ik met de Ok knop kan importeren en dat de Cancel knop ervoor zorgt dat ik niet importeer. Als ik een ander scherm wil kan ik in de tree klikken?
3: Ik verwacht dat het project wordt geïmporteerd en dat de Patterns hun layers op unspecified hebben. De layers moeten niet mee zijn genomen.
4: Er zijn meer waardes voor dit onderdeel dus ik kan snel zien wat alles is.
5: Ik denk dat dat kan door te dubbelklikken omdat ik dat al een keer bij jou heb gezien.
6: De patterns worden erg traag geïmporteerd en ik zie een hoop rood maar dat is vanwege de layers die niet zijn meegenomen.

Algemene bevindingen:

1. Maakt het scherm breder zodra deze start.
2. Gebruikt de snelle opties bovenin het scherm.

Algemene opmerkingen:

1. Lege onderdelen zijn niet grijs, zou zo kunnen zijn.
2. De header van het scherm zegt alleen import daar zou meer kunnen staan.
3. Ok button zou import button moeten zijn.
4. De lijsten zijn klein en erg lang, zou makkelijk zijn om meer onderdelen in de lijst te zien.

Antwoorden Rob G (18-05-2015)

1: Veel informatie.
2: Met het muiswieltje kan ik door de lijst heen en ik denk dat ik kan navigeren met de tree.
3: Layers worden niet meegenomen en de rest wel.
4: Sommige velden zijn disabled.
5: Leest de tooltip niet en klikt een beetje overal op. Uiteindelijk dubbelklikt hij op een rij en dat opent het scherm.
6: Het project geeft aan dat er veranderingen zijn en dat je kunt opslaan.

Algemene bevindingen:

1. Las de tooltip helemaal niet terwijl die meerdere keren in het scherm stond.
2. Snapt map to existing niet helemaal (Wat doet dit dan en waar zit niet importen).

Algemene opmerkingen:

1. Waarom zijn de velden onder global parameters disabled.
2. Het zou handiger zijn als je een hele rij kan selecteren in plaats van een kolom (dit doet nog niks).
3. Disabled text velden zijn soms inconsequent.
4. Een contextmenu zou handig zijn voor mensen die geen tooltips lezen.

Antwoorden Rob O (18-05-2015)

1: Een lijst.
2: Met de tree onderdelen?
3: Dat alles geïmporteerd wordt behalve layers.
4: Valt niet zo veel op, ze komen volgens mij wel overeen.
5: Ziet tooltip niet en klikt voornamelijk op combo box. Vind na korte tijd echter de tooltip.
6: Er staan een heleboel patterns bij.

Algemene bevindingen:

1. Gebruikt de tree niet meenemen in plaats van de quick opties.
2. Weet niet wat het vraagteken betekend.

Algemene opmerkingen:

1. Hernoem de tree items naar de bijbehorende workspaces. Dat zorgt ervoor dat de gebruiker sneller weet wat er verwacht kan worden op een pagina.
2. Vraagteken veranderen naar (C = conflict, I = import as-is, M = import modified, X = don't import) met handige kleuren hiervoor?
3. '-' als tabel header mag weg dat kan gewoon leeg.
4. Het zou handig zijn als de oude zichtbaar waren.

Antwoorden Jos

1: Tabel linksbovenin heeft een vreemde overgang maar dat kan ook door mijn scherm komen.
2: Scrollen met de muis. En de tree kun je waarschijnlijk in klikken?
3: Het project zonder de layers.
4: Er ontbreekt volgens mij een kolom.
5: *leest de tooltip* door dubbel te klikken op een rij.
6: Er zijn patterns bijgekomen volgens mij.

Algemene bevindingen:

1. Navigeren met scrollen is riskant als je over een combobox heen gaat met de muis.

Algemene opmerkingen:

1. Verwijder '-' uit de header van de tabel.
2. Maak combobox niet scrollbaar.

12.7. Bijlage 7 – Testscripts

Import part of project

```
ClickMenu,      "",      "",      menuEdit
ClickMenu,      "",      menuEdit,     actionImportWidget
PressKeys,      "",      inputDialogEdit, "/Project1/import1.tgz", -389,      -43
ClickObject,    "",      inputDialogAccept, 18,      6,      LMouseButton
ClickList,      "",      PatternComboImport0, LMouseButton,  Add to project
ClickList,      "",      PatternComboImport0, LMouseButton,  Map to existing
ClickList,      "",      projectTree,   LMouseButton,  Layers
ClickList,      "",      LayerComboImport0, LMouseButton,  Add to project
ClickList,      "",      LayerComboImport0, LMouseButton,  Map to existing
ClickList,      "",      projectTree,   LMouseButton,  Rule instances
ClickList,      "",      RuleInstanceComboImport0, LMouseButton,  Add to project
ClickList,      "",      RuleInstanceComboImport0, LMouseButton,  Map to existing
ClickList,      "",      projectTree,   LMouseButton,  Tests
ClickList,      "",      TestComboImport0, LMouseButton,  Add to project
ClickList,      "",      TestComboImport0, LMouseButton,  Map to existing
ClickList,      "",      projectTree,   LMouseButton,  Global parameters
ClickList,      "",      ParameterComboImport0, LMouseButton,  Add to project
ClickList,      "",      ParameterComboImport0, LMouseButton,  Map to existing
ClickObject,    "",      acceptButton,  46,      18,      LMouseButton
ClickMenu,      "",      "",      menuFile
ClickMenu,      "",      menuFile,     actionSave_Project
```

Import patterns

```
ClickMenu,      "",      "",      menuEdit
ClickMenu,      "",      menuEdit,     actionImportWidget
PressKeys,      "",      inputDialogEdit, "/Project1/import1.tgz", -467,      -123
ClickObject,    "",      inputDialogAccept, 50,      14,      LMouseButton
ClickList,      "",      projectTree,   LMouseButton,  Layers
ClickList,      "",      LayerComboImport0, LMouseButton,  Add to project
ClickList,      "",      LayerComboImport0, LMouseButton,  Map to existing
ClickList,      "",      LayerComboImport1, LMouseButton,  Add to project
ClickList,      "",      LayerComboImport1, LMouseButton,  Map to existing
ClickList,      "",      LayerComboImport2, LMouseButton,  Add to project
ClickList,      "",      LayerComboImport2, LMouseButton,  Map to existing
ClickObject,    "",      acceptButton,  46,      10,      LMouseButton
ClickMenu,      "",      "",      menuFile
ClickMenu,      "",      menuFile,     actionSave_Project
```

Import and rename all conflicts

```
ClickMenu,      "",      "",      menuEdit
ClickMenu,      "",      menuEdit,     actionImportWidget
PressKeys,      "",      inputDialogEdit, "/Project1/import1.tgz", -476,      -117
PressKey,       "",      inputDialogEdit, Return,      -476,      -117
ClickList,      "",      PatternComboImport0, LMouseButton,  Overwrite existing element
ClickList,      "",      PatternComboImport0, LMouseButton,  Rename import element
ClickList,      "",      PatternComboImport1, LMouseButton,  Overwrite existing element
ClickList,      "",      PatternComboImport1, LMouseButton,  Rename import element
ClickList,      "",      projectTree,   LMouseButton,  Layers
ClickList,      "",      LayerComboImport2, LMouseButton,  Overwrite existing element
```

```
ClickList,      "",      LayerComboImport2, LMouseButton,  Rename import element
ClickList,      "",      projectTree,   LMouseButton,  Rule instances
ClickList,      "",      RuleInstanceComboImport0, LMouseButton,  Overwrite existing element
ClickList,      "",      RuleInstanceComboImport0, LMouseButton,  Rename import element
ClickList,      "",      RuleInstanceComboImport1, LMouseButton,  Overwrite existing element
ClickList,      "",      RuleInstanceComboImport1, LMouseButton,  Rename import element
ClickList,      "",      projectTree,   LMouseButton,  Global parameters
ClickList,      "",      ParameterComboImport0, LMouseButton,  Overwrite existing element
ClickList,      "",      ParameterComboImport0, LMouseButton,  Rename import element
ClickObject,    "",      acceptButton,   972,      9,      LMouseButton
ClickMenu,      "",      "",      menuFile
ClickMenu,      "",      menuFile,    actionSave_Project
```

Improve existing project with import project

```
ClickMenu,      "",      "",      menuEdit
ClickMenu,      "",      menuEdit,    actionImportWidget
PressKeys,      "",      inputDialogEdit, "/../in/",      -472,      -117
ClickObject,    "",      inputDialogEdit, 195,      12,      LMouseButton
PressKeys,      "",      inputDialogEdit, "import1.tgz",   195,      12
ClickObject,    "",      InputDialog,   160,      8,      LMouseButton
PressKey,       "",      inputDialogEdit, Return,      149,      -24
ClickObject,    "",      acceptButton,   972,      9,      LMouseButton
ClickMenu,      "",      "",      menuFile
ClickMenu,      "",      menuFile,    actionSave_Project
```

Import in new project

```
ClickMenu,      "",      "",      menuEdit
ClickMenu,      "",      menuEdit,    actionImportWidget
PressKeys,      "",      inputDialogEdit, "/Project1/import1.tgz", -421,      -44
PressKey,       "",      inputDialogEdit, Return,      -421,      -44
ClickObject,    "",      acceptButton,  34,      18,      LMouseButton
ClickMenu,      "",      "",      menuFile
ClickMenu,      "",      menuFile,    actionSave_Project
```

Import without any conflicts

```
ClickMenu,      "",      "",      menuEdit
ClickMenu,      "",      menuEdit,    actionImportWidget
PressKeys,      "",      inputDialogEdit, "/imp",      -496,      -128
PressKey,       "",      inputDialogEdit, Backspace,    -496,      -128
PressKey,       "",      inputDialogEdit, Backspace,    -496,      -128
PressKey,       "",      inputDialogEdit, Backspace,    -496,      -128
PressKeys,      "",      inputDialogEdit, "Project1/import1.tgz", -496,      -128
PressKey,       "",      inputDialogEdit, Return,      -496,      -128
ClickList,      "",      PatternComboImport0, LMouseButton,  Overwrite existing element
ClickList,      "",      PatternComboImport0, LMouseButton,  Keep original element
ClickList,      "",      PatternComboImport1, LMouseButton,  Overwrite existing element
ClickList,      "",      PatternComboImport1, LMouseButton,  Keep original element
ClickList,      "",      projectTree,   LMouseButton,  Layers
ClickList,      "",      LayerComboImport0, LMouseButton,  Overwrite existing element
ClickList,      "",      LayerComboImport0, LMouseButton,  Keep original element
ClickList,      "",      LayerComboImport1, LMouseButton,  Overwrite existing element
ClickList,      "",      LayerComboImport1, LMouseButton,  Keep original element
```

```
ClickList,        "",        LayerComboImport2, LMouseButton,  Overwrite existing element
ClickList,        "",        LayerComboImport2, LMouseButton,  Keep original element
ClickList,        "",        projectTree,   LMouseButton,  Rule instances
ClickList,        "",        RuleInstanceComboImport0, LMouseButton,  Overwrite existing element
ClickList,        "",        RuleInstanceComboImport0, LMouseButton,  Keep original element
ClickList,        "",        RuleInstanceComboImport1, LMouseButton,  Overwrite existing element
ClickList,        "",        RuleInstanceComboImport1, LMouseButton,  Keep original element
ClickList,        "",        projectTree,   LMouseButton,  Global parameters
MousePress,       "",        ParameterComboImport0, LMouseButton,  35,         17
MouseDrag,        "",        ParameterComboImport0, LMouseButton,  38,         21
ClickList,        "",        ParameterComboImport0, LMouseButton,  Keep original element
ClickObject,      "",        acceptButton,    972,        9,          LMouseButton
ClickMenu,        "",        "",         menuFile
ClickMenu,        "",        menuFile,     actionSave_Project
```