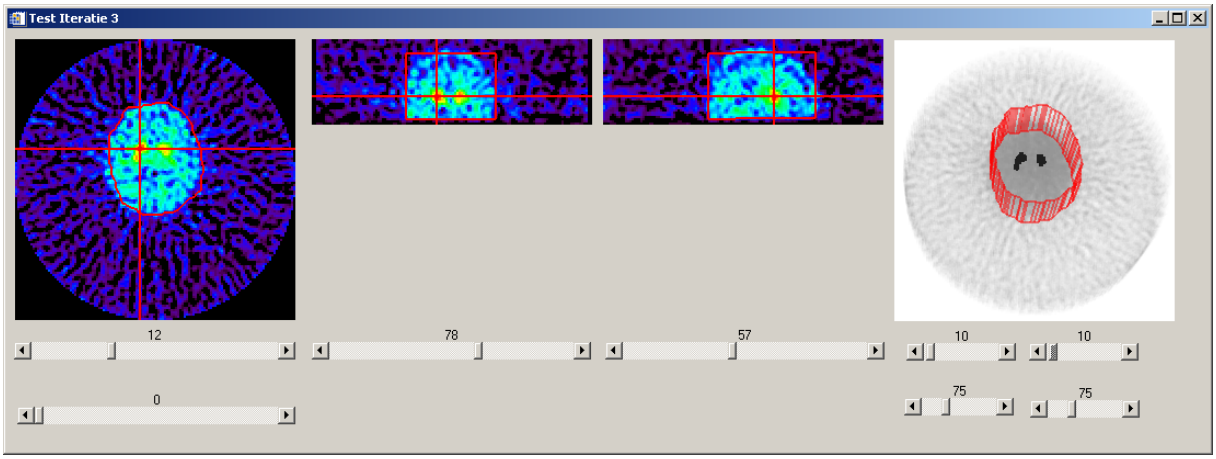
AFSTUDEERVERSLAG



3D Visualisatie in iMed

Datum voltooid: 08-06-2009
Auteur: Johan van Soest
Versie: 1.0

AFSTUDEERVERSLAG VOOR FONTYS HOGESCHOOL ICT

iMed Framework

3D Edition

Gegevens student:
J.P.A. van Soest
Studentnummer 2075764

Opleiding Informatica (afstudeerrichting Software Engineering)
Voltijd

Afstudeerperiode van 9-2-2009 t/m 10-7-2009

Afstudeerbedrijf:
UMC St. Radboud
Afdeling Nucleaire Geneeskunde
Nijmegen

Bedrijfsbegeleiders:
Ing. L. de Groot - Senior Software Engineer
Ing. E. Janssen - Senior Software Engineer

Docentbegeleider:
Ir. L.F.M. van Gestel

Gegevens verslag:
3D visualisatie in iMed
Uitgifte: 8-6-2009

Getekend voor gezien door bedrijfsbegeleiders:

Datum:

De bedrijfsbegeleiders,

Voorwoord

Door de doorlopen minor (Medische Technologie) was de zoektocht voor een afstudeeropdracht van tevoren al beïnvloed. Deze minor had voor een deel laten zien wat er op medisch gebied mogelijk is met informatica. Hierdoor ging de keuze al snel naar een opdracht die gericht was op de gezondheidszorg of medische wereld.

Met dit idee in het achterhoofd heb ik gezocht naar een afstudeeropdracht. Hierbij kwam het UMC St. Radboud al snel naar voren. Via kennissen zou het misschien mogelijk zijn om op het "klinisch fysisch laboratorium" een afstudeeropdracht uit te voeren, maar daar kwam verder geen informatie over terug.

Tijdens een zoektocht door het Stage- en Afstudeer Informatie Systeem (SAIS) van de opleiding kwam een oude afstudeeropdracht naar voren vanuit de afdeling Nucleaire Geneeskunde van het UMC St. Radboud. Aangezien de afstudeeropdracht al enkele jaren in het systeem stond, heb is er contact gezocht of deze nog wel relevant was. Het bleek al snel dat de mogelijkheid voor een afstudeeropdracht aanwezig was.

Dit afstudeerverslag geeft een samenvatting over de uitgevoerde werkzaamheden, en de uitdagingen die hierbij zijn ontstaan/opgelost.

Graag wil ik een aantal mensen bedanken voor het mogelijk maken van deze afstudeeropdracht. Allereerst zou ik de medewerkers van de afdeling Nucleaire Geneeskunde willen bedanken voor de openheid en gezelligheid op de werkvloer, en voor het meedenken tijdens de opdracht. Een speciaal woord van dank gaat hierbij naar Ing. Laurens de Groot en Ing. Erik Janssen voor de begeleiding. Hierbij was bij aanvang niet duidelijk of het aantal contacturen/begeleiding voldoende zou zijn, maar dit blijkt achteraf geen probleem te zijn geweest.
Verder zou ik drs. Dennis Vriens willen bedanken voor de input vanuit de gebruikersgroep (de artsen) waarbij hij zelf ook regelmatig met technische ideeën aan kwam zetten.
Natuurlijk ben ik prof. dr. Wim Oyen ook dank verschuldigd voor het beschikbaar stellen van deze afstudeeropdracht.

Nijmegen, juni 2009

Documenthistorie

Revisies

Versie	Status	Datum	Wijzigingen
0.1	concept	09-03-2009 - 27-03-2009	Opzet van Template & eerste implementatie
0.2	concept	05-04-2009 – 04-05-2009	Aanpassingen vanuit feedback docentbegeleider, en verdere uitwerking tot huidige status afstudeerstage
0.3	concept	07-05-2009 – 08-05-2009	Aanpassingen vanuit feedback bedrijfsbegeleiders, en toevoegen van bijlage betreffende architectuur / codefragmenten.
0.4	concept	20-05-2009	Paragraaf over object graphics toegevoegd, en opmerkingen docentbegeleider verwerkt
0.5	concept	02-06-2009	Feedback bedrijfsbegeleider (versie 0.4) verwerkt
0.6	concept	05-06-2009	Spellingscontrole en feedback docentbegeleider
1.0	Final	08-06-2009	Laatste feedback docentbegeleider en bedrijfsbegeleider, en figuurnummers op orde gezet

Goedkeuring

Dit document behoeft de volgende goedkeuringen:

Versie	Datum goedkeuring	Naam	Functie	Paraaf
1				

Distributie

Dit document is verstuurd aan:

Versie	Datum verzending	Naam	Functie
0.1	27-03-2009	Ir. L.F.M. van Gestel	Docentbegeleider
0.2	04-05-2009	Ing. L. de Groot, Ing. E. Janssen	Bedrijfsbegeleiders
0.3	08-05-2009	Ir. L.F.M. van Gestel	Docentbegeleider
0.4	28-05-2009	Ing. L. de Groot	Bedrijfsbegeleider
0.5	04-06-2009	Ir. L.F.M. van Gestel	Docentbegeleider
0.6	05-06-2009	Ir. L.F.M. van Gestel	Docentbegeleider
0.6	06-06-2009	Ing. E. Janssen	Bedrijfsbegeleider
1.0	09-06-2009	Ir. L.F.M. van Gestel	Docentbegeleider
1.0	09-06-2009	Ing. L. de Groot, Ing. E. Janssen	Bedrijfsbegeleiders
1.0	09-06-2009	Ir. A. Kasper	Voorzitter afstudeercommissie
1.0	09-06-2009	n.n.b.	Hogeschooldeskundige

Samenvatting

In de diagnostiek van tumoren wordt regelmatig gebruik gemaakt van PET-scans. Deze scans geven een beeld van bijvoorbeeld de locatie of activiteit van de tumor.

Binnen het UMC St. Radboud worden deze scans gemaakt op de afdeling Nucleaire Geneeskunde. De beelden die hieruit voortkomen worden door de dienstdoende Nucleair Geneeskundige beoordeeld. Van daaruit zal dan, waar nodig, een behandeling gekozen worden.

Een van de mogelijkheden voor het analyseren en weergeven van de scan-data is het e.Soft Workstation. In het Workstation zijn allerlei functies beschikbaar die een arts ondersteunen in het analyseren van de beelden.

Deze analysesoftware kan uitgebreid worden met eigen programma's. Hiervoor zijn twee informatici op de afdeling werkzaam. Deze informatici hebben een framework gemaakt voor standaard computaties en weergaves die vaak voorkomen bij het ontwikkelen van nieuwe uitbreidingen op het e.Soft systeem.

Mijn afstudeeropracht had als doel het huidige framework uit te breiden met nieuwe functionaliteit. Het betrof in dit geval 3D visualiseringen en standaard-berekeningen. Het project is in drie iteraties verlopen waarbij in elke iteratie een deel van de functionaliteit is gebouwd.

In dit afstudeerverslag wordt beschreven hoe de ontwikkeling van de 3D uitbreiding is verlopen, en wat de uitdagingen hierbij waren. Verder geeft het een indicatie over de manier waarop het proces en het vooraf opgestelde tijdsplan is uitgevoerd.

Summary

In tumor-diagnostics, PET-scans are commonly used. Those scans give information to the doctors about the location or activity of the tumor.

At the UMC St. Radboud, those PET-scans are made at the department of Nuclear Medicine. These scans are interpreted by the Nuclear Medicine doctor on-duty. This doctor reviews the images, and makes up the result of the scan. From there on, some sort of therapy will be chosen (only if needed).

One of the options to view and analyze the scan data, is to use the e.Soft Workstation. In this workstation, there are all kinds of functions available to support a doctor analyzing the images.

The software to analyze those images can be extended with self-created programs. Therefore there are two informaticians working part-time at the department. They have created a framework to speed-up the development of extension programs. This framework contains standard computations and display types which are easy to implement in the self-written applications.

My graduation assignment was about to extend the current framework with new functionality. This functionality was to visualize images in 3D, and to build standard-computations. This assignment was divided into three iterations, in which the functionality was build piece by piece.

This graduation report describes how the development process of the 3D extension of the framework went, and which sort of challenges have risen during the process. It also gives an indication how the actual schedule went, opposite of the predefined schedule.

Woordenlijst

Coronaal

Een aanzicht waarbij het beeld "van voren" wordt weergegeven bij de mens. Een ander woord voor dit aanzicht is het frontaal aanzicht.

DICOM

Een bestandsindeling die wordt gebruikt voor het opslaan van afbeelding(en) en afbeeldinginformatie. Hierin worden mogelijk meerdere afbeeldingen opgeslagen, en kan er informatie over de patiënt en de scan worden opgeslagen.

DrawWidget

Een drawWidget is een klasse binnen IDL. Het is een Widget (zie Widget) waarop "getekend" kan worden. Dat wil zeggen dat lijnen, afbeeldingen of vormen zoals een kubus of cirkel hierop weergegeven kunnen worden.

e.Soft

e.Soft is een workstation van Siemens voor beeldverwerking in de Nucleaire Geneeskunde. Bij dit workstation is het mogelijk om eigen programma's te schrijven die worden aangeroepen door het e.Soft systeem. Het e.Soft Workstation bestaat uit de computer (de hardware) met bijbehorende software. Het is een totaalpakket dat door Siemens wordt geïnstalleerd. Mocht het nodig zijn dat de hardware (de computer) of software een update krijgt, zal dit door Siemens worden uitgevoerd.

IDL

IDL is de afkorting voor Interactive Data Language. Dit is de programmeertaal waarin de uitbreidingen voor e.Soft worden geschreven. Deze programmeertaal is gebaseerd op C, waarbij libraries beschikbaar zijn die het mogelijk maken om eenvoudig applicaties voor beeldverwerking te ontwikkelen.

iMed

De naam van het intern-ontwikkelde framework door de programmeurs die werkzaam zijn op de afdeling Nucleaire Geneeskunde van het UMC St. Radboud, Nijmegen.

PET

Afkorting voor Positron Emissie Tomografie. Een techniek waarbij met behulp van sensoren wordt bepaald waar een positron botst met een elektron. Door deze botsingen te tellen kan een 3D dataset worden gemaakt waarin de verdeling van activiteit/botsingen ligt opgeslagen.

Plane

Één afbeelding uit een volume in een vooraf gedefinieerd aanzicht. Dit aanzicht kan transversaal, sagittaal of coronaal zijn.

Radiofarmacon

Zie tracer

Radionucleïde

Een ander woord voor radioactief isotoop. Dit is een stof die vervalt, waarbij het verval zorgt voor het ontstaan van radioactieve straling.

ROI

ROI is de afkorting voor een Region of Interest. Dit is een gebied binnen een 2D afbeelding waarop berekeningen worden uitgevoerd.

Sagittaal

Aanzicht waarbij het beeld "van de zijkant" wordt weergegeven bij de mens. Dit is te vergelijken met een zijaanzicht van een object.

Tijdsframe

Een afbeelding (of groep afbeeldingen) binnen een bepaalde tijdsduur. De tijdsduur waar één tijdsframe over gaat is in het DICOM-bestand te achterhalen. Evenals het aantal tijdsframes.

Tracer

Een stof waarin een radioactief isotoop is verwerkt. Binnen de Nucleaire Geneeskunde zal een tracer meestal synthese van een (biologisch) natuurlijke stof zijn.

Transversaal

Aanzicht waarbij het beeld "van boven" wordt weergegeven bij de mens. Dit is te vergelijken met het bovenaanzicht van een object.

VOI

VOI is de afkorting voor een Volume of Interest. Dit is een gebied binnen een 3D omgeving waarop berekeningen worden uitgevoerd.

Widget

Aanduiding voor een grafische component binnen IDL. Een widget kan een venster, knop, tekstvak of een ander vergelijkbaar component zijn.

Inhoudsopgave

1 Inleiding

Dit document heeft als doel mijn afstudeerstage in het UMC St. Radboud te Nijmegen (verder genoemd UMCN) te beschrijven. In dit document zal duidelijk worden gemaakt wat mijn taak is geweest, en hoe ik deze heb uitgevoerd. Ook zal worden toegelicht waarom sommige keuzes en stappen zijn gemaakt.

Allereerst zal een inleiding worden gegeven over de afdeling Nucleaire Geneeskunde binnen het UMC. Hierin zal uitgelegd worden welke activiteiten op de afdeling plaatsvinden. Verder is een afdeling tegenwoordig zelf verantwoordelijk voor hun bedrijfsvoering. Er zal dus ook worden uitgelegd hoe de bedrijfsvoering in zijn werk gaat.

In het hierop volgende hoofdstuk zal er uitleg worden gegeven over de achtergrondinformatie. Hierbij wordt onder andere uitgelegd met wat voor stoffen op de afdeling gewerkt wordt, en hoe een scanner hiervan een beeld maakt. Ook wordt (voor zover van toepassing zijne op de opdracht) uitgelegd wat voor (relevante) systemen en programmeertalen binnen de afdeling worden gebruikt.

Na deze twee algemene hoofdstukken zal er meer op de uitgevoerde afstudeeropdracht worden ingezoomd. Allereerst zal de initiële opdrachtomschrijving worden uitgelegd. Na het introducerende hoofdstuk over de opdracht, zal er verder worden ingegaan op het verloop van de opdracht. Hierbij wordt besproken wat de uitdagingen in het proces waren, en hoe deze zijn aangepakt.

Nadat het proces in kaart is gebracht, zal het resultaat in een kort hoofdstuk worden beschreven. Hierin wordt uitgelegd hoe ver de opdracht is gevorderd tot aan de inleverdatum van dit verslag. Ook zal er in dit hoofdstuk een visie op de toekomst worden gegeven, eventueel met aanbevelingen ter verbetering van het framework.

De laatste twee hoofdstukken zijn de literatuurlijst en de evaluatie. De literatuurlijst bevat de bronnen die tijdens de afstudeerstage geraadpleegd zijn, en die in het verslag terug komen. In de evaluatie is beschreven hoe de opdrachtnemer (in dit geval de afstudeerder) het proces heeft ervaren, wat hij heeft bijgeleerd, en wat zijn kijk op het product is.

Ook zijn er nog een aantal bijlagen aan dit document toegevoegd. Hieronder valt het Plan van Aanpak (Project Initiation Document), een aantal onderzoeksdocumenten, de technische en functionele ontwerpen, en het logboek.

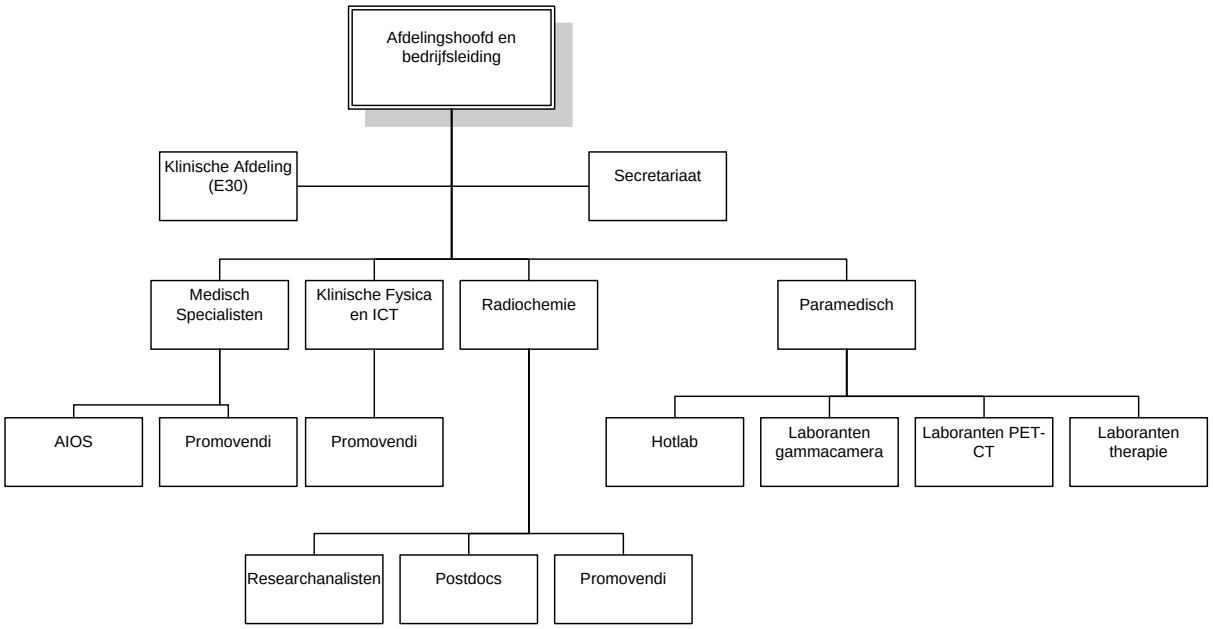
2 Bedrijfsbeschrijving

De afdeling waar de afstudeerstage is uitgevoerd, is de afdeling Nucleaire Geneeskunde van het UMC St. Radboud te Nijmegen. Het Universitair Medisch Centrum St. Radboud is een kenniscentrum voor academische geneeskunde en gezondheidszorg. Naast de zorg voor patiënten wordt er onderwijs verzorgd, en op tal van gebieden onderzoek uitgevoerd. Binnen het UMC St. Radboud zijn 8500 medewerkers, en 3000 studenten werkzaam.

Een van de afdelingen binnen het UMCN is de afdeling Nucleaire Geneeskunde. Deze geeft ondersteuning op gebied van diagnostiek, voornamelijk op het gebied van tumoren. Ook wordt er onderzoek gedaan ter verbetering van nucleaire technieken. Dat onderzoek is voornamelijk gericht op het gebruik van andere stoffen voor het aantonen van verschillende weefsels. Naast diagnostiek en het onderzoek worden patiënten ook behandeld op de afdeling. Bij deze behandeling wordt er bijvoorbeeld Indium-153 of Lutetium-177 bij de patiënt toegediend. Door het inbrengen zit de stof dicht bij het doelweefsel, en wordt het mogelijk gezwellen/tumoren te verkleinen/verwijderen.

2.1 Structuur van afdeling

In onderstaand organigram is de structuur van de afdeling Nucleaire Geneeskunde gegeven. In dit diagram ontbreken de stagiaires. Dit is gedaan omdat binnen een bepaalde subafdeling maar een beperkt aantal stagiaires werkzaam zijn.



Figuur 1: Structuur van de afdeling Nucleaire Geneeskunde

Zelf ben ik als stagiair werkzaam geweest binnen Klinische Fysica en ICT. Het hoofd hiervan is dr. E. Visser (Klinisch Fysicus). Binnen deze subafdeling zijn ook mijn begeleiders werkzaam.

2.2 Kernactiviteiten

Patiëntenzorg

Een van de kerntaken voor de afdeling Nucleaire Geneeskunde is de patiëntenzorg. Binnen de afdeling wordt de diagnostiek uitgevoerd met gamma, PET en PET-CT beelden. Deze dienen als ondersteuning voor andere afdelingen/specialisaties binnen het UMCN. Ook heeft de afdeling een therapeutische taak. Deze betreft (na)behandeling door gebruik van radiofarmaca (uitgelegd in paragraaf 4.1 en 4.2). Hierbij speelt de afdeling E30 een rol, waar de patiënt dan tijdelijk wordt ondergebracht.

Onderzoek

Binnen de afdeling wordt veel wetenschappelijk onderzoek gedaan naar verschillende gebieden binnen de Nucleaire Geneeskunde. Een voorbeeld hiervan is de ontwikkeling van nieuwe radiofarmaca die betere of meer specifieke eigenschappen bezitten. Dit gebeurt op het gebied van diagnostiek (het aantonen van weefsels binnen het menselijk lichaam) of radiotherapie (het behandelen van patiënten door middel van ingebrachte radioactieve stoffen).
Ook wordt gezocht naar samenwerkingen met andere afdelingen. Hierbij kan gedacht worden aan oncologie (medische studie en behandeling van kanker), of instituten/ziekenhuizen buiten het UMC.

Onderwijs

De afdeling heeft regelmatige samenwerking met de Universiteit Twente. Vooral met de curricula Geneeskunde, Biomedische Wetenschappen en Technische Geneeskunde. Dit gebeurt in de vorm van keuzeblokken en masterclasses.
Intern heeft de afdeling meerdere AIOS' in dienst (Arts In Opleiding tot Specialist).
Buiten deze twee onderwijstypes biedt de afdeling (afstudeer)stages aan voor studenten geneeskunde, biomedische wetenschappen, technische geneeskunde, moleculaire levenswetenschappen, MBRT (Medisch Beeldvormende en Radiotherapeutische Technieken), HLO (Hogere Laboratorium Opleiding), HTS en ICT.

3 Achtergrondinformatie

3.1 Werking van tracers

In de nucleaire geneeskunde wordt gebruik gemaakt van een tracer (of radiofarmacon). Hierbij wordt een radionucleïde (ofwel radioactieve stof) gekoppeld aan een stof die in het menselijk lichaam wordt opgenomen. Een voorbeeld hiervan is FDG (Fluorodeoxyglucose). Deze stof wordt door het menselijk lichaam als glucose gezien. Theoretisch gezien is FDG geen glucose, maar is het een molecuul waar een radioactief isotoop in is opgeslagen.

Na toediening zal een tracer (in dit geval FDG) zich over het lichaam verspreiden. We nemen aan dat de persoon die FDG toegediend krijgt, een tumor heeft. Hierbij komt de fysiologische eigenschap van een tumor naar voren, namelijk dat deze een hoog metabolisme heeft. Dit betekent dat in de cellen van de tumor veel stofwisseling plaatsvindt, aangezien de cellen van de tumor energie nodig hebben om te groeien. Deze grote vraag naar energie heeft weer tot gevolg dat er veel glucose naar de cellen wordt getransporteerd. Aangezien het FDG als glucose wordt behandeld in het lichaam, zal ook een verhoogde factor FDG naar deze cellen worden getransporteerd. Dit heeft als gevolg dat er in verhouding meer radioactieve isotopen in het tumorweefsel aanwezig zullen zijn, waardoor meer straling wordt uitgezonden vanuit het tumorweefsel. Er wordt dan gesproken van een hogere uptake.

Doordat de scanners de radioactiviteit kunnen registreren, wordt er een (duidelijke) verhoging gemeten en kan er bepaald worden hoe actief de tumor is, of wat de afmetingen van de tumor zijn. In de volgende paragraaf wordt hierover meer uitgelegd.

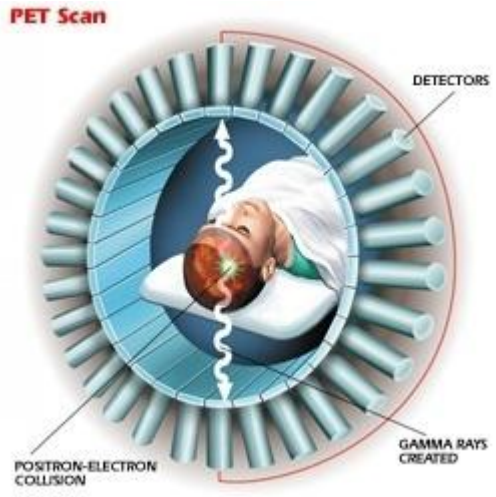
Tracers kunnen ook worden gebruikt als destructief middel tegen kwaadaardige weefsels. Hierbij zal door radioactief verval het weefsel worden aangetast met als gevolg dat (een deel van) het weefsel wordt vernietigd. Dit principe wordt bij de behandeling van bijvoorbeeld de schildklier toegepast. Deze radiofarmaca bezitten namelijk een ander, sterker radioactief isotoop. Aangezien de straling van deze isotoop buiten het lichaam kan komen en de halveringstijd een stuk langer is, zal de patiënt worden opgenomen in het ziekenhuis om mensen in zijn directe omgeving te beschermen.

3.2 Werking PET-scanner

De afkorting PET staat voor Positron Emissie Tomografie. Zoals in de vorige paragraaf uitgelegd bevat de tracer een radionucleïde. Deze komt vrij in het weefsel, en zal hier een kleine afstand afleggen. Zodra deze tot stilstand komt, vervalt deze, en zal er een positron vrijkomen. Dit positron zal met een elektron in aanraking komen, waarna twee gammastralen worden uitgezonden. De gammastralen worden beide in tegenovergestelde richting uitgezonden. Dit is het principe waar de diagnostiek op is gebaseerd.

In een PET-scanner bevinden zich ringen met detectoren. Deze kunnen dergelijke gammastraling opvangen. Als twee gammastralen vrijwel tegelijk door de detectoren worden ontvangen, en deze 180 graden van elkaar liggen, komen deze van hetzelfde positron. Door het tijdsverschil in detectietijd kan worden bepaald waar de gammastralen zijn ontstaan. Dit betekent dus dat bekend is waar het positron zich in het lichaam bevindt.

Doordat de locatie van de positron-elektron botsing bekend is, kan een beeld worden gereconstrueerd. Aangezien de PET-scanner meerdere detectorringen achter elkaar bevat, zal hier informatie in drie dimensies uit ontstaan. Hierbij is opgeslagen hoe vaak een positron in aanraking is gekomen met een elektron, door middel van de gammastraal die ontstaat uit de botsing.



Figuur 2: Het detecteren van de gammastralen

Bij het scannen van patiënten dient altijd een afweging gemaakt te worden tussen scantijd en de dosis. Deze afweging heeft met de ethische en fysiologische gevolgen te maken. Het ethische aspect is dat een scan maken voor de patiënt niet comfortabel is. De gantry (de ronde opening van de scanner) waar de patiënt in wordt geschoven is vrij smal. Het kan voor de patiënt aanvoelen alsof hij klem komt te zitten. Het fysiologische aspect is dat de hoeveelheid radioactieve stof zo laag mogelijk moet blijven, om kans op aantasting voor weefsels van de patiënt te voorkomen. Hierbij ontstaat een afweging aangezien er wel een goede signaal-ruis verhouding moet ontstaan. In het uiteindelijke beeld moet wel te zien zijn wat ruis is, en wat (doel)weefsels van de patiënt zijn. Dit kan verkregen worden door langer te scannen, of door meer radioactieve stof toe te dienen. Hierdoor wordt in de doelweefsels meer opgenomen, en ook meer uitgestraald dan de omgeving. De scanner krijgt dan als gevolg meer detecties in het weefsel in verhouding tot de achtergrond (ruis).

3.3 DICOM afbeeldingen

DICOM staat voor Digital Imaging and Communications in Medicine. Het is een bestandsformaat voor afbeeldingen, waarbij veel extra informatie opgeslagen kan worden. Deze informatie bevindt zich in de headers. Hierin staat patiëntinformatie (naam, geboortedatum, etc.), maar ook scanner instellingen, en instellingen betreffende de opname. Bij opnameparameters kan gedacht worden aan de resolutie van de scanner, de datum van de scan, hoe lang de scan heeft geduurd, de gebruikte radionucleïde, etc.

Binnen één DICOM-bestand kunnen meerdere afbeeldingen worden opgeslagen, hierdoor ontstaat een driedimensionaal array met data.

De derde dimensie van deze array kan de z-as zijn als dit wordt vergeleken met een patiënt die horizontaal op een bed ligt. Hierbij stelt iedere waarde in de z-as een andere afbeelding voor vanuit het hoofd richting de voeten (genaamd een "plane"). Bij het hoofd is de z-as 0, en bij de voeten heeft de z-as een n waarde (n is gekozen omdat niet altijd van het hele lichaam een scan wordt gemaakt, bovendien is het afhankelijk van de plane-dikte die per scanner verschillend kan zijn).

Ook is het mogelijk om de derde dimensie voor tijd te gebruiken (genaamd een "frame"). Hierbij wordt in de header aangegeven hoe lang één tijdsframe duurt. Er kan dan van een bepaalde plak van het lichaam door de tijd gekeken worden hoe deze van intensiteit of omvang verandert. Dit wordt een dynamische scan genoemd, aangezien de tracer zich door het lichaam verspreid/verplaatst. Dit is over meerdere afbeeldingen te zien.

Zodra er gebruik wordt gemaakt van tijd én z-as, wordt er gebruik gemaakt van 4 dimensies. De 3^e of 4^e dimensie zal dan in meerdere afzonderlijke bestanden worden opgeslagen. In praktijk betekent dit dat de z-as, of de tijd wordt gebruikt om een splitsing in aparte bestanden mogelijk te maken. Dit biedt als voordeel dat de bestandsgrootte beperkt kan worden. Applicaties hoeven dan niet de gehele data in te lezen, maar kunnen hier stukken van gebruiken.

Als voorbeeld hebben we een patiënt, waarvan 40 plakken gescand worden, dit gebeurt over 2 minuten waarbij iedere 15 seconden een nieuwe afbeelding wordt gemaakt. 120 seconden delen door 15 seconden houdt in dat er 8 tijdframes zijn. Er zullen dus 320 (40 maal 8) afbeeldingen gemaakt worden, die in 8 DICOM bestanden worden weggeschreven. In ieder DICOM bestand zullen 40 afbeeldingen beschikbaar zijn. In deze situatie bezit ieder DICOM-bestand een momentopname van alle plakken.

Theoretisch is het ook mogelijk om deze twee eigenschappen (z-as en tijd) in de derde dimensie te gebruiken. Hierbij zal in de header gedefinieerd worden hoeveel afbeeldingen over de z-as gemaakt worden in één tijdsframe. Hierdoor kunnen "blokken" worden gemaakt, allemaal met een ander tijdsframe.

In praktijk blijkt de laatste optie minder vaak toegepast te worden. Meestal wordt de keuze gemaakt om de z-as van de patiënt in meerdere bestanden op te slaan, dus bevat elk bestand één plane over alle metingstijden. Een andere optie is om per tijdsframe één bestand te maken. De keuze tussen dergelijke opslagmethodes is per fabrikant verschillend.

3.4 Siemens e.Soft

Siemens levert bij hun PET-scanners ook software om de scanners aan te sturen. Dit softwarepakket wordt samen met de hardware het e.Soft Workstation genoemd. De applicatie van dit framework wordt schermvullend op het Windows besturingssysteem gebruikt. De eindgebruiker heeft verder niets met het Windows systeem te maken, en ziet alleen zijn e.Soft Workstation. Dit Workstation stuurt de scanners aan, en maakt DICOM afbeeldingen van de verkregen data. Deze DICOM bestanden worden dan opgeslagen in de database die aanwezig is op het systeem, of op een netwerkstation.

Verder kan het e.Soft Workstation verschillende studies uitvoeren. Deze studies (analyses) worden uitgevoerd op basis van de DICOM afbeeldingen. Het e.soft systeem is in staat IDL programma's aan te spreken, en worden indien gewenst aan het e.Soft Workstation toegevoegd. Zodra een studie wordt gestart, kan men in een IDL programma de betreffende DICOM afbeelding(en) ophalen. Zodra de applicatie is voltooid, wordt een dicom-bestand teruggegeven aan het e.Soft programma. Binnen het iMed framework worden bijvoorbeeld screenshots gemaakt van de resultaten. Deze worden dan in een dicom-bestand weggeschreven, zodat de afbeeldingen met de resultaten te zien zijn in het e.Soft systeem. Op deze manier worden de resultaten van analyses overzichtelijk bij de patiënt/studie opgeslagen.

3.5 Programmeertaal IDL

IDL is een afkorting voor Interactive Data Language. Deze programmeertaal heeft een syntax die afgeleid is van FORTRAN en C. IDL is ontwikkeld in het Laboratory for Atmospheric and Space Physics op de universiteit van Colorado. Daarna is het in handen gekomen van het bedrijf RSI. Een aantal jaar geleden is RSI overgenomen door ITT Industries, onder de naam ITT Visual Information Solutions (ITT VIS). De kracht van deze taal zit in de beeldverwerking, specifiek de arraybewerkingen.

In IDL is het mogelijk om object georiënteerd te werken. Verder is het mogelijk pointers te gebruiken, en om optionele variabelen in functieaanroepen te gebruiken. Het laatste is zeer handig om broncode van applicaties backwards compatible te houden met oudere versies van IDL, of het eigen framework.

De kracht van IDL zit in de verwerking van arrays. Normaal (3^e generatie programmeertalen) zouden twee loops nodig zijn om een afbeelding in de x en y richting te doorlopen. In IDL is dit niet nodig. De programmeertaal ondersteunt het gebruik van wildcards, waardoor arraybewerkingen zoals optellen, afrekken, delen of vermenigvuldigen met één statement beschreven kunnen worden. De programmeertaal zal dan zelf op een efficiënte manier de array muteren.

Aangezien DICOM afbeeldingen vaak uit driedimensionale (of vierdimensionale) arrays bestaan, is IDL een geschikte taal om berekeningen uit te voeren op deze afbeeldingen. Dit array kan dan ook met een aantal eenvoudige handelingen worden gesplitst, bijvoorbeeld als de tijd of de z-as van de gescande persoon in de derde dimensie staat. Dit maakt de programmeertaal extra geschikt voor medische beeldverwerking. Ook wordt de taal veel gebruikt in de meteorologie en bij de NASA. Dit omdat de taal door zijn vereenvoudigde arraybewerkingen makkelijk in gebruik is om analyses op (grote) afbeeldingen uit te voeren.

Aangezien de programmeertaal zeer specialistisch is, wordt het zoeken naar bruikbare informatie op internet lastiger. In nieuwsgroepen waar vragen worden gesteld over problemen komen veelvuldig dezelfde personen met ideeën aanzetten. Verder zijn de discussies op internet vaak van dermate hoog wiskundig/natuurkundig niveau, dat het voor een HBO-informatica afstudeerder niet te volgen is. Hierdoor wordt al snel duidelijk dat de taal wereldwijd maar door een selecte groep (academici) wordt gebruikt. Het draagvlak voor de taal is dus niet zo groot als bij Java of een van de .NET programmeertalen.

3.6 Object Graphics binnen IDL

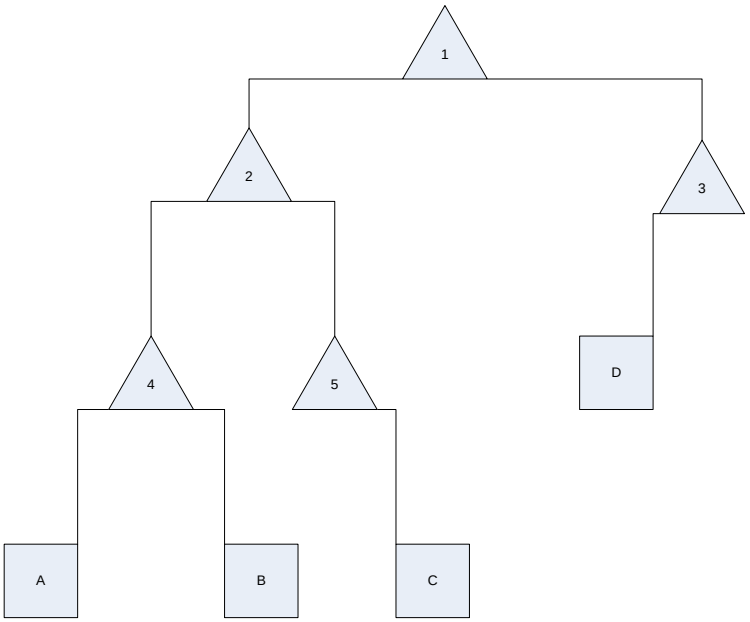
Binnen IDL zijn er twee methodes beschikbaar om grafische objecten weer te geven, namelijk Direct Graphics en Object Graphics. Direct Graphics maakt gebruik van een tekenveld, waar steeds nieuwe informatie over elkaar heen getekend wordt. Het is tevens minder flexibel in gebruik voor het tekenen van afbeeldingen die opgebouwd zijn uit meerdere afbeeldingen/bronnen. Bij Object Graphics is het mogelijk om een complete structuur van verschillende objecten te maken, en om deze vanuit een bepaald perspectief weer te geven.

Object Graphics bestaan uit meerdere objecten. Dit kunnen grafische objecten zijn, of objecten om aan elkaar te koppelen. Dit kunnen onderlinge koppelingen, of koppelingen naar IDL widgets zijn. Deze objecten zijn verbonden met elkaar door een hiërarchische structuur.

Onderaan in de Object Graphics hiërarchie staan de Atomic Graphics Objects (of Graphic Atoms genaamd). Dit zijn objecten die weergegeven kunnen worden. Een dergelijk object kan een polygon, volume, ROI, Image of tekst zijn. Aan deze objecten kan data toegekend worden in de vorm van coördinaten. Verder is het mogelijk om aan deze objecten kleurtabellen, alpha-channels (doorlaatbaarheid) of andere variabelen mee te geven.

Dergelijke Graphic Atoms kunnen gekoppeld worden aan een model. Dit model is een object dat één of meerdere Graphic Atoms of andere modelobjecten bevat. Op dit object kan een rotatie, translatie of schaling worden toegepast. Alle objecten die gekoppeld zijn aan het model, worden dan in de weergave gedraaid. De oorspronkelijke data binnen de Graphic Atom objecten blijft onveranderd.

Door het gebruik van modelobjecten is het mogelijk een boomstructuur te maken. Hierin kunnen objecten gedraaid worden, of kunnen er groepen gemaakt worden die gedraaid worden. In figuur 3 is hiervan een voorbeeld weergegeven. In dit figuur stellen de driehoeken de modellen voor, en de vierkanten de Graphic Atoms. Zodra model 2 een rotatie krijgt, zullen Grapic Atom A, B en C worden gedraaid. Als model 4 wordt geroteerd, zal alleen Graphic Atom A en B in de weergave worden gedraaid.



Figuur 3: Mogelijk voorbeeld van een boomstructuur waarbij de vierkanten de Atomic Graphic Objecten voorstellen, en de driehoeken de Model objecten.

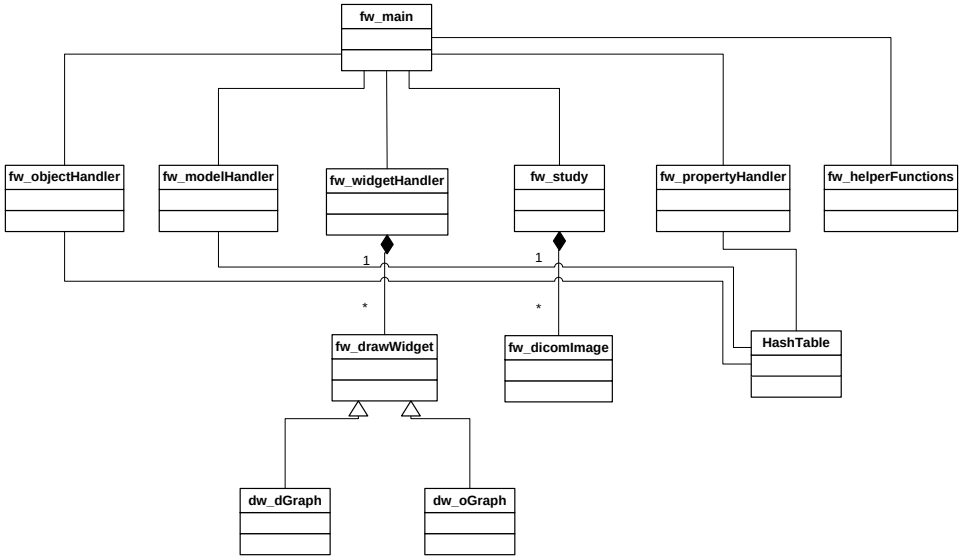
Een dergelijke boomstructuur van modellen en Graphic Atoms kan gekoppeld worden aan een View object. Deze berekent op basis van de informatie uit de modellen en graphic atoms hoe de (mogelijk 3D) informatie moet worden weergegeven.

3.7 iMed framework

Het iMed framework is ontwikkeld binnen de afdeling Nucleaire Geneeskunde van het UMCN. Zoals in paragraaf 3.1 is uitgelegd zijn er twee informatici part-time werkzaam binnen deze afdeling. Door hen worden (studie-)applicaties ontwikkeld die standaard niet aanwezig zijn in het e.Soft systeem. Deze studies worden geschreven in de programmeertaal IDL.

Aangezien dergelijke studies veel overeenkomende functionaliteit bezitten, is de keuze gemaakt om een framework te maken waarbij een bepaalde set aan standaardfunctionaliteiten beschikbaar is. Dit framework wordt het iMed framework genoemd. Binnen het framework is het mogelijk om DICOM bestanden in te lezen, en hier 2D weergaves van te maken. Deze weergaves zijn door de eindgebruiker en programmeur aan te passen. Zo is het bijvoorbeeld mogelijk om uit verschillende kleurtabellen te kiezen. Hierdoor biedt het framework de vrijheid aan de arts/eindgebruiker om voor verschillende studies (of onder verschillende omstandigheden) de kleurtabel te kiezen die op dat moment het meest bruikbaar is.

In deze weergaves is het dan mogelijk om bewerkingen voor berekeningen toe te passen. Een voorbeeld hiervan is het maken van ROI's. De afkorting ROI staat voor Region of Interest. Zodra een arts/specialist informatie uit een deel van de afbeelding wil weten, kan hij een vierkant, ellips of een object in vrije vorm tekenen. Dit object wordt dan over de afbeelding gelegd. Hierdoor kan bepaald worden welke pixels binnen of buiten het object vallen. De pixels die binnen het object vallen, worden dan gebruikt voor berekeningen met betrekking tot (radio)activiteit. Binnen IDL is hiervoor een object beschikbaar. Deze heeft de naam IDLgrROI. Dit object heeft de eigenschap om ROI informatie op te slaan, en tegelijkertijd de weergave ervan te verzorgen. Binnen het framework wordt een IDLgrROI object gekoppeld aan een weergave. Deze weergave bevat tevens de DICOM afbeelding, waardoor berekeningen uitgevoerd kunnen worden op de informatie van de DICOM afbeelding.
Om de werking van het framework duidelijker te maken, staat op de volgende pagina het domeinmodel van het framework (Figuur 4) zoals deze bij aanvang van mijn afstudeerstage was. De namen van IDL klassen beginnen allemaal met "IDL". Eigen klassen uit het framework beginnen met "fw_" of "dw_". Meer informatie over de functionaliteit van de klassen is te vinden in bijlage C, hoofdstuk 1.



Figuur 4: Domeinmodel van het framework bij aanvang van de afstudeeropdracht

Aangezien de vraag is ontstaan om studies in drie dimensies te ontwikkelen, is de keuze gemaakt om het framework met 3D functionaliteit uit te breiden. Deze keuze vormt de directe aanleiding tot de afstudeeropdracht. Met deze 3D functionaliteit zal het mogelijk moeten worden om verschillende weergaves te genereren, en om VOI's te creëren. De afkorting VOI staat voor Volume of Interest. Het principe achter een VOI is het zelfde als achter een ROI. Het verschil tussen een VOI en een ROI zit in het aantal dimensies. Een ROI bestaat uit twee dimensies, terwijl een VOI uit drie dimensies bestaat. Hierdoor is een VOI geschikt om over een 3D dataset (mogelijk DICOM afbeeldingen) te leggen, en hiermee berekeningen uit te voeren.
De exacte omschrijving van de afstudeeropdracht wordt beschreven in het volgend hoofdstuk (hoofdstuk 4, Opdrachtomschrijving).

4 Opdrachtomschrijving

Voor onderzoek of behandeling kan het mogelijk zijn dat nieuwe studie-applicaties geschreven moeten worden. Met het iMed framework is het mogelijk om deze applicaties in 2D te schrijven. Hierbij worden afbeeldingen vanuit één aanzicht gebruikt voor weergave.
Aangezien berekeningen en/of weergaves vanuit andere aanzichten de mogelijkheid bieden om meer of andere informatie te onderzoeken, ontstaat hieruit de vraag om het iMed framework met 3D visualisatie uit te breiden. Dit maakt het mogelijk om vanuit andere aanzichten naar de afbeeldingen te kijken, en waarbij een wijziging automatisch wordt doorgestuurd naar de andere aanzichten. Verder wordt het mogelijk om volume-berekeningen toe te passen.

In de volgende paragrafen van dit hoofdstuk wordt omschreven wat de oorspronkelijke opdrachtomschrijving voor het afstuderen was. Deze opdracht is onder te verdelen in drie iteraties, namelijk het weergeven van 3 verschillende 2D aanzichten van objecten, het tekenen van Volumes of Interest op deze 3x 2D weergave, en het genereren van een 3D afbeelding.

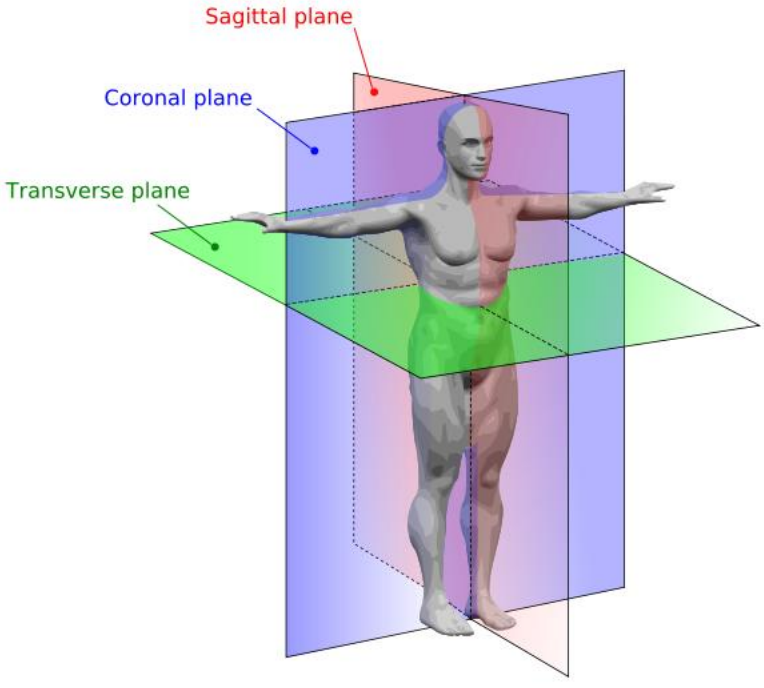
4.1 3x 2D weergave

De eerste iteratie bestond uit het weergeven van 3 verschillende 2 dimensionale aanzichten. Hierbij wordt het mogelijk om vanuit verschillende aanzichten naar de afbeelding te kijken. Dit geeft voor de arts een duidelijker beeld over de vorm van een tumor/weefsel. De drie aanzichten die hierbij worden gebruikt zijn de transversale, coronale, en sagittale aanzichten.

Het transversaal aanzicht houdt in dat er (bij een persoon) van boven wordt gekeken. Hierbij wordt virtueel de persoon in horizontale plakken gesneden, waarbij van bovenaf iedere plak bekeken wordt. Het transversaal aanzicht zou vergeleken kunnen worden met het bovenaanzicht.

Bij het coronaal aanzicht worden plakken gemaakt die van voren naar achter bij een persoon lopen. Deze plakken gaan over de lengte van de persoon. In eerste instantie zal alleen een neus zichtbaar zijn, en als men verder naar achter gaat komt men de ogen tegen, later zijn de oren nog zichtbaar, en zo voorts. Het coronaal aanzicht zou vergeleken kunnen worden met het vooraanzicht

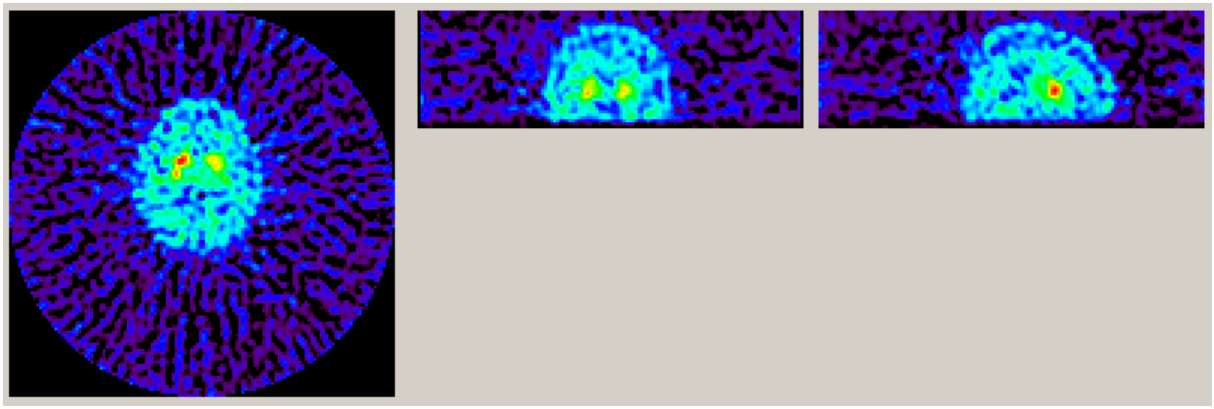
Wordt de patiënt van de zijkant getekend, dan spreekt men van het sagittaal aanzicht. Hierbij wordt de persoon ook weer in plakken over de verticale as verdeeld, maar loopt het nu van links naar rechts. Eerst zal de rechter arm zichtbaar zijn in plakken, daarna het lichaam zelf, en als laatste de linker arm. Het sagittaal aanzicht zou vergeleken kunnen worden met het zijaanzicht.



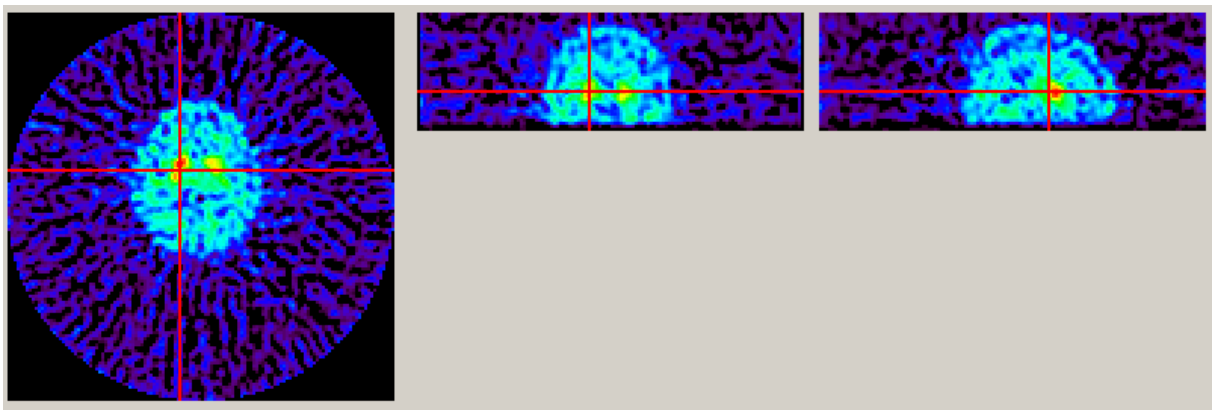
Figuur 5: De verschillende soorten aanzichten ten opzichte van het menselijk lichaam

Deze aanzichten moesten aan elkaar worden gekoppeld, waardoor met lijnen in het ene aanzicht aangegeven kon worden waar het andere aanzicht zich bevind. Als iemand in het transversaal aanzicht een ander plane zou selecteren, moest en de lijnen in het coronaal en transversaal aangepast worden zodat deze de juiste locatie zouden aangeven.
Verder moest het ook mogelijk zijn dat doormiddel van het verplaatsen van lijnen, de planes in de aanzichten zouden veranderen. Dit betekend dat als de gebruiker in het coronaal aanzicht de lijn van het sagittaal aanzicht verplaatst, de afbeelding in het sagittaal aanzicht verplaatst moet worden naar het gekozen plane.

Door het gebruik van deze drie weergaves, met (indicatie)lijnen van de andere weergaves, kan een Nucleair Geneeskundige een beeld krijgen van het weefsel dat veel straling heeft uitgezonden. Zonder deze lijnen zou een Nucleair Geneeskundige moeten gokken welke plak wordt weergegeven, omdat dit in de andere aanzichten niet altijd duidelijk is (zie figuur 6 en 7)



Figuur 6: De verschillende aanzichten zonder indicatielijnen (v.l.n.r. transversaal, coronaal en sagittaal)



Figuur 7: De verschillende aanzichten met indicatielijnen (v.l.n.r. transversaal, coronaal en sagittaal)

4.2 Volume of Interest

Doordat de eerste iteratie het mogelijk heeft gemaakt om de drie dimensies weer te geven (door gebruik van verschillende aanzichten), is het mogelijk geworden om hierop berekeningen uit te voeren. Een arts (nucleair geneeskundige) wil bijvoorbeeld weten hoe groot een bepaald weefsel is, of hoe actief dit weefsel is (in termen van een hogere uptake). Hiervoor moet het mogelijk zijn om in drie dimensies een stuk te selecteren. Wanneer een oppervlak in twee dimensies wordt aangeduid, wordt het een ROI (Region of Interest) genoemd. Deze functionaliteit was reeds beschikbaar. In drie dimensies wordt er niet meer over een oppervlakte, maar over een volume gesproken. Dit wordt dan een Volume of Interest (afgekort VOI) genoemd.

Een Volume of Interest is in feite een kubus of een ander drie-dimensionaal voorwerp. Dit voorwerp wordt op een bepaalde coördinaat binnen de afmetingen van de dicom-afbeelding gezet. Aangezien de dicom-afbeelding een kubus is met pixel/voxel coördinaten (een voxel is een pixel in drie dimensies), kan met de VoI-informatie worden bepaald welke pixels van de DICOM-afbeelding binnen het volume vallen. Deze pixels/voxels worden dan meegenomen bij het bepalen van de activiteit.

Voor de arts betekend een dergelijk volume meer informatie, waardoor de berekeningen en conclusies betrouwbaarder worden. Zonder een volume of interest, in combinatie met de 2x 3D weergave zal een arts op ieder plane een ROI moeten tekenen om informatie van een volume te berekenen. Dit levert meer kans op onnauwkeurigheden. De arts kan op plane 25 een te grote ROI tekenen, terwijl op plane 26 een te kleine ROI staat, die ook deels buiten het doel valt. De kans op deze onnauwkeurigheden worden door een VOI verkleind aangezien het mogelijk wordt om op één plane een vorm te tekenen, en deze uit te trekken over meerdere planes.

4.3 3D afbeelding genereren

Als de eerste twee iteraties zijn gerealiseerd is de vervolgstap om een afbeelding te genereren dat een 3-dimensionaal volume kan weergeven. Deze afbeelding maakt het mogelijk om in een 3-dimensionale weergave de DICOM informatie te projecteren. Deze afbeelding kan door de eindgebruiker worden geroteerd, gezoomd of worden verplaatst, zodat deze kan bepalen vanuit welke hoek hij naar de afbeelding wil kijken. Eventueel is het mogelijk om de gedefinieerde VOI's hierin weer te geven. Hierdoor zal de arts eenvoudiger de positie van de tumor kunnen lokaliseren binnen het lichaam. Dit komt weer ten goede op het gebied van behandeling.

5 Uitvoering en onderzoeken

In dit hoofdstuk wordt omschreven hoe het project is uitgevoerd, en worden de gemaakte keuzes tijdens het proces toegelicht en verantwoord. In de eerste drie paragrafen wordt de gang van zaken per iteratie beschreven. Daarbij zal telkens naar voren komen hoe de uitvoering is gegaan ten opzichte van de planning, en welke uitdagingen naar voren zijn gekomen.

In de daarop volgende paragrafen worden een aantal kleine onderzoeken en bevindingen beschreven die tijdens de afstudeerperiode zijn uitgevoerd of ondervonden. Deze onderzoeken hebben allemaal een relatie met de opdracht, waarbij het resultaat is meegenomen in de ontwikkeling van de uitbreiding op het iMed framework.

5.1 De startfase en uitvoering iteratie 1

In de startfase is de tijd genomen om een goed plan van aanpak te maken en om zelf bekend te worden met de programmeertaal IDL en het iMed framework. Iteratie 1 had als doel om een weergave te maken met 3 vensters, waarbij deze vensters allemaal één aanzicht representateren. In de andere aanzichten moest het mogelijk gemaakt worden om het huidige plane zichtbaar te maken met een lijn. Verder moest het mogelijk zijn om deze lijn te verplaatsen, waardoor de andere aanzichten automatisch werden bijgewerkt
Aangezien deze twee fases (startfase en iteratie 1) overlap met elkaar hadden (vooral op het gebied van kennismaking met IDL en het framework), worden beide fases in deze paragraaf besproken.

Allereerst is het project gestart met een verkenning van de programmeertaal IDL. Voor mijn afstudeerstage was de programmeertaal bij mij niet bekend. Zoals in paragraaf 4.5 beschreven heeft IDL ook geen grote gebruikersgroep. In deze eerste twee weken ben ik dan ook vooral bezig geweest met het leren van de syntax, en om zelf (test)applicaties te schrijven om te kijken of mijn interpretatie van de taaleigenschappen juist was.

Aangezien dit vrij soepel verliep, ben ik aan het einde van week 2 aan de inhoud van het Plan van Aanpak (Project initiation Document, bijlage A) verder gegaan. Dit plan van aanpak heb ik ook naar mijn begeleiders gestuurd voor feedback, zodat het document in week 3 naar de docentbegeleider opgestuurd kon worden. Tijdens de afronding van het plan van aanpak (in week 3) is de start gemaakt met iteratie 1. Allereerst is hiervoor een functioneel ontwerp gemaakt. Dit volgde door een technisch ontwerp en eindigde met de realisatie van de iteratie. Aan het begin van deze iteratie (week 4) werd door de bedrijfsbegeleiders de huidige versie van het framework beschikbaar gesteld. Het eerste idee hierbij was om een aantal overervingen van klassen te schrijven, terwijl de bedrijfsbegeleiders meer het idee hadden om bestaande klassen uit te breiden. Hierover zijn een aantal gesprekken gevoerd, en is de conclusie getrokken dat het beter was om de klassen uit te breiden. De argumentatie hierbij was dat het een framework uitbreiding betreft, en geen uiteindelijke applicatie. Het framework moet geen wildgroei aan klassen krijgen zodat het overzicht verloren gaat. Door een wildgroei wordt het lastiger te bepalen welk object het meest geschikt is voor een bepaalde implementatie. Verder bevat het framework al een klasse voor de weergave van dicom-afbeeldingen. Deze klasse maakt het al mogelijk om weergave-objecten aan elkaar te koppelen. Door dit uit te breiden, zou ik ook mijn nieuwe functionaliteit kunnen herbergen. Dit is dan ook in het Technisch Ontwerp verwerkt.

Hierbij ontstond een (tot op dat punt onbekend) probleem wat voorheen nog niet aanwezig was, namelijk het wel of niet cachen van de dicom-bestanden. In deze bestanden is de afbeeldinginformatie in drie dimensies opgeslagen, en wordt deze als een 3D-array binnen IDL gebruikt. Bij het weergeven van verschillende aanzichten is het nodig om de gehele array in het geheugen beschikbaar te hebben. Aangezien deze arrays best groot kunnen worden, was het de vraag of het verstandiger is om de dicom-bestanden van tevoren te cachen in de verschillende aanzichten. Hierbij zouden meerdere arrays in het geheugen worden bewaard. Deze arrays representeren dan één aanzicht, om op die manier sneller informatie weer te geven.

Om te kijken of het cachen snelheidsvoordeel had heb ik een aantal tests uitgevoerd met arrays die wel en niet werden gecached. De resultaten hiervan staan in bijlage B, in het hoofdstuk "Snelheid in Arraybewerkingen". Met deze resultaten ben ik naar mijn begeleiders gestapt. Zelf had ik als conclusie hieruit genomen dat het "on the fly" verwerken een betere oplossing was. Dit omdat het vooraf cachen zeer lang duurt. Bij een dicom-set bestanden die als normaal wordt beschouwd duurde het 23 seconden voor één aanzicht te berekenen. Met deze conclusie waren de bedrijfsbegeleiders het eens, en de keuze werd gemaakt om "on the fly" verwerking te implementeren.

De verdere realisatie van iteratie 1 verliep zeer vlot. Daardoor kon vóór de ingeplande datum aan iteratie 2 worden begonnen.

5.2 Uitvoering iteratie 2

Iteratie 2 had als doel om Volumes of Interests te creëren binnen het framework. Aangezien bij iteratie 1 de mogelijkheid was gemaakt om afbeeldingen in drie aanzichten weer te geven, was het creëren van VOI's de meest logische volgende stap. Een Volume of Interest is namelijk een 3-dimensionaal volume waar berekeningen op uitgevoerd kunnen worden. Om deze volumes te maken zijn er meerdere aanzichten nodig. Allereerst is hiervoor een Functioneel Ontwerp gemaakt. Voor dit Functioneel Ontwerp is het document uit iteratie 1 als basis gebruikt. Hierop heb zijn aanpassingen/toevoegingen gemaakt die betrekking hadden op de gevraagde functionaliteit van iteratie 2.

Na het Functioneel Ontwerp was de volgende stap het maken van het Technisch Ontwerp. Hier zat de grootste uitdaging voor iteratie 2. Er waren namelijk een paar uitdagingen die gaandeweg zijn ontstaan tijdens het opzetten en uitwerken van het Technisch Ontwerp. De twee belangrijkste vragen die hierbij ontstonden waren "hoe kunnen VOI's het beste worden opgeslagen?" en "hoe kunnen VOI's het beste worden weergeven in de drie verschillende aanzichten?".

5.2.1 Opslag van Volume of Interests

In twee dimensies heeft IDL een goed systeem voor het opslaan en weergeven van Regions of Interests. Het object dat de data bevat voor het rekenen met deze ROI's biedt tevens ook de mogelijkheid om de weergave te presenteren in een Object Graphics venster. Hierdoor is een programmeur er zeker van dat het getekende figuur ook de juiste data achter de schermen bevat.

Voor drie dimensies heeft IDL geen "out-of-the-box" systeem voor het intern opslaan en weergeven van Volume of Interests. De klasse die ROI's beheert en representeert, kan alleen overweg met 2-dimensionale data. Alle berekeningen van deze klasse zijn dan ook gericht op 2-dimensionale data.

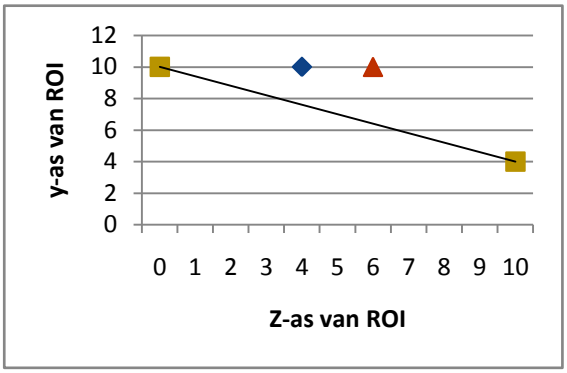
Door het "gebrek" aan een bruikbaar 3-dimensionaal object, moest er gekeken worden naar een aantal opties om hier omheen te werken. Allereerst zouden er allerlei IDLgrROI objecten gemaakt kunnen worden, die allemaal een gemeenschappelijke variabele hadden. Door aanpassingen op een ROI te maken die de gemeenschappelijke variabele zou bezitten, zou het mogelijk zijn het "volume" aan te passen. Hierbij zou voor iedere actie betreffende ROI's of volumes in het framework (denk aan verplaatsen, schalen, roteren, tekenen) gecontroleerd moeten worden of het één ROI was, of dat het een verzameling van meerdere ROI's (een VOI) waren.

Deze optie was niet toepasbaar, aangezien hiervoor erg veel administratieve aanpassingen in het framework nodig waren. Dat zou de onderhoudbaarheid van het framework flink aantasten. Verder zou de weergave in andere aanzichten voor veel problemen zorgen.

Een andere optie was om een container te maken, die alle ROI objecten bezit. Hiervoor is binnen IDL een object beschikbaar, genaamd IDLgrROIGroup. Deze klasse bezit tevens de functionaliteit om in 3 dimensies berekeningen uit te voeren over de ROI's binnen de groep.
Wel was er één nadeel aan deze klasse tijdens het schalen van ROI's. Dit schalen houdt in dat de grootte van de ROI's wordt aangepast, of dat de ROI's verder van elkaar af, of dichter bij elkaar worden geplaatst.

Als er twee ROI's in een IDLgrROIGroup achter elkaar staan, en deze worden geschaald in de z-richting van de ROI, worden de ROI's alleen verder uit elkaar gezet, maar wordt er géén opvulling voor de lege plaats gerealiseerd.
Dit zorgt voor een probleem bij de berekening van de activiteit binnen een volume. Hieronder wordt dit probleem gedemonstreerd met de functie containsPoints. Deze functie berekent of een punt binnen, of buiten een IDLgrROIGroup valt.

Als voorbeeld nemen we 2 ROI vlakken die achter elkaar staan. Door middel van de methode Scale wordt de positie van beide ROI's aangepast. Beide afbeeldingen beginnen op [0,0,z] waarbij de eerste ROI een z van 0 heeft, en de tweede ROI een z van 10. Verder heeft de eerste ROI een oppervlakte van 10x10 pixels, en heeft de tweede ROI een oppervlakte van 4x4 pixels. In de grafiek hiernaast staan de z- en de y-richting van dit verhaal uitgewerkt.



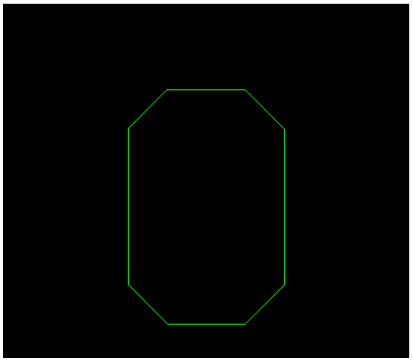
Zodra met containsPoints wordt opgevraagd of bij z=4 de hoogte 10 is, zal deze Juist terug geven. Als de hoogte lineair zou aflopen, zou dit false moeten zijn, aangezien deze de hoogte 7,6 zou moeten hebben vanwege de volgende lineaire berekening: $P1_Y-(((P1_Y-P2_Y)/(P2_Z-P1_Z))*(n-P1_X))$. In de grafiek wordt dit weergegeven met het blauwe ruitje.
Als met containsPoints wordt opgevraagd of z op 6 de hoogte 10 heeft, zal deze methode False terug geven (weergegeven met het rode driehoekje). Dit zou voor mijn implementatie wel juist zijn.
Dit probleem ontstaat doordat containsPoints op zoek gaat naar het dichtstbijzijnde punt wat deze kan vinden in de beschikbare regions. Voor z=6 betekend dit dat punt 2 (z=10) het dichtst in de buurt is. Voor z=4 betekend dit dat punt 1 (z=0) het dichtst in de buurt is. Daarop wordt dan gebaseerd of een punt binnen of buiten de ROIGroup valt. Voor het beoogde gebruik is containsPoints niet toereikend om een ROI te tekenen op de start-dimensie, en een ROI op de eind-dimensie. De tussenliggende lege dimensies moeten opgevuld worden om deze methode zijn werk goed te laten doen.

Door deze bevindingen is de keuze gemaakt om IDLgrROIGroup toch te gebruiken, maar door middel van overerving de schalingsmethode te wijzigen. Hierdoor kunnen de lege z-locaties opgevuld worden met de beoogde relevante informatie. Zou dit niet opgevuld worden, dan was het niet mogelijk om de 3D functies van IDLgrROIGroup te gebruiken vanwege het "ontbreken" van de nodige informatie. De methodes van IDLgrROIGroup gaan zelf geen ROI-planes aan elkaar koppelen om op die manier een volume te creëren. Door de lege z-locaties op te vullen, is het wel mogelijk om berekeningen zijnde een volume uit te voeren.

5.2.2 Weergave van een VOI

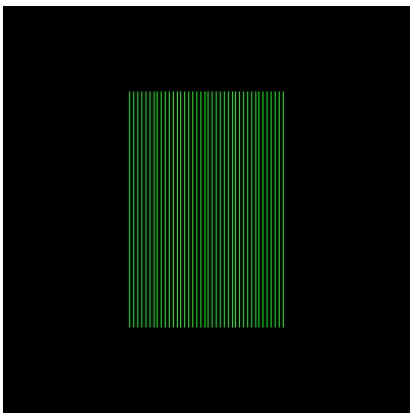
Zoals in de vorige paragraaf is uitgelegd bestaat een Volume of Interest uit meerdere Regions of Interest die in een container bij elkaar worden opgeslagen. In de vorige paragraaf is beschreven hoe de data opgeslagen kan worden, maar wat betreft de weergave blijft er echter nog een grote uitdaging over. Hoe is het mogelijk om dergelijke informatie in een ander aanzicht weer te geven?

IDLgrROIGroup is een Atomic Graphics Object (zie paragraaf 3.6). Hierdoor is het mogelijk een grafische weergave van het IDLgrROIGroup object te maken.
De weergave van het IDLgrROIGroup object bleek helaas niet van toepassing te zijn voor de gekozen implementatie, aangezien deze niet het gewenste beeld gaf bij de verschillende aanzichten. Als in het transversaal aanzicht een figuur getekend wordt, zou de VOI hiervan alleen in het transversaal venster als kader zichtbaar zijn (zie figuur 8). In het sagittaal aanzicht (zij aanzicht) zou er maar één lijn zichtbaar zijn, aangezien de region nog geen volume is, en (nog) geen diepte bezit.



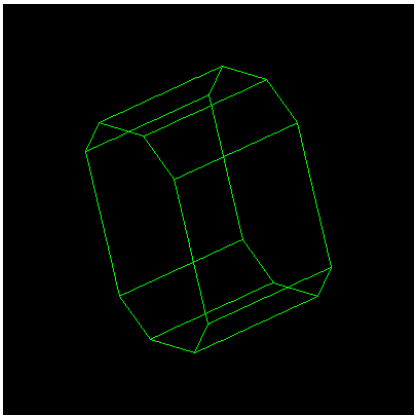
Figuur 8: IDLgrROIGroup presentatie vanuit teken-aanzicht (in dit geval transversaal)

Zodra deze ROI sagittaal wordt uitgetrokken (lees: er worden extra ROI's in de z-richting gemaakt), worden er meerdere lijnen achter elkaar getekend en ontstaat er een VOI (zie figuur 9). Hierdoor zou er in het sagittaal en coronaal aanzicht meerdere rechte lijnen zichtbaar zijn. Deze lijnen representeren de randen van de ROI's. In theorie klopt dit met de opslag van data, maar is dit niet wenselijk in de praktijk. De arts denkt namelijk aan één volume, en niet aan meerdere 2-dimensionale vlakken. In feite zal de illusie gecreëerd moeten worden dat er maar één object bestaat.



Figuur 9: IDLgrROIGroup presentatie vanuit zij-aanzicht (in dit geval sagittaal)

Dit probleem was op te lossen door de datapunten uit de verschillende ROI's van IDLgrROIGroup te gebruiken. Als men deze punten gebruikt om lijnen te tekenen, is het mogelijk om een visueel beeld te maken van het object. Hiervoor kunnen objecten van de klasse IDLgrPolyline gebruikt worden. Aan dit object kan een lijst met punten, en een verbindingslijst worden meegegeven. Door deze verbindingslijst wordt bepaald welke punten aan elkaar verbonden dienen te worden. Hiermee lukte het om een 3D-weergave van het object te maken (zie figuur 10). Ook is het mogelijk om op basis van deze punten de randen van de VOI in de verschillende aanzichten te bepalen. Hiermee kan dan een 2-dimensionaal aanzicht worden gemaakt op basis van de 3D informatie.



Figuur 10: IDLgrROIGroup presentatie door middel van IDLgrPolyline waarbij de knooppunten en verbindingslijst zijn samengesteld uit de informatie van IDLgrROIGroup

5.2.3 2D representatie VOI

Zoals uit de vorige paragraaf blijkt, is het mogelijk om met behulp van de originele datapunten de randen van een VOI in de verschillende aanzichten te bepalen. De vraag hierbij is vanuit welk aanzicht de vorm van de VOI is getekend? Bij een transversaal aanzicht zou de vorm in het x-y vlak worden getekend, terwijl bij een sagittaal aanzicht de vorm in het x-z vlak wordt getekend. Bij een coronaal aanzicht zou de vorm in het y-z vlak worden getekend.

Om te bepalen of de ROI's de x, y of z-dimensie hebben gebruikt voor de diepte is erg lastig. Daarom is er gekozen om bij een VOI op te slaan welke dimensie de diepte van de VOI vertegenwoordigd. Verder zullen voor het sagittaal en coronaal aanzicht de 2-dimensionale tekencoördinaten hervormd moeten worden. De coördinaten die aan het ROIGroup object worden gegeven zijn x-y-z voor het betreffende aanzicht, terwijl dit in de VOI opgeslagen moet worden als x-z-y of z-x-y. Dit is van belang bij het rekenen met de originele data. Als iemand wil controleren of een bepaalde pixel/voxel binnen of buiten het volume valt, zullen de coördinaten wel met elkaar overeen moeten komen.

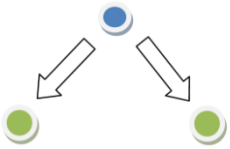
Verder zat er een grote uitdaging in de 2D representatie van het Volume of Interest. De representatie in het vlak waar de ROI is getekend, is het eenvoudigst.. De betreffende ROI wordt opgehaald uit de groep, en de punten van deze ROI worden gebruikt voor de representatie. Deze punten kunnen netjes achter elkaar worden doorverbonden.

De opbouw van de andere twee aanzichten wordt wel lastiger. Allereerst dient gekeken te worden welk aanzicht het "doel" is. Van daaruit wordt bepaald welke dimensie doorzocht moet worden, en wat de waarde van deze dimensie moet zijn (het plane-nummer).

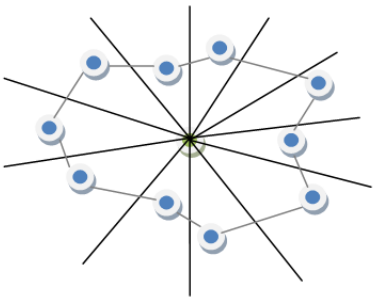
Voorbeeld
Als voorbeeld wordt het coronaal aanzicht genomen. In dit aanzicht is (vanuit originele x-y-z specificatie) de 2e dimensie (y) de diepte. Als vanuit het coronaal aanzicht plane 9 wordt gekozen, betekent dit dat in alle ROI's gezocht moet worden naar knooppunten waarbij de 2e dimensie de waarde 9 heeft.

Dit is nog niet de grootste uitdaging. De grootste uitdaging ontstaat bij het koppelen van de gevonden punten. Dit zal in de juiste volgorde moeten gebeuren. Zodra dit niet gebeurt zullen er vormen ontstaan die een arts/gebruiker niet heeft getekend, hierdoor wordt voor het gevoel van de arts/gebruiker de betrouwbaarheid van het systeem beschadigd.

Allereerst was het idee om deze punten te koppelen door een kortste afstand te berekenen. Hierbij zou per knooppunt gekeken worden welk knooppunt het dichtst bij lag. Tussen deze twee punten zal dan een lijn getrokken worden. Dit bleek niet mogelijk aangezien er wel een willekeurig startpunt gekozen kon worden, maar het niet uit de knooppunten te halen was wat het eindpunt was. Verder neemt de kortste route ook niet alle punten mee, terwijl dat bij deze implementatie wel wenselijk is. Ook kwam het voor dat punten dezelfde afstand hadden. Hierdoor kon ook geen keuze gemaakt worden welk punt verbonden moest worden (zie figuur 11).



Figuur 11: Situatie waarbij de groene punten allebei dezelfde afstand tot het blauwe punt hebben



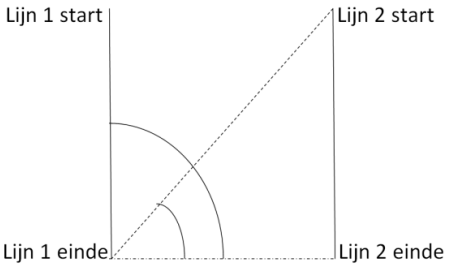
Figuur 12: Knooppunten verbinden op basis van radialen, berekend vanuit het centrum van de knooppunten

De volgende poging was op basis van een circel. Hierbij werd het middelpunt van alle punten berekend. Daarna werd op basis van middelpunt en knooppunt bepaald wat de hoek was ten opzichte van een loodrechte, verticale lijn door het middelpunt. Door de lijst van hoeken te sorteren op grootte, was het mogelijk een lijn te trekken door de verschillende punten (zie figuur 12).
Dit blijkt helaas niet te werken met bijvoorbeeld U-vormen, S-vormen of 8-vormen. Hierbij komen er meerdere punten op één hoek-afstand, en is het lastig te bepalen welk punt verbonden moet worden.

De laatste poging was op basis van hoeken. Allereerst werd de lijst met punten gesorteerd. Eerst op ROI-dimensie, en daarna op array-index. Deze array-indexen komen van de ROI-data, en staan in de volgorde van creatie. In ieder plane zit dus een x-y waarde met een bepaalde index. Als in de volgende ROI de x-y waarde wordt opgehaald op dezelfde index, blijkt de x-y waarde (niet de ROI-dimensie) hetzelfde. Dit komt door het principe dat een ROI gedupliceerd wordt om een VOI te krijgen. Alle x-y waarden worden dan gekopiëerd, met uitzondering van de z-waarde.
Door deze sortering als verbindingslijst toe te passen, ontstaan er meerdere lijnen van boven naar beneden. Door tussenliggende "lijnen" uit de lijst te laten, is het mogelijk een vierkant te maken dat het zij-aanzicht presenteert.

Aangezien al deze lijnen van boven naar beneden lopen, is het niet mogelijk deze aan elkaar te verbinden. Dit geeft een soort van N-figuren. Hiervoor is een oplossing bedacht op basis van hoeken. Als het volgende punt een kleinere hoek heeft dan het eindpunt, dient de volgorde van de tweede lijn omgedraaid te worden.

Voorbeeld
We nemen hierbij lijn 1 als linker lijn, en lijn 2 als rechter lijn. Het einde van lijn 1 en lijn 2 vormt samen met het begin en einde van lijn 1 een hoek van 90 graden. Het einde van lijn 1 en het begin van lijn 2, samen met het begin en einde van lijn 1 zal altijd een kleinere hoek geven (zie figuur 13).



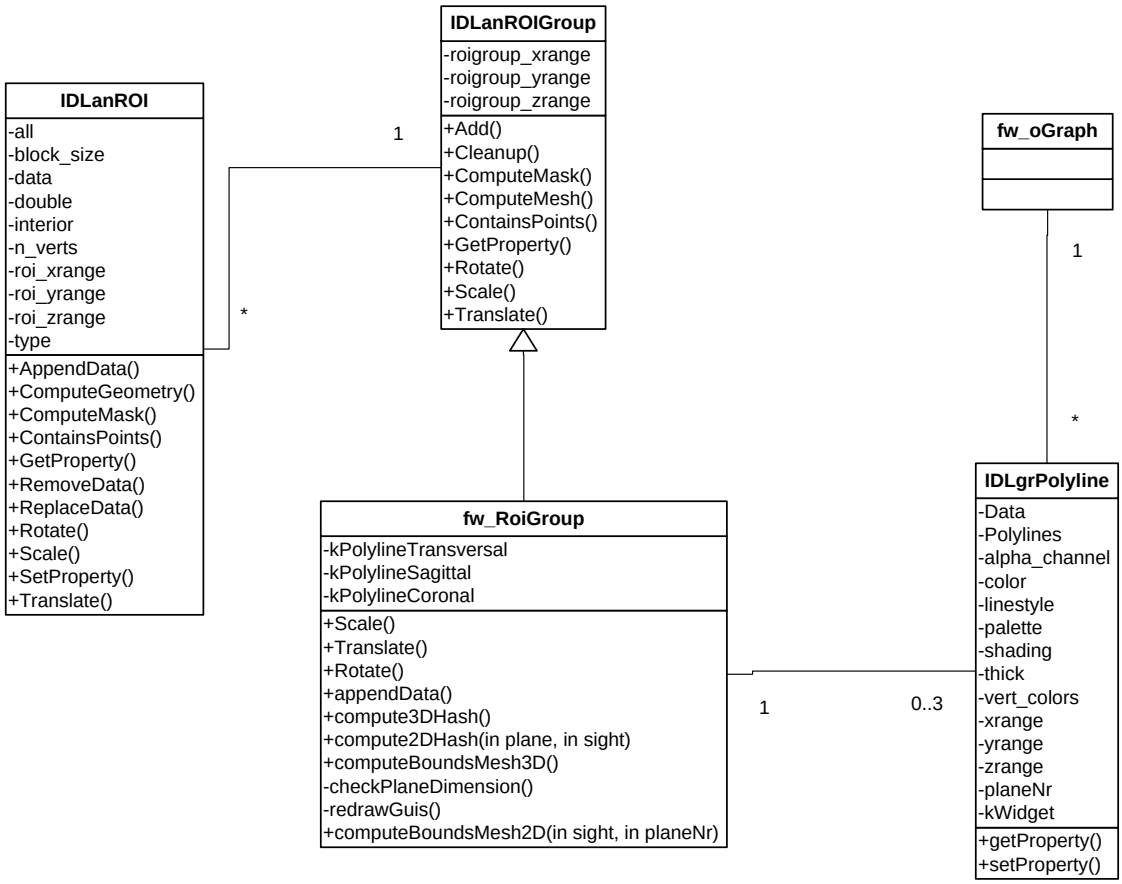
Figuur 13: Berekenen van de hoek

5.2.4 Uitwerking binnen het framework

Door deze bevindingen ben ik bij een technisch ontwerp uitgekomen waarbij 1 object wordt gebruikt voor het bijhouden van de data (een ROIGroup), en waaraan 3 grafische objecten (van het type IDLgrPolyline) worden gekoppeld. Dit object heb ik laten overerven van IDLanROIGroup, en heeft de naam fw_RoiGroup gekregen. De klasse IDLanROIGroup heeft dezelfde functionaliteit als IDLgrROIGroup, met uitzondering van de grafische presentatie. Hier heb ik een aantal methodes toegevoegd, en een aantal methodes overruled. Hierdoor beheert het nieuwe object de knooppunten, maakt deze de verbindingslijsten, en beheert/update deze de polyline objecten. Zodra er iets word verandert aan de VOI, zal er een aanroep naar het fw_RoiGroup object plaatsvinden. Deze zal intern acties ondernemen en de polyline objecten (grafische objecten) bijwerken. De vensters waar deze polyline-objecten zich op bevinden zijn lid van een "aanzichts-collectie". Hierdoor is het bekend welke vensters een link met elkaar hebben, zodat ze elkaar kunnen updaten als de informatie voor de polyline-objecten is aangepast.

Op de volgende pagina staat een diagram dat illustreert hoe de uitbreiding er uit ziet. Hierbij wordt in het IDLgrPolyline een referentie geplaatst naar het RoiGroup object, en kan het IDLgrPolyline object aan de weergave gekoppeld. Door de koppeling aan de weergave is het polyline-object ook aan te roepen binnen het framework. Het RoiGroup object zal niet direct worden aangeroepen, aangezien deze niet gekoppeld kan worden aan een event vanuit de GUI. Het polyline-object kan wel aan een event worden gekoppeld.
Door via het polyline object het RoiGroup object op te halen, is het mogelijk aanpassingen aan de VOI (RoiGroup) uit te voeren. Deze implementatie is gekozen aangezien grafische objecten uit de view geselecteerd kunnen worden. In dit geval is dat het IDLgrPolyline object. Het ROIGroup object is niet beschikbaar als Graphic Atom, en kan dus niet visueel worden geselecteerd.

Het technisch ontwerp met de beschreven uitwerking heb ik voorgelegd aan de bedrijfsbegeleiders. Aanvankelijk was het de bedoeling om ook het IDLgrPolyline over te erven en deze te gebruiken als interface voor het fw_RoiGroup object. Dit zou ervoor zorgen dat hierdoor minder aanpassingen nodig waren aan het huidige framework, aangezien fw_RoiGroup niet direct aan te roepen is.
In de feedback van het Technisch Ontwerp kwam naar voren dat het niet netjes was om een grafisch object te gebruiken om allerlei informatie door te sturen die indirect nodig was voor dat object. Het polyline-object zal zelf niets met die data doen, en object heeft als doel om een weergave te creëren voor een bepaald aanzicht. In theorie is dit wel juist. Alleen was hiervoor gekozen zodat de klasse als soort van interface zou fungeren. Dit zou het aantal aanpassingen in het huidige framework minimaliseren. Na een korte inventarisatie en overleg met de bedrijfsbegeleiders bleek dat het aantal aanpassingen wel meeviel, en dat de functionele netheid boven het aantal code-aanpassingen ging. Hierdoor werd een overerving van het polyline-object overbodig. Het zou namelijk geen extra functionaliteit nodig hebben.
Uiteindelijk is het systeem ook geïmplementeerd zoals het in figuur 14 wordt geïllustreerd.



Figuur 14: Klassendiagram voor uitbreiding van het framework

5.3 Uitvoering iteratie 3

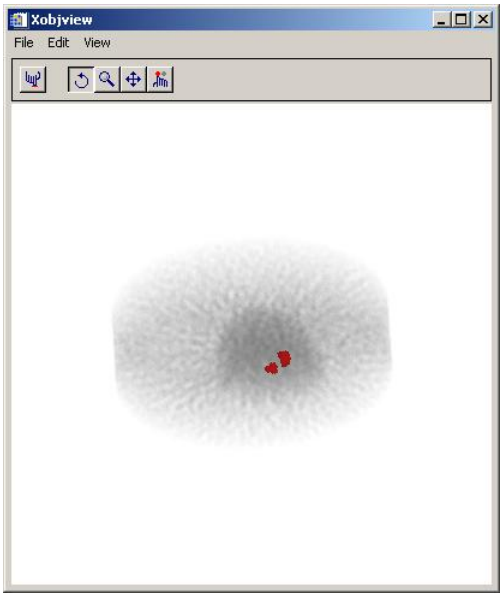
Op het einde van iteratie 2 verliep alles nog steeds volgens het tijdsplan uit het plan van aanpak. Voor iteratie 3 kon daardoor ook volgens de planning worden gestart met het functioneel ontwerp. Dit ontwerp was vrij snel klaar aangezien de functionaliteit van het 3D-venster bekend was, waardoor er vrij snel gestart kon worden met het Technisch Ontwerp.

Tijdens een vroegtijdige zoektocht naar 3D weergave kwam een object naar voren dat Atomic Graphic Objects kon weergeven, roteren, verplaatsen en zoomen. Dit was het xobjview object (zie figuur 11). In feite is in dit object de gevraagde functionaliteit voor een groot deel al uitgewerkt.

Het nadeel aan dit object is de flexibiliteit. Het xobjview object maakt zelf een nieuw venster. In dit venster wordt de afbeelding weergegeven, maar zijn er ook een aantal menu's en extra knoppen beschikbaar. Deze knoppen en menu's zijn niet uit de aanroep te verwijderen. Verder moest het mogelijk zijn om het grafisch deel in een eigen venster (of widget) te plaatsen, dit om een naadloze implementatie met het framework te verkrijgen.

Na een korte zoektocht door de help-functie van IDL bleek dat van xobjview de broncode beschikbaar was. Deze klasse is opgezocht, waarna onderzocht is of de grafische component uit het object kon worden gehaald.

Tijdens het ontrafelen van xobjview kwam ik op internet (zie literatuurlijst, referentie 4) bij een pagina uit over een aantal ongedocumenteerde klassen binnen IDL. De zogenaamde IDLex klassen. Op basis van deze klassen is het xobjview object dan ook gemaakt.

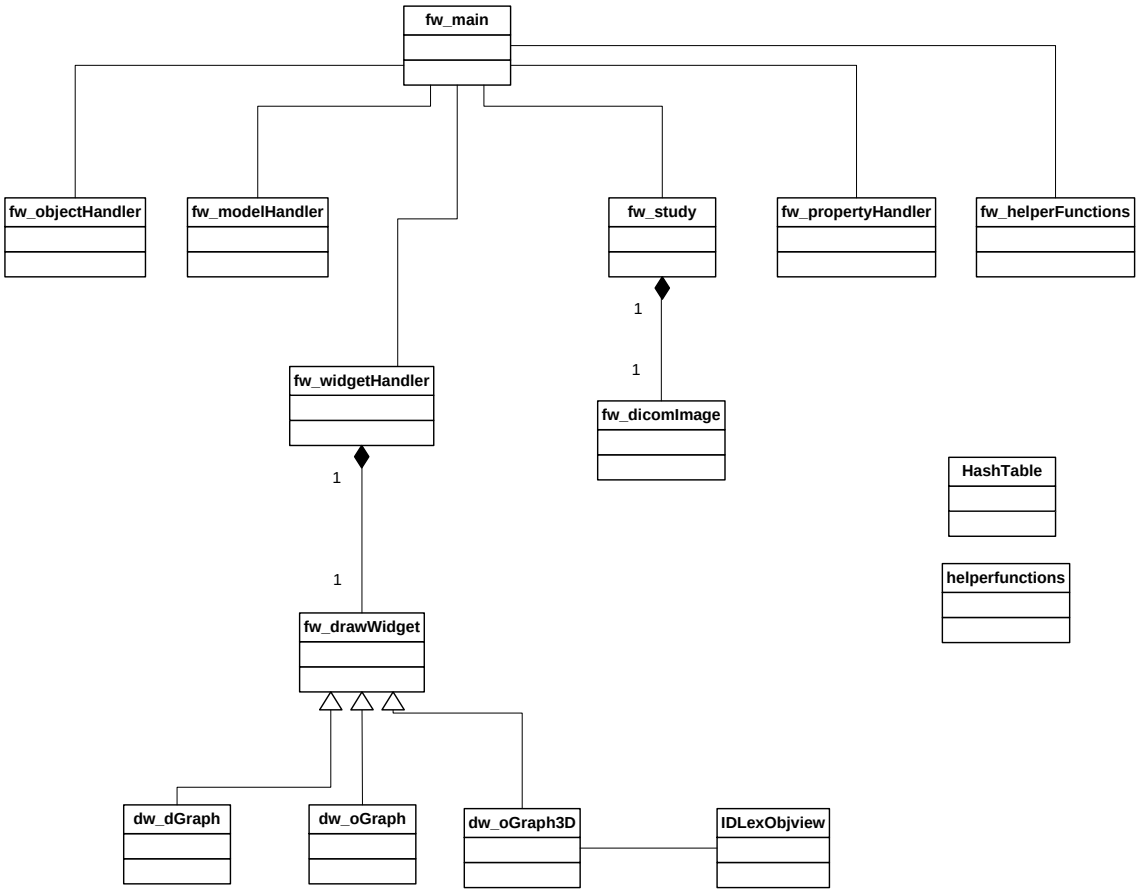


Figuur 15: Het xobjview object

Na een bestudering van het klassendiagram van de IDLex familie bleek dat IDLexObjview de gevraagde functionaliteit van het grafische venster bevatte (voor meer informatie over de IDLex klassen, zie bijlage C, hoofdstuk 2).

Aangezien in iteratie 2 al duidelijk was hoe de grafische presentatie van een ROI gemaakt kon worden (paragraaf 5.2.2), was het eenvoudig om dit te verwerken in het IDLexObjview object. Door de updates van het fw_roiGroup object ook door te sturen naar het polyline-object in IDLexObjview, was het mogelijk om real-time aanpassingen te verrichten in de verschillende 2D vensters en deze door te geven aan het 3D venster.

Verder moest het IDLexObjview object in het framework worden geïmplementeerd. Aangezien het een compleet andere weergavemethode betreft, is er gekozen om een nieuwe klasse bij te maken, namelijk dw_oGraph3D (zie figuur 12). Zoals de naam al aangeeft wordt er in deze klasse een object graphics object beheerd. Verder worden de rotatie, verplaatsing en zoom methodes doorverwezen naar het IDLexObjview object, aangezien deze dergelijke functionaliteit al bezit.
Door het gebruik van dw_oGraph3D wordt het voor een programmeur eenvoudiger om gebruik te maken van het 3D venster. Het enigste wat deze moet doen is het opgeven van de dicom-data, en het koppelen van het venster in een aanzichts-collectie. Deze collectie wordt ook al gebruikt in de 3x 2D weergave om bij VOI's te bepalen in welke vensters de representatie getekend moet worden.



Figuur 16: Domeinmodel voor uitbreiding Iteratie 3

5.4 Efficiënte arraybewerkingen

In IDL is het mogelijk om in één regel arraybewerkingen uit te voeren door middel van wildcards. Hiervoor zijn bij andere programmeertalen meerdere regels nodig, bijvoorbeeld door gebruik van loops.

Voorbeeld

Als voorbeeld staan hieronder twee manieren om een array te vullen. De eerste manier is zoals het in Java (en andere 3e generatie programmeertalen gebeurt). In het tweede voorbeeld staat hoe dit mogelijk is binnen IDL.

Java:

```
int[][] arrayVar = new int[20][20];
for(int i=0; i<20; i++) {
   for(int j=0; j<20; j++) {
      arrayVar[i][j] = 12;
   }
}
```

IDL:

```
arrayVar = intarr(20,20)
arrayVar[*,*] = 12
```

Aangezien voor visualisatie verschillende aanzichten nodig zijn, zullen er ook vanuit een drie-dimensionaal array verschillende waardes opgehaald moeten worden. Voor een transversale afbeelding zullen alle waardes van X en Y nodig zijn, terwijl bij een sagittale afbeelding alle waardes van Y en Z nodig zijn.

Nu was de vraag hoe snel IDL gegevens uit een array kan halen. Het moet namelijk mogelijk zijn om uit een 3D of 4D array een 2D plaatje te extraheren. Dit plaatje zal een bepaald plane (en tijdsframe) beschrijven. Zodra een arts hierop moet wachten, zal het bestempeld worden als hinderlijk. Daarom dient er onderzocht te worden wat de snelste methode is om informatie uit een 3 of 4-dimensionaal array te halen.

Het blijkt dat bladeren door een array met een maximale grootte van 128x128x47x64 elementen "on the fly" mogelijk is. Dit zou een afbeelding van 128x128 pixels kunnen zijn, gemaakt over 47 planes met 64 tijdsframes. Zodra meer elementen gebruikt worden, zal er bij het bladeren een merkbare vertraging optreden voor de eindgebruiker.
Caching bleek hier geen optie, aangezien dat een zeer lange tijd nodig had om de arrays van tevoren om te vormen. Bij 128x128x47x64 elementen duurde het 23 seconden voordat van één aanzicht een cache was gemaakt.
Verder bleek dat het gebruik van wildcards (het gebruik van * in arraybewerkingen) sneller verliep in de laatste dimensies, dan in de eerste dimensies. Er wordt aangenomen dat dit te maken heeft met het gebruik van geheugenplaatsen. Als achteraan in de dimensies een wildcard wordt gebruikt, kan het systeem een aaneengesloten stuk geheugen muteren of ophalen. Als vooraan in de dimensies een wildcard gebruikt wordt, betekend dit dat het systeem op allerlei verschillende plaatsen in het geheugen de informatie moet muteren/ophalen. Dit laatste heeft klaarblijkelijk een langere tijd nodig.

Concreet betekent dit dat het verstandiger is om de "on the fly" methode te gebruiken. Bij caching zal de tijd veel te snel oplopen als de array groter wordt. Bij "on the fly" kan een grotere dataset gebruikt worden zonder dat het hinderlijk wordt voor de eindgebruiker.

Meer informatie over dit onderzoek is te vinden in bijlage B, hoofdstuk 1.

5.5 Multithreading binnen IDL

Uit onderzoek blijkt dat de programmeertaal IDL gebruik kan maken van meerdere processoren, en hierdoor dus multithreading kan toepassen. Dit zou goed bruikbaar zijn bij het maken van een service om het cache-probleem van grote datasets op te lossen. Deze service zou op de achtergrond bestanden kunnen inladen. Zodra de mogelijkheid zou bestaan dat de informatie uit deze bestanden binnen korte tijd nodig zou zijn, zou de service deze bestanden op de achtergrond kunnen inladen.

De functionaliteit van deze service zou gebruik maken van een aparte thread die stukken van de dataset bijhoudt, en die aangeroepen wordt om informatie te verkrijgen. Hierbij zou de thread controleren wat de laatst opgeroepen informatie is, en of deze rond het midden van de ingelezen bestanden ligt. Als de opgeroepen informatie zich aan de rand van de gecachte bevindt, wordt een nieuwe set bestanden ingelezen, zodat de laatst aangeroepen informatie weer in het midden ligt. Doordat het een aparte thread is, zal de tijd voor het inlezen los komen van de tijd voor het opvragen. Hierdoor kan binnen een bepaalde range informatie snel opgeroepen worden. Dit snel oproepen geeft voor de eindgebruiker het idee dat de complete set snel doorlopen kan worden.

Dit concept met multithreading bleek helaas niet mogelijk te zijn. IDL bekijkt namelijk zelf of een berekening over meerdere processoren verdeeld kan worden, maar stelt de gebruiker niet in staat om zelf threads te maken. Dit komt omdat de taal ook wordt gebruikt door niet-informatici, en deze minder goed problemen zoals deadlocks of starvation kunnen voorzien. Nu bestaat de mogelijkheid om zelf een aantal C-functies te schrijven en deze via IDL aan te roepen. Hierdoor is het mogelijk de DICOM-service in C schrijven, en aan te roepen wanneer het nodig is. Deze optie vermijdt het probleem dat IDL maar één event queue heeft, en hierdoor geen simultane acties kan uitvoeren. Dit kan een oplossing zijn, maar is niet aan te raden aangezien alle C-pointers in IDL bewaard zouden worden. IDL roept alleen maar een functie aan. Er zal dan een functie zijn die een aparte thread heeft, maar die oneindig doorgaat (endless loop). IDL zal dan een andere C-functie aanroepen die dezelfde pointer naar de cached-data heeft als de endless loop, waardoor het mogelijk is de data naar IDL te transporteren. De c-thread krijgt dan een nieuwe waarde als input, waarop hij zijn middelpunt kan bepalen.

De hierboven beschreven oplossing brengt nog een ander probleem mee, namelijk dat gecontroleerd moet worden of beide threads niet tegelijk bezig zijn met het lezen/schrijven van dezelfde geheugenplaats. Dit kan namelijk weer leiden tot fouten.

5.6 IDL ten opzichte van 3e generatie programmeertalen

Aangezien in de toekomst het iMed framework mogelijk voor meerdere doeleinden gebruikt gaat worden (andere type scans zoals MRI of CT, of grotere type datasets van PET-scans) zal de hoeveelheid data die verwerkt moet worden mogelijkerwijs ook toenemen.

In de vorige paragraaf is beschreven dat IDL niet om kan gaan met multithreading voor het beheren van threads. Hierdoor worden de mogelijkheden voor het cachen van arrays beperkt (zie paragraaf 5.5).
Door deze limitaties van IDL ontstond de vraag of deze programmeertaal wel geschikt was voor deze doeleinden. IDL is een 4e generatie programmeertaal waarbij een aantal functies zijn vereenvoudigd, zodat er ook wetenschappelijk door andere beroepsgroepen mee geprogrammeerd kan worden. Een voorbeeld hiervan is de manier waarop IDL array-handelingen kan uitvoeren. Voor het ontwikkelen van grote (enterprise) applicaties zijn dergelijke talen minder handig in gebruik. Er worden namelijk aannames gemaakt waardoor er voor een programmeur opties ontbreken die hij graag zou willen toepassen. Een voorbeeld hiervan is het gebruik van multithreading. ITT VIS (het bedrijf dat IDL ontwikkeld) geeft zelf aan dat het programmeren van threads is weggelaten omdat het bij onjuist gebruik problemen kan introduceren. Verder geven ze aan dat IDL voor één event queue ontwikkeld is. Voor de implementatie waarbij zelf threads gemaakt kunnen worden, zou het achterliggend systeem van IDL herschreven moeten worden. Hiervan ziet ITT VIS af, en kiest ze om alleen binnen berekeningen meerdere processoren automatisch toe te passen waar dat mogelijk is (zie literatuurlijst, referentie 3). Voor een wetenschapper is deze implementatie van multithreading fijn, omdat deze geen rekening hoeft te houden met de lopende threads. Voor een programmeur is dit een nadeel aangezien deze zelf threads wil schrijven in plaats van de automatische implementatie.

Ook was voor mij de vraag of een 4e generatie programmeertaal wel snel genoeg is. IDL is namelijk gebaseerd op objective-c. Hierdoor komt er een laag bovenop C, wat volgens mij merkbaar zou zijn in performance.
Om deze redenen heb ik IDL vergeleken met C# en Java. In deze vergelijking ben ik uitgegaan van de snelheid waarmee arraybewerkingen worden uitgevoerd.
Het resultaat van deze test was vrij verassend. IDL was het traagst, terwijl Java het snelst was. C# kwam gemiddeld vlak achter Java aan. Dit was tegen mijn (subjectieve) verwachtingen in. Zelf had ik gedacht dat Java het traagst zou zijn, en IDL het snelst.

Meer informatie over de uitvoering en het resultaat van dit onderzoek is te vinden in bijlage B, hoofdstuk 2.

6 Resultaat

De primaire doelstelling van de afstudeeropdracht was om de weergave van de DICOM informatie vanuit meerdere aanzichten mogelijk te maken. Na een kennismaking met IDL, en een kennismaking met het framework is deze primaire doelstelling bereikt.

In de initiële opdrachtomschrijving vanuit het bedrijf (de afdeling Nucleaire Geneeskunde van het UMCN) stond dat de weergave vanuit verschillende aanzichten, en de 3D weergave het doel van de stageopdracht was. Het tekenen van 3-dimensionale regions (VOI's) zou onderzocht moeten worden, en waar mogelijk geïmplementeerd.
Tijdens het opstellen van het Plan van Aanpak is het tekenen van VOI's en het creëren van een 3D weergave omgedraaid. Hierdoor kreeg de VOI-functionaliteit een hogere prioriteit dan aanvankelijk was gesteld.
In de praktijk is gebleken dat deze verschuiving geen problemen heeft opgeleverd voor de afstudeerstage en bijhorende proces. Allereerst is het mogelijk gemaakt om in de weergave met meerdere aanzichten een VOI te tekenen. Tevens is de implementatie gebouwd om dergelijke volumes te roteren, verplaatsen of schalen binnen de verschillende aanzichtsvensters.

Hierbij moet wel een detail worden opgemerkt met betrekking tot de weergave van volumes. Zodra een volume wordt geroteerd over een as anders dan de plane-as, ontstaan er op dit moment nog kleine artefacten in de weergave in het venster dat één plane weergeeft. Aangezien de punten worden geroteerd, zullen deze decimale waardes bevatten na de rotatie. Pixels daarentegen zijn op gehele getallen gebaseerd. Hierdoor kan een punt "verdwijnen" dat eigenlijk zichtbaar zou zijn. Dit kan verholpen worden door deze punten af te ronden, maar daardoor worden ook andere (ongewenste) punten zichtbaar gemaakt. Hierdoor kan de weergave kleine oneffenheden vertonen. De achterliggende data (en de 3D weergave) zal wel correct zijn, aangezien deze wel met decimale getallen overweg kunnen.

Door de manier waarop datapunten van een Volume of Interest worden opgeslagen, is het ook mogelijk om in de toekomst de volumes aan te passen in de weergave waarin niet de originele vorm is getekend. Hierdoor kan een volume ook andere vormen dan een cilinder, kubus of U-vorm aannemen. Bijvoorbeeld een bol of piramide.

Het resultaat van dit project is de uitbreiding op het framework. Door een goede, systematische aanpak is het resultaat een nuttige uitbreiding op het framework. Het is mogelijk geworden om met het iMed framework informatie te verkrijgen en weer te geven in drie dimensies. Iets wat van tevoren niet mogelijk was. Deze uitbreiding kan in de toekomst gebruikt worden voor studies en onderzoeken.

Voor de toekomst kan onderzocht worden of het mogelijk is de DICOM-data op een efficiënte manier te gebruiken. Zoals in dit verslag aangetoond is het niet mogelijk om multithreading op een dergelijke manier toe te passen dat er zelfstandig draaiende services ontwikkeld kunnen worden. Deze services zouden van pas komen bij het efficiënt gebruiken van de DICOM-informatie uit verschillende DICOM-bestanden. Er is een mogelijkheid om een filepointer te maken binnen IDL, zodat er arrays gebruikt kunnen worden die groter zijn dan het geheugen toestaat. Dit kan vergeleken worden met een "swap-file". De vraag hierbij is dan hoe snel informatie aangeroepen kan worden uit deze pointers. Verder is het probleem van multithreading nog niet opgelost. Een alternatief voor multithreading zou zijn om een apart IDL proces (lees: aparte applicatie) te starten die op de achtergrond draait. Door (file)pointers tussen de applicaties uit te wisselen, of door het gebruik van sockets zou het mogelijk moeten zijn om de processen met elkaar te laten communiceren.

Buiten dit resultaat en de aanbevelingen is de vraag ontstaan of IDL in de toekomst volstaat voor het iMed framework. Dit aangezien de informatiestromen in de toekomst mogelijk groter zullen worden (meer data per scan). Zoals in het onderzoeksparagraaf beschreven is, er een aantal (kleine) punten waarin IDL niet, of minder uit de doeken komt in vergelijking met andere programmeertalen. Natuurlijk zijn er ook een aantal grote voordelen aan IDL zoals eenvoudige arraybewerkingen en de aanwezige DICOM-klasse. Deze staan ook beschreven bij het onderzoeksresultaat (zie hoofdstuk 2, bijlage B). Als deze voordelen door andere (populaire) talen zoals Java of C# worden bijgehaald, ontstaat er een mogelijkheid dat deze populaire talen de voorkeur krijgen boven IDL. Er zijn op dit moment al een aantal libraries beschikbaar voor C# en Java, maar deze zijn verder niet onderzocht. Aan IDL wordt ook steeds verder ontwikkeld, en er zal ook functionaliteit toegevoegd of verbeterd

worden. Binnen het UMCN wordt namelijk gebruik gemaakt van versie 6.3, terwijl de nieuwste versie 7.1 is. Het kan dus goed zijn dat sommige beschreven problemen reeds zijn opgelost. De keuze om nog geen upgrade uit te voeren heeft te maken met het e.Soft workstation. Dit workstation wordt (zoals eerder geschreven) door Siemens beheert. Op de meeste e.Soft workstations waar het iMed framework wordt gebruikt is IDL versie 6.3 beschikbaar. Zodra het framework gebruik gaat maken van versie 7.1 zal op de betreffende workstations de versie van IDL bijgewerkt moeten worden. Dit zal de eenvoud bij het implementeren van een studie niet ten goede komen.

7 Evaluatie

7.1 Procesevaluatie

Allereerst was het een zeer boeiende en interessante afstudeerstage. Vooral de medische achtergrond heeft me zeer aangesproken, ook tijdens de opdracht. Van deze achtergrond heb ik dan ook veel geleerd tijdens de stage. Je bent namelijk met een opdracht bezig die in de toekomst een maatschappelijke bijdrage kan leveren. Daarom wordt er ook een grotere nadruk gelegd op het weten wat een stuk code doet, en er zeker van zijn dat de informatie juist is. Natuurlijk is dit in andere opdrachten ook van belang, maar in deze situatie betekent het of iemand behandeld kan of moet worden. Hierdoor krijgt de correctheid van de programmatuur een grotere rol.

Terugkijkend op de afstudeerstage kan ik zeer tevreden zijn over de aanpak en de uitvoering van het project. Tegenover mijn vorige stage had ik tijdens deze stage meer vrijheid om mijn opdracht naar eigen inzicht in te plannen. Vooraf heb ik wel eens getwijfeld of de planning niet te ruim of te strak was, maar bleek dit in de praktijk mee te vallen.
Vooral door een goede aanpak op het gebied van technische en functionele ontwerpen zijn vóór de daadwerkelijke realisatie al een aantal knelpunten opgelost. Ik vermoed dat herschrijven van stukken code veel meer tijd in beslag had genomen.
Ook het wiskunde gedeelte vond ik best interessant. Hiermee doel ik op het gebied van snijvlakken, raakpunten, hoekberekeningen, aanzichten en dergelijke. Binnen de opleiding is er nooit veel aandacht aan besteedt hoe dit uitgewerkt kan worden. Dit was dus een goed leerproces met betrekking tot het uitwerken van wiskundige aspecten in programmatuur.

Verder was de begeleiding naar mijn idee prima. In het Plan van Aanpak had ik dit nog als projectrisico opgenomen, maar het aantal contactmomenten met de bedrijfsbegeleiders was meer dan voldoende. Hierin komt het de werking van het functioneel en technisch ontwerp naar voren. Door deze documenten eerst te laten beoordelen, was het duidelijk of ik op de goede weg zat. Deze aanpak was naar mijn idee best prettig om mee te werken.
Ook de schoolbegeleiding was naar mijn idee goed. Natuurlijk is Nijmegen niet naast de deur vanuit Eindhoven, maar is dit vrij goed verlopen via e-mail. Ook de feedback vanuit de docentbegeleider op verschillende punten was terecht en constructief.

7.2 Productevaluatie

Aan het begin van de stage had ik nog wel vraagtekens bij het realiseren van het product. Een onbekende programmeertaal in combinatie met veel nieuwe (achtergrond)informatie is niet echt een goed uitgangspunt om de afstudeerstage mee te beginnen. Het voordeel was wel dat ik de programmeertaal vrij vlot onder de knie had. Wat betreft de (achtergrond)informatie heeft de afdeling ook meegeholpen door een open instelling. Ook de arts-onderzoeker die op dezelfde kamer zijn werkplek had was geïnteresseerd en droeg ook regelmatig ideeën aan om naar te kijken. Vooral op gebied van usability en data-verantwoording.

Wat betreft het eindproduct ben ik zeer tevreden. Het punt wat ik het mooiste aan het product vind is dat het in praktijk ook echt goed werkt zoals ik had gehoopt/verwacht. Hiermee doel ik op de koppeling tussen verschillende schermen tijdens aanpassingen aan het VOI-object. Vooral dat tijdens de rotatie in 2D het 3D venster direct op de juiste manier meewerkte was voor mezelf een bewijs dat de uitbreiding goed was opgezet.

8 Literatuurlijst

1.: IDL Programming Techniques

David W. Fanning, "IDL Programming Techniques – Second Edition", Fanning Software Consulting (2007)

2.: Power Graphics with IDL

Ronn Kling, "Power Graphics with IDL – A beginners guide to IDL Object Graphics", Kling Research and Software (2002)

3.: Multi-threading in IDL

http://www.ittvis.com/portals/0/pdfs/idl/IDL_MultiThread.pdf

4.: XOBJVIEW: Under the Hood

http://www.paulsorenson.com/underthehood.html

5.: MRI, ROI and displaying as a meshed object

http://groups.google.com/group/comp.lang.idl-pvwave/browse_thread/thread/502a65a52a08cbca/0da316c8ae020ffb?lnk=gst&q=IDLgrRoiGroup#0da316c8ae020ffb

Bijlage A: Plan van Aanpak

Documenthistorie

Revisies

Versie	Status	Datum	Wijzigingen
0.1	concept	03-03-2003	Opzet van Template
0.2	concept	12-02-2009	Eerste conceptversie
0.3	concept	16-02-2009	Bijlages ingevoegd, en door auteur gereviewd
0.4	concept	20-02-2009	Aanpassingen n.a.v. feedback bedrijfsbegeleider
0.5	concept	09-03-2009	Aanpassingen n.a.v. feedback docentbegeleider

Goedkeuring

Dit document behoeft de volgende goedkeuringen:

Versie	Datum goedkeuring	Naam	Functie	Paraaf
0.5	10-03-2009	B. van Gestel	Docentbegeleider	

Distributie

Dit document is verstuurd aan:

Versie	Datum verzending	Naam	Functie
0.3	16-02-2009	Ing. L. de Groot & Ing. E. Janssen	Stagebegeleiders
0.4	20-02-2009	Ing. L. de Groot & Ing. E. Janssen	Stagebegeleiders
0.4	26-02-2009	Dhr. B. van Gestel	Docentbegeleider
0.5	10-03-2009	Dhr. B. van Gestel	Docentbegeleider

Managementsamenvatting

Doel van dit document

Dit document heeft tot doel het project te definiëren, als basis te dienen voor het management ervan en de beoordeling van het resultaat mogelijk te maken.

De belangrijkste reden voor gebruik van dit document is om te dienen als basisdocument op grond waarvan de opdrachtgever en de opdrachtnemer de voortgang en wijzigingen kunnen toetsen en bewaken, en vragen omtrent de geldigheid van het project tijdens de uitvoering ervan te kunnen beoordelen.

Aanleiding

De afdeling Nucleaire Geneeskunde van het UMC St. Radboud te Nijmegen heeft een globale afstudeeropdracht ingevoerd binnen het Stage- en Afstudeer Informatie Systeem (hierna genoemd SAIS) van Fontys Hogeschool ICT te Eindhoven. Aangezien de student op zoek was naar een afstudeerplaats binnen de medische wereld, was dit een passende opdracht.
Wel bleek dat deze opdracht al sinds 2005 in het systeem stond. Nadat de student contact had opgenomen met de contactpersoon van het UMC, is er in overleg besloten een opdracht te definiëren om het intern ontwikkelde en gebruikte software framework uit te breiden met 3D ondersteuning/visualisatie.

Globale aanpak

Allereerst zal er een periode worden ingepland waarbij de student zich bekend maakt met de systemen en programmeertalen die gebruikt worden binnen de afdeling Nucleaire Geneeskunde. Hierbij zal een aantal weken worden besteed aan zelfstudie in de programmeertaal IDL. Tijdens deze periode zal de student dit Plan van Aanpak / Project Initiation Document (PiD) opstellen. Hierna zal er een korte studie plaatsvinden over het huidige framework.

Tijdens/na deze studie zal er gestart worden met de drie iteraties waarin het framework wordt uitgebreid. In deze iteraties zal het functioneel ontwerp, het technisch ontwerp, en de bouw steeds aan de orde komen om een nieuw productonderdeel te realiseren.

Tijdens de opdracht zal er technische documentatie geschreven worden over de nieuw ontwikkelde functionaliteit. Deze zal als commentaar in de code staan (door middel van pre en post condities), en in het technisch ontwerp.

Globale kosten en doorlooptijd

De doorlooptijd van dit project bedraagt 20 weken. Aangezien het een afstudeerstage is, zullen de kosten berusten op de stagevergoeding die de student hiervoor ontvangt.

Inhoudsopgave

1 Inleiding

1.1 Doel van dit document

Dit document is opgesteld om alle relevante basisinformatie en uitgangspunten van het project vast te leggen om het op de juiste wijze te kunnen besturen. Het heeft tot doel het project te definiëren, als basis te dienen voor het management ervan en de beoordeling van het succes van dit project mogelijk te maken.

Dit Project Initiation Document behandelt de volgende fundamentele aspecten van het project:

- Wat beoogt men met het project te bereiken?
- Waarom is het belangrijk om deze doelstellingen te bereiken?
- Wie zijn er betrokken bij het managen van het project en wat zijn hun rollen en verantwoordelijkheden?
- Hoe en wanneer zullen de maatregelen die in dit PID besproken worden gerealiseerd worden?

Het document wordt gebruikt als basisdocument op grond waarvan de begeleiders de voortgang en wijzigingen kunnen toetsen en bewaken.

1.2 Opbouw van dit document

Om aan te geven welke onderdelen worden bijgewerkt en dus nieuwe versies zullen krijgen tijdens de voortgang van het project is dit Project Initiation Document verdeeld in twee secties: een statisch gedeelte en een dynamisch gedeelte:

Het "statische" deel bestaat uit de hoofdstukken en bijlagen:

- Achtergrond (Hoofdstuk 2)
- Projectdefinitie (Hoofdstuk 3)
- Projectorganisatiestructuur (Hoofdstuk 4)
- Projectbeheersing (Hoofdstuk 5)
- Communicatieplan (Bijlage A)

Het "dynamische" deel bestaat uit de bijlage:

- Initieel Projectplan (Bijlage B)

2 Achtergrond

Binnen de afdeling Nucleaire Geneeskunde van het UMC St. Radboud spelen PET-scans een grote rol. Voordat deze scan begint, krijgt de patiënt nucleïden toegediend. Een voorbeeld van een nucleïde is FDG. Dit is een glucosemolecuul samengevoegd met radioactief fluor. Aangezien het menselijk lichaam dit FDG als glucose ziet, wordt het getransporteerd naar organen en weefsels die op dat moment veel energie nodig hebben.

Aangezien een tumor een afwijking in de cellen is waardoor deze sneller groeien, zal er ook meer energie nodig zijn om de celdeling mogelijk te maken. Hier komt dus een verhoogde concentratie FDG terecht. Door deze verhoogde concentratie zal er ook meer radioactieve straling uitgezonden worden op dit punt. Deze straling wordt opgevangen door de PET-scanners, en hierdoor kunnen er beelden gevormd worden op basis van meer of minder straling op bepaalde punten. Beelden van deze scans worden gebruikt om conclusies te trekken over tumoren of schildklierweefsel. Deze beelden worden dan weer gebruikt voor het stellen van een diagnose. Verder wordt er veel research gedaan naar de mogelijkheden met nieuwe/verschillende soorten nucleïden.

Op de afdeling worden ook SPECT-scanners gebruikt (gammacamera's). Deze werken volgens het zelfde principe als de PET-scanners, maar hebben een lagere resolutie.

Het maken van dergelijke PET-scans wordt op deze afdeling uitgevoerd met apparatuur van Siemens. Hierbij is een softwarepakket beschikbaar, genaamd e-Soft. Deze software draait op een Workstation. Op deze software kunnen zelf uitbreidingen geschreven worden. Zo wordt er een scan gemaakt, en kunnen verschillende bewerkingen uitgevoerd worden ter bevordering van de diagnostiek. Deze uitbreidingen worden in de programmeertaal IDL geschreven.

IDL staat voor Interactive Data Language en is ontwikkeld in het Laboratory for Atmospheric and Space Physics op de universiteit van Colorado. Daarna is het in handen gekomen van het bedrijf RSI. Een aantal jaar geleden is RSI overgenomen door ITT Industries onder de naam ITT Visual Information Solutions. De kracht van deze taal zit in de beeldverwerking, specifiek de arraybewerkingen. Een aantal veel voorkomende arraybewerkingen bij (medische) afbeeldingen zijn vereenvoudigd zodat ze één regel in beslag nemen (in plaats van meerdere loops). Hierbij kan gedacht worden aan vermenigvuldigingen, delingen, optellingen en dergelijke.

Binnen de afdeling zijn 2 informatici parttime werkzaam. Zij hebben verschillende uitbreidingen voor het e-Soft systeem gemaakt. Deze uitbreidingen maken ze op basis van een intern ontwikkeld framework, genaamd iMed. Het voordeel van dit framework is dat er heel gemakkelijk en snel programma's ontwikkeld kunnen worden voor nieuwe studies/analyses. Het nadeel hiervan is dat er een aantal software mogelijkheden beschikbaar zijn. Voor nieuwe softwaremogelijkheden zal het framework uitgebreid moeten worden.

Er is de behoefte vanuit de afdeling/externen om applicaties te ontwikkelen waarbij 3D berekeningen op scans uitgevoerd kunnen worden. Hierbij kan gedacht worden aan volumeberekeningen en dergelijke. Hiervoor dient het iMed framework uitgebreid te worden, aangezien er op dit moment nog geen 3D ondersteuning beschikbaar is.

3 Projectdefinitie

3.1 Projectdoelstellingen

De projectdoelstelling is om een uitbreiding te realiseren aan het iMed framework. Deze uitbreiding moet het mogelijk maken om in de toekomst eenvoudig 3D applicaties te kunnen ontwikkelen met het framework. Dit betekent dat er uitbreidingen gedaan worden aan het model, maar ook aan de weergavemogelijkheden binnen het framework. Dit moet ervoor zorgen dat een programmeur eenvoudig een applicatie kan schrijven om een diagnose of research in 3D uit te voeren.

3.2 Gekozen oplossing of aanpak

Het project wordt benaderd doormiddel van de projectstrategie DSDM. Dit staat voor Dynamic Systems Development Method. Hierbij wordt aan het begin globaal vastgelegd wat de eisen (functioneel en niet functioneel) van het project zijn. Tijdens het verloop van het project wordt er dan steeds meer specifieke informatie over het project duidelijk.

Bij deze methode is de opdrachtgever/gebruiker nauw betrokken bij het ontwikkelproces. Er wordt namelijk per iteratie een deelproduct opgeleverd. Na iedere iteratie zal het functioneel en technisch ontwerp worden uitgebreid in overeenstemming met de opdrachtgever. Hierdoor wordt voorkomen dat een project niet aan de verwachte eisen van de opdrachtgever voldoet. Het project kan namelijk na iedere iteratie bijgesteld worden.

Er is niet gekozen voor een Scrum of Extreme Programming aanpak. Bij een scrum aanpak zijn er meerdere ontwikkelaars beschikbaar. Deze werken dan heel nauw samen om een iteratie zo goed mogelijk op te leveren. Extreme Programming berust op het feit dat de wensen en eisen nog niet bekend zijn bij de start van het project. Aangezien dat bij dit project niet het geval is, wordt deze aanpak niet gebruikt.

Zoals hierboven geschreven zijn er meerdere iteraties in dit project. In de 1e iteratie zullen er 3x 2D beelden gegenereerd worden. Deze drie beelden zijn projecties vanuit verschillende aanzichten van het object. In de 2e iteratie zullen er 3-dimensionaal regions getekend worden binnen de weergaves die gemaakt zijn in de eerste iteratie. In de 3e iteratie zal er onderzocht worden hoe het mogelijk is om een 3D weergave te maken. Mocht het mogelijk zijn zal dit ook worden uitgewerkt.

3.3 Scope van het project

Bij het project is de afdeling Nucleaire Geneeskunde van het UMC St. Radboud betrokken. De uitbreiding op het framework zal onder andere binnen deze afdeling gebruikt worden. Met dit framework kunnen applicaties ook voor andere ziekenhuizen/afdelingen ontwikkeld worden. Dit valt in principe buiten de scope van het project, aangezien dit andere projecten zijn en het huidige project zich richt op het uitbreiden van het framework.

3.4 Producten c.q. eindresultaat

Als product zal een uitbreiding op het iMed framework geleverd worden, samen met de bijbehorende documentatie. Dit uitgebreide framework bevat dan de functionaliteit om deze beelden op meerdere manieren te weergeven. Ook kunnen hier basis-berekeningen met behulp van ROI's (Region of Interest) op uitgevoerd/getekend worden.

3.5 Uitsluitingen

Alle afdelingen buiten Nucleaire Geneeskunde, of andere ziekenhuizen die producten van dit framework gebruiken, zijn niet betrokken tot het project. Verder zijn de applicaties, die ontwikkeld zijn met het framework, geen projectresultaat. Verder zijn de ontwikkelde applicaties met het framework geen projectresultaat. Deze dienen enkel ter aanduiding dat de uitbreiding op het framework ook echt meerwaarde heeft en werkt.

3.6 Afhankelijkheden

Tijdens dit project zijn er afhankelijkheden met personen/diensten die niet binnen de projectgroep vallen. Hieronder staan deze afhankelijkheden per categorie weergegeven:

- Het moet mogelijk zijn om de gevraagde bewerkingen in de programmeertaal IDL uit te voeren. De opdrachtnemer is dus afhankelijk van de beschikbare functies binnen de programmeertaal.
- De bedrijfsbegeleiders (en tevens consultants voor IDL) zijn parttime werkzaam binnen de afdeling. Hierdoor wordt het aanroepen van consult beperkt tot een aantal uren per week. Dit zal geen direct probleem zijn aangezien ze via mail of telefoon bereikbaar zijn, maar zou in enkele situaties een kleine vertraging op kunnen leveren.

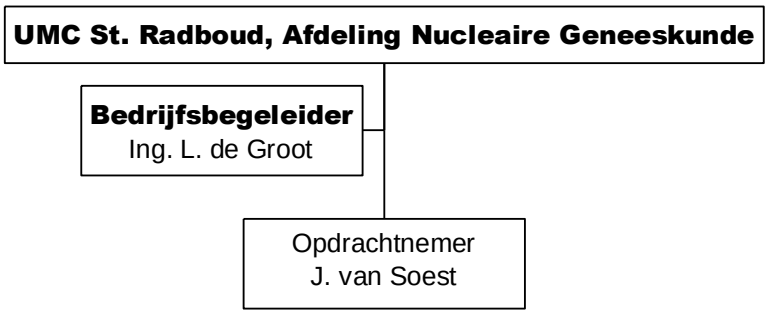
3.7 Randvoorwaarden

Allereerst zal er een werkplek beschikbaar moeten zijn zodat de opdrachtnemer het project kan uitvoeren. Verder zal op deze werkplek een PC beschikbaar moeten zijn. Op deze pc zal de opdrachtnemer een account moeten verkrijgen (binnen het netwerk), en zal er software beschikbaar moeten zijn om het project uit te voeren. Dit zal in de vorm van een IDE zijn voor programmeerwerkzaamheden, en een tekstverwerker (en eventuele tekentools) om de documenten te produceren.

3.8 Aannames

Er is op de afdeling voldoende kennis en documentatie over de programmeertaal, en over het huidige framework beschikbaar om deze opdracht uit te voeren.

4 Projectorganisatiestructuur



4.1 Opdrachtgever

1.1.1. Rolbeschrijving

De opdrachtgever voor dit project is het UMC St. Radboud, specifiek de afdeling Nucleaire geneeskunde. Deze afdeling heeft de opdracht uitgeschreven om het Siemens e-Soft systeem uit te breiden met nieuwe functionaliteit. Als persoon zal Ing. L. de Groot zich vertegenwoordigen als opdrachtgever, en tevens als bedrijfsbegeleider.

1.1.2. Projectgerelateerde taken

De opdrachtgever (in de persoon Ing. L. de Groot) dient betrokken te zijn bij de voortgang van het project. Deze zal een sturende taak hebben met betrekking tot dit project.

1.1.3. Specifieke verantwoordelijkheden

De opdrachtgever is er verantwoordelijk voor dat de opdracht binnen het UMC St. Radboud uitgevoerd kan worden. Hiermee wordt bedoeld dat er werkplekken beschikbaar zijn, en dat er aan de eerder omschreven randvoorwaarden en aannames wordt voldaan.

4.2 Bedrijfsbegeleider

1.1.4. Rolbeschrijving

De projectbegeleider binnen dit project is Ing. L. de Groot. Samen met zijn collega Ing. E. Janssen zullen zij de benodigde sturing geven aan het project. Zij zijn werkzaam op de afdeling Nucleaire Geneeskunde namens Siemens AG.

1.1.5. Projectgerelateerde taken

De projectbegeleider dient als eerste aanspreekpunt voor de opdrachtnemer. Hij zal tevens als consultant/begeleider op het gebied van de programmeertaal en architectuur van het framework optreden.

Verder ontvangt hij de docentbegeleider bij het bedrijfsbezoek en bespreekt met hem en de afstudeerder het sociaal en vakinhoudelijk functioneren.

1.1.6. Specifieke verantwoordelijkheden

Samen met de opdrachtgever is de bedrijfsbegeleider er verantwoordelijk voor dat de opdrachtnemer een werkplek tot zijn beschikking heeft, en dat er aan de eerder omschreven randvoorwaarden en aannames wordt voldaan.

4.3 Opdrachtnemer

1.1.7. Rolbeschrijving

De opdrachtnemer binnen dit project is dhr. J. van Soest, afstuderende binnen de Fontys Hogeschool ICT te Eindhoven. Tijdens het afstuderen zal hij werkzaam zijn binnen de afdeling Nucleaire Geneeskunde in het UMC St. Radboud te Nijmegen.

1.1.8. Projectgerelateerde taken

De opdrachtnemer dient het project uit te voeren binnen de tijdsspanne die beschikbaar is voor een afstudeerperiode, namelijk 20 weken. Hij is verantwoordelijk voor het eindproduct en dient regelmatig overleg te houden met de bedrijfsbegeleider/opdrachtgever, en te rapporteren aan de docentbegeleider.

1.1.9. Specifieke verantwoordelijkheden

De opdrachtnemer dient zelf initiatief te nemen voor rapportages, consult met betrokkenen en dergelijke acties ten behoeve van het project.

5 Projectbeheersing

5.1 Rapportage

Hieronder is in een matrix uitgewerkt wie aan welke personen rapporteert.

Partij: / Rapport:	Opdrachtgever / Bedrijfsbegeleider	Opdrachtnemer	Docentbegeleider
Projectvoorstel	O	I	I
PID	A + G	O + D	A + G
Document van Eisen	A + G	O + D	
Functioneel en Technisch ontwerp	A + G	O + D	
Afwijkingsrapport	I + G	O + D	I + G
Voortgangsrapport	I	O + D	I
Faseplan	I + A	O + D	
Fase-eindrapport	I + G	O + D	I
Projectissues	I + A	O + D	
Projecteindrapport	I	O + D	A + T

Legenda:

O	Opstellen	**A**	Adviseren	**I**	Ontvangen ter informatie
T	Toetsen	**D**	Distribueren/archiveren	**G**	Goedkeuren

5.2 Voortgangsbewaking

Hieronder staat een matrix. In deze matrix staat aangegeven wanneer er contact is en met wie. Dit heeft betrekking tot het bewaken van de voortgang tijdens/van dit project.

Overleg	Aanwezig	Frequentie	Tijdstip	Doel	Onderwerpen	Notulen
Voortgangsoverleg	Bedrijfsbegeleider en opdrachtnemer,	1x per week	Nader te bepalen	Afstemming over de voortgang en prioriteiten binnen het project	Voortgang, Open Issues, Risico's	Opdrachtnemer
Faseafronding	Bedrijfsbegeleider en opdrachtnemer	Faseovergang	Nader te bepalen	Bespreking faseproduct Goedkeuring start volgende fase	Fase-eindproduct, Faseplan, Projectplan, Risico's	Opdrachtnemer
Consult programmatuur	Bedrijfsbegeleider, opdrachtnemer	Waar nodig is	Nader te bepalen	Het vinden van oplossingen voor problemen binnen de architectuur of bouw van de uitbreiding	Het aangedragen probleem c.q. uitdaging	Opdrachtnemer
Bedrijfsbezoek docentbegeleider	Docentbegeleider, bedrijfsbegeleider en opdrachtnemer	1x	Nader te bepalen	Bespreken van huidige voortgang	Voortgang, opgeleverde documenten	Opdrachtnemer

5.3 Risicomanagement

Elk project heeft bepaalde vormen van risico. In dit project zijn ook diverse risico's van toepassing.

Een duidelijke en effectieve tijdsplanning is voor dit project ook zeer gewenst. Dit om inzicht te krijgen in de tijd, en om hierop bij te sturen waar nodig is. Door de tijdsplanning na iedere fase te reviewen, kunnen deze problemen worden voorkomen.

Omdat consultants/gebruikers worden geraadpleegd is het een pre dat deze een goede communicatie met de opdrachtnemer hebben. Indien er in een vroeg stadium fouten worden gemaakt ten aanzien van het model is dit achteraf moeilijk te corrigeren. Tijdens een faseovergang wordt er overleg gevoerd met de consultants/gebruikers of zij het eens zijn met het product van de fase.

Er zal ook een risico kunnen ontstaan dat de opdrachtnemer niet in staat is om het project uit te voeren. Bijvoorbeeld door ziekte of persoonlijke omstandigheden. Dit zal de opdrachtnemer dan zelf melden aan de bedrijfsbegeleider en de docentbegeleider. Als dit problemen voor de opdracht met zich mee zal brengen, zal er in overleg beslist worden hoe het project verder gaat. Hierbij zal er dus een revisie komen op bijlage B van dit document (het projectplan).

5.4 Afwijkingsprocedure

Zoals in bijlage B is te zien, zijn er per fase een uitloopweek beschikbaar. Indien de uitloopweek te kort is, moet de gehele planning worden herzien, en eventueel het project worden verlengd. Dit wordt beslist door de opdrachtnemer, samen met de bedrijfsbegeleider. Deze planning kan eventueel tussentijds bijgesteld worden indien men tijdens het project erachter komt dat er toch meer tijd nodig is voor een bepaald onderdeel.
Indien de wensen van de opdrachtgever niet realiseerbaar zijn in het project dient dit in een faseafronding bijgesteld te worden. Hierbij is overleg met de betreffende personen van belang. Zo kan er gekeken worden of een deel van de specificatie wel realiseerbaar is, of dat deze op een andere manier wordt geïmplementeerd.
Dergelijke afwijkprocedures worden formeel afgevangen door een change. Deze change moet worden doorgevoerd en dit levert een wijzigingsdocument op, of een nieuwe versie van een bestaand document.

Bijlage A: Communicatieplan

Revisies

Versie	Status	Datum	Wijzigingen
0.1	concept		geen wijzigingen
0.2	concept	12-02-2009	Eerste conceptversie
0.3	concept	16-02-2009	Review van bijlage uitgevoerd door auteur

Inleiding

Dit communicatieplan benoemt alle partijen die belang hebben bij het project en de wijze waarop zij bij het project zullen worden betrokken en welke communicatievormen daarbij gebruikt worden. Het gaat hierbij om partijen en communicatie buiten de formele projectmanagementstructuur zoals beschreven in het PID.

Belanghebbenden bij het project

Wie	Namens	Belang	Communicatievorm
Ing. L. de Groot	Afdeling Nucleaire Geneeskunde	Deze persoon is de bedrijfsbegeleider	Overleggen, adviseren, informeren
UMC St. Radboud, Afdeling Nucleaire Geneeskunde	-	Het uitgebreide framework zal hier als eerste in gebruik genomen worden	Informeren
B. van Gestel	Fontys Hogeschool ICT	Deze persoon is de docentbegeleider	Adviseren, informeren, besluiten, accepteren
J. van Soest	De student	Deze persoon is de student die de opdracht uitvoert	Overleggen, informeren, uitvoeren, besluiten

Communicatiekanalen

Van	Naar	Informatie	Medium	Frequentie of data
J. van Soest	Ing. L. de Groot	PvA/PiD en communicatieplan	Email	20-02-2009
J. van Soest	B. van Gestel	PvA/PiD en communicatieplan	Email	20-02-2009
J. van Soest	B. van Gestel	Eerste bedrijfsbezoek	Overleg	27-02-2009 of 06-03-2009
J. van Soest	Ing. L. de Groot	Product van fase	Email en overleg	Aan het eind van iedere fase. Zie hiervoor bijlage B
J. van Soest	B. van Gestel	Inleveren afstudeerverslag concept 1	Email	27-03-2009
B. van Gestel	J. van Soest	Feedback op concept 1 van afstudeerverslag	Email	03-04-2009
J. van Soest	B. van Gestel	Inleveren afstudeerverslag concept 2	Email	08-05-2009
B. van Gestel	J. van Soest	Feedback op concept 2 van afstudeerverslag	Email	15-05-2009
J. van Soest	B. van Gestel	Inleveren definitief	Post	10-06-2009

		afstudeerverslag		
J. van Soest	B. van Gestel	Kleine voortgangsrapportage over de gedane werkzaamheden	Email	Bij het afronden van de iteratie
J. van Soest	B. van Gestel, ing. L. de Groot	Proefpresentatie afstuderen (Locatie: UMC St. Radboud, Nijmegen)	Presentatie	19-06-2009 (nader definitief te bepalen)
J. van Soest	Afstudeercomissie	Presentatie afstuderen (Locatie: Fontys Hogeschool ICT, Eindhoven)	Presentatie	Nader te bepalen

Bijlage B: Project Plan

1.1.10. Revisies

Versie	Status	Datum	Wijzigingen
0.1	concept		geen wijzigingen
0.2	concept	12-02-2009	Eerste conceptversie
0.3	concept	16-02-2009	Review van bijlage uitgevoerd door auteur
0.4	Concept	20-02-2009	Aanpassingen n.a.v. feedback stagebegeleider

1.1.11. Goedkeuring

Dit document behoeft de volgende goedkeuringen:

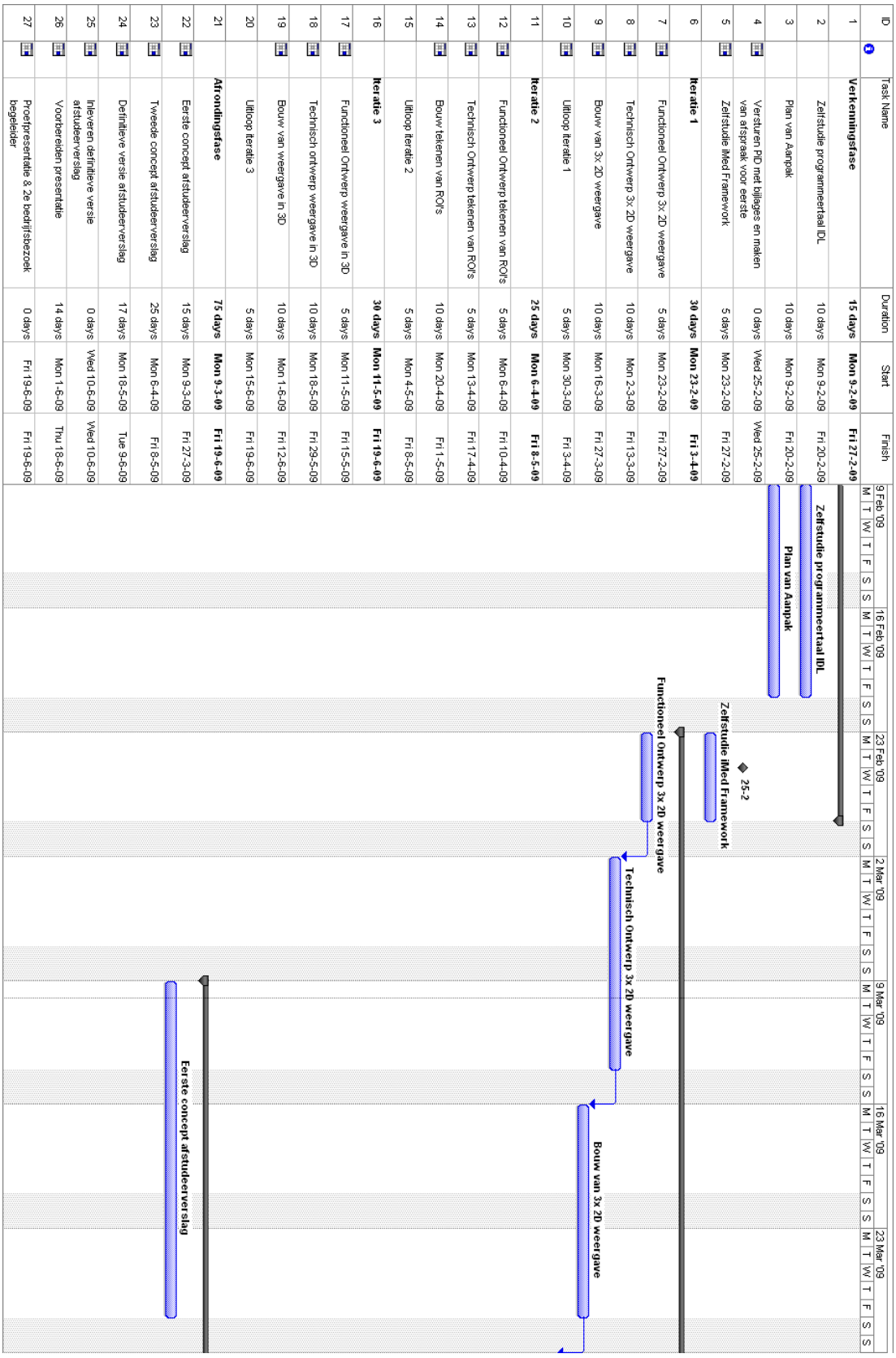
Versie	Datum goedkeuring	Naam	Functie	Paraaf
1				

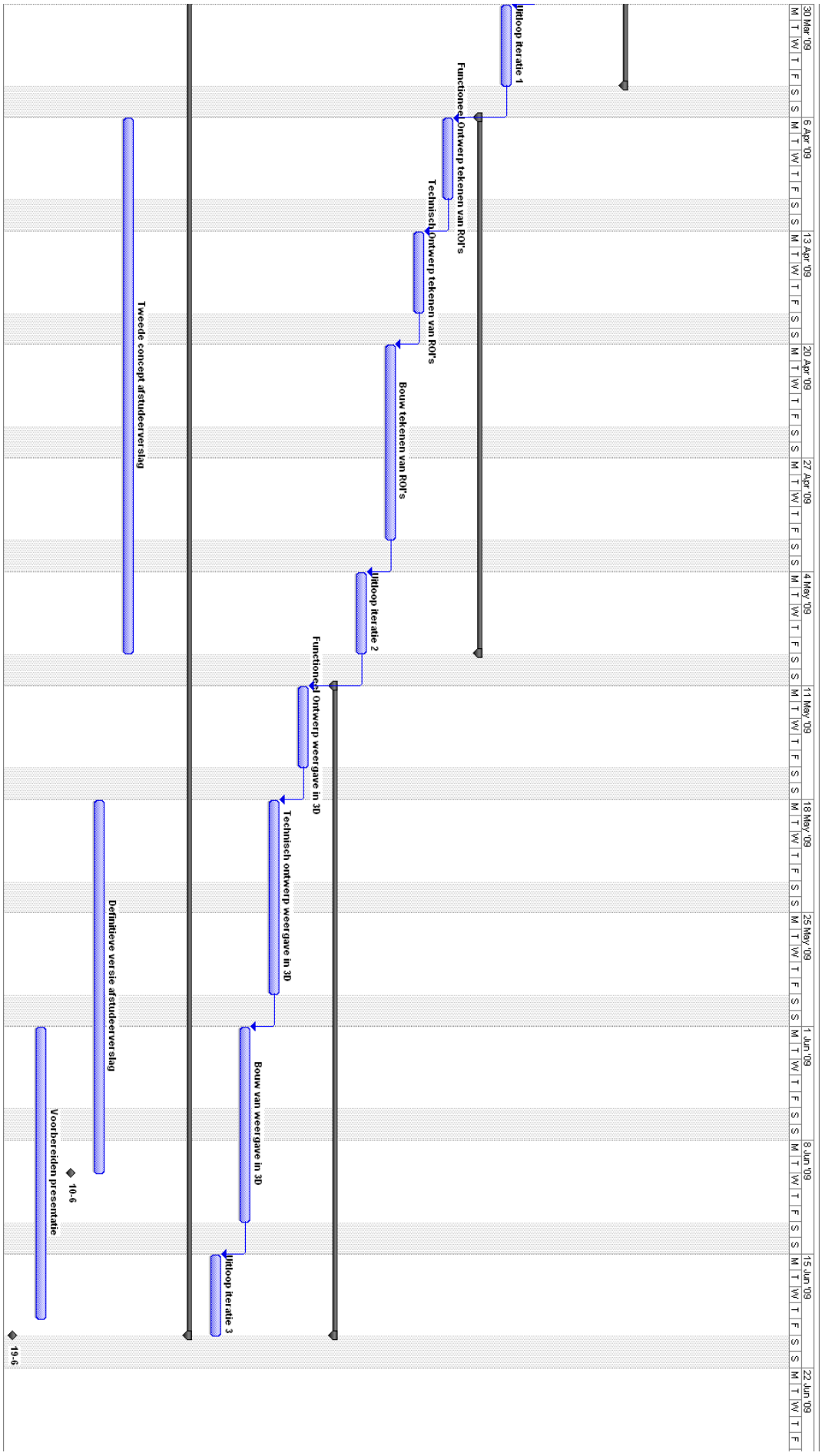
1.1.12. Distributie

Dit document is verstuurd aan:

Versie	Datum verzending	Naam	Functie
0.3	16-02-2009	Ing. L. de Groot & Ing. E. Janssen	Stagebegeleiders
0.4	20-02-2009	Ing. L. de Groot & Ing. E. Janssen	Stagebegeleiders

In deze bijlage worden de stappen in het ontwikkelproces aangegeven. Hierbij is een afbeelding (gantt-chart) gemaakt waarin visueel te zien is hoe lang een fase duurt, en welke taken elkaar overlappen.




30 Mar '09
6 Apr '09
13 Apr '09
20 Apr '09
27 Apr '09
4 May '09
11 May '09
18 May '09
25 May '09
1 Jun '09
8 Jun '09
15 Jun '09
22 Jun '09
Uitloop iteratie 1
Functioneel Ontwerp tekenen van ROI's
Technisch Ontwerp tekenen van ROI's
Bouw tekenen van ROI's
Uitloop iteratie 2
Functioneel Ontwerp weergave in 3D
Technisch ontwerp weergave in 3D
Bouw van weergave in 3D
Uitloop iteratie 3
Tweede concept afstudeerverslag
Definitieve versie afstudeerverslag
10-6
Voorbereiden presentatie
19-6

Bijlage B: Onderzoeksresultaten

1 Efficiëntie in arraybewerkingen

1.1 Inleiding

Normaal gesproken worden afbeeldingen in de eerste en tweede dimensie als transversaal plane opgeslagen. Zodra er een derde dimensie bij betrokken is, kan dit een tijdsframe, of een aantal planes zijn.
IDL kan voor een programmeur handig helpen met wildcards op het gebied van array handelingen. Het is dus mogelijk om eenvoudig een transversaal plane uit een array te halen. Als voorbeeld nemen we een array-object met de naam pixelData. Deze array heeft drie dimensies (128x128x39). Om de eerste afbeelding uit deze array te halen, kan men gebruik maken van de volgende code:
plane = pixelData[*,*,0]
Dit biedt als voordeel dat de programmeur geen extra loops voor een eenvoudige bewerking hoeft te schrijven. Maar werkt dit wel zo efficiënt?
Om dit te controleren is een test gedaan met twee verschillende DICOM bestanden. Het eerste bestand heeft een grootte van 128x128x39 pixels, en het tweede bestand heeft een grootte van 64x64x500 pixels. Als er wordt nagerekend hoeveel het tweede bestand groter is, blijkt dit een factor 3,2 (door: (64*64*500) / (128*128*39) uit te rekenen).

1.2 Testopstelling

Er is een test gemaakt betreffende het ophalen van gegevens uit een array. Hiervoor zijn twee methodes geschreven in IDL. De eerste methode zal verschillende aanzichten gaan cachen. Er worden dus 2 extra array's aangemaakt, en tijdelijk in het geheugen opgeslagen. Zodra het gevraagde aanzicht wordt opgevraagd, zullen de eerste twee dimensies wildcards zijn, en de derde dimensie een vaste waarde (bijvoorbeeld [*,*,5]). Door de twee extra array's is het dan niet meer nodig om de vaste waarde in de eerste of tweede dimensie te zetten.
De tweede methode heeft één array, en haalt door middel van vaste waarden en wildcards, zijn aanzicht op. Hierbij wordt dus een aanroep gedaan in de volgende mogelijkheden:

- [*,*,5] (transversaal)
- [*,5,*] (coronaal)
- [5,*,*] (sagittaal)

Bij beide methodes willen we de transversale, coronale en het sagittale aanzicht opvragen in 2D. De opgegeven parameters voor beide methodes zijn het aanzicht, en het frame/plane nummer. Het verschil zit dus in het cachen. Methode 1 heeft alle informatie beschikbaar, en gebruikt alleen de 3^{e} dimensie als vaste waarde, terwijl bij methode 2 alle dimensies mogelijk gebruikt kunnen worden als vaste waarde. Methode 2 wordt dus ook wel "on the fly" genoemd, omdat deze geen caching gebruikt, en dus niets voorbereid.
Voordat de methode wordt aangeroepen wordt via het commando SYSTIME(/SECONDS) opgevraagd wat de huidige tijd is (als unix timestamp, in seconden.milliseconden). Direct na het aanroepen van deze methode wordt dit nogmaals gedaan, en wordt bepaald wat de tijdsduur was.

1.3 Uitvoering

Bij methode 1 is 20x de informatie opnieuw ingelezen, per verschillend aanzicht. Er is dus 2x keer opnieuw een cache gemaakt van de aanzichten. Dit is voor beide bestanden gedaan.
Bij methode 2 is het gehele bestand in de opgegeven weergave doorgelopen. Dit is 3x per weergave gedaan (3 runs) om redelijk betrouwbare informatie te krijgen. Dit is ook weer voor beide bestanden gedaan.
De meetresultaten hiervan zijn in een Excel document opgeslagen.

1.4 Resultaat

In beide methodes is het berekenen van het transversaal vlak bijna onmeetbaar met de systeemtijd. Caching wordt hier namelijk niet op toegepast aangezien dit al vanuit de dicom bestanden in de juiste x-y-z volgorde wordt aangeleverd. Bij on the fly in het transversaal vlak wordt [*,*,5] toegepast, en aan de resultaten te zien kost dit bijna geen snelheid (dus weinig processorkracht). Bij een groot bestand zien we wel een aantal keer dat het tijd kost, maar dit kan ook door externe factoren zijn beïnvloed (bijvoorbeeld multitasking van de processor).
Het coronaal vlak berekenen duurt bij beide bestanden langer. In de 2e methode (on the fly) wordt iedere keer gebruik gemaakt van [*,5,*]. Wat hierbij wel uit de resultaten komt is dat als met methode 1 door het bestand wordt gescrolld, de tijd voor het scrollen 0 is, zodra het aanzicht is gecached.
Het sagittaal vlak berekenen duurt bij beide bestanden berekenen nog langer als het coronaal vlak. In de 2e methode (on the fly) wordt hier gebruik gemaatk van [5,*,*]. Ook komt hier weer uit dat zodra bij methode 1 het aanzicht is gecached, de tijd voor het scrollen door de planes 0 is.

1.5 Conclusie

De arraybewerkingen waarbij de laatste dimensie als vaste waarde wordt gebuikt (bijvoorbeeld [*,*,5]), is het meest efficiënt voor IDL. Als tussen de twee bestanden wordt vergeleken hoeveel hoger de tijdsfactor ligt tussen 128*128*39 en 64*64*500, blijkt dit een factor 3,37 te zijn. Als dit wordt vergeleken met de grootte-factor van de arrays, zijn deze bijna gelijk. Om te controleren of deze factoren gelijk zijn, zou er meer data en metingen nodig zijn.
Aangezien bij een tijd van 0,2 seconden de vertraging zichtbaar wordt (het plaatje veranderd niet snel genoeg als men door de planes wordt gescrolld), zal het bij grote bestanden (of bij een extra dimensie) problemen kunnen opleveren.
De waarde 0,2 is uit de test gekomen, aangezien bij de eerste aanroep van de cache-methode de cache-array pas wordt opgebouwd. Dit was zichtbaar bij het tekenen van het plane/frame op het scherm.

2 IDL versus C# en Java

2.1 Inleiding

IDL is een programmeertaal van de 4e generatie. De taal wordt wetenschappelijk gebruikt voor het uitvoeren van grote berekeningen. Hiervoor zou deze taal zeer geschikt zijn. Voor de programmeur heeft deze taal een aantal syntax vereenvoudigingen waardoor bepaalde basisfunctionaliteit makkelijker geprogrammeerd kan worden. Vooral op het gebied van arrays zijn deze vereenvoudigingen duidelijk merkbaar als men regels code gaat vergelijken.
Verder is IDL gebouwd op de taal C (objective-C). Hierdoor is het mogelijk om stukken C-code in een IDL applicatie aan te roepen. Door het gebruik van C als basis van de programmeertaal, zou deze de kenmerken van C bevatten (zeer snel, maar zelf geheugen managen). Wel zijn er een aantal vereenvoudigingen gemaakt die voor een programmeur mogelijk ongewenst zijn. Zo is multithreading beperkt beschikbaar. De programmeur kan zelf niet aangeven of er objecten of methodes zijn die een aparte thread moeten krijgen. De makers van IDL hebben hier extra niet voor gekozen om deadlocks, livelocks, starvation etc. uit te sluiten. Bovendien zou het er voor zorgen dat IDL op vele plaatsen herschreven dient te worden.
Zodra de voor- en nadelen van IDL met elkaar worden vergeleken, is het verstandig om ook andere programmeertalen hierbij te betrekken. Deze programmeertalen komen uit de 3e generatie, en hebben meer kennis van programmeren nodig dan 4e generatie programmeertalen. In dit rappport zal ik IDL vergelijken met C# en Java (jdk/jre versie 1.6). Java is volgens TIOBE Software (een onafhankelijke kwaliteitscontroleur van software) de nummer 1 als meest gebruikte programmeertaal volgens de index van April 2009[1]. C# blijkt volgens TIOBE op plaats 7 te staan, maar is stijgende.
Java en C# worden vaak met elkaar vergeleken waarbij geen echte winnaar tussen beide wordt aangewezen. Er wordt vaak beweerd dat C# sneller is dan Java, terwijl de Java programmeurs dit steevast ontkennen. Daarom zijn beide talen in dit rapport opgenomen, zodat dit ook met elkaar vergeleken kan worden.

[1] http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

2.2 Voor- en nadelen van de programmeertalen

In deze paragraaf wordt beschreven wat de voor- en nadelen van de verschillende programmeertalen zijn. Deze worden in onderstaande matrix weergegeven. Voor Java wordt hierbij eclipse als IDE[2] gebruikt. Voor IDL wordt versie 6.3 (VM en IDE) gebruikt, en voor C# wordt Visual Studio 2008 gebruikt met .NET framework versie 3.5.

	IDL	Java (eclipse)	C#
Grafische editor	Ja	Nee (wel via plug-ins)	Ja
Vereenvoudig gebruik van arrays	Ja	Nee	Nee
Gebruik van collections (ArrayList, HashMap, HashSet, LinkedList etc.)	Nee (zelf schrijven)	Ja	Ja
Eigen implementatie Multithreading	Nee	Ja	Ja
Gebruik van Interfaces, static methoden, singleton model	Nee	Ja	Ja
Aanwezigheid van DICOM functies	Ja	Nee (via libraries)	Nee (via libraries)
Aanwezigheid van ROI / VOI functies	Ja	Nee (via libraries)	Nee (via libraries)
Aanwezigheid van 3D omgevingen	Ja	Ja (via Java3D framework)	Niet bekend
Score:	**5,5**	**5,5**	**5**

In bovenstaande tabel is te zien dat de talen allemaal hun voor- en nadelen hebben, maar dat dit aantal voordelen voor bijna alle talen gelijk is. Er is een klein voordeel voor Java en IDL. Bij java zijn bijna alle punten bij te werken met libraries of plug-ins op de IDE. IDL maakt geen gebruik van libraries. Hiervoor zal de programmeur zelf stukken code moeten invoegen in het project.
De score is geteld door alle "ja" antwoorden 1 punt te geven, en alle "nee" antwoorden die via een extra optie mogelijk zijn 0,5 punt te geven.
In deze tabel is geen wegingsfactor meegenomen. Dit ligt namelijk voor iedere persoon anders, en is afhankelijk van smaak. Geeft iemand de voorkeur voor eenvoudig gebruik van arrays, dan zal IDL stijgen in aandeel. Geeft iemand de voorkeur aan het gebruik van multithreading of interfaces, dan zullen Java en C# een voordeel krijgen.

[2] IDE staat voor Integrated Development Environment. Dit is het programma (omgeving) waarin applicaties voor een bepaalde taal worden geschreven.

2.3 Benchmarking

Bij deze benchmark is gemeten wat de benodigde tijd is voor het alloceren en vullen van arrays in de drie verschillende programmeertalen. Aangezien IDL een syntax-vereenvoudiging bij het vullen van arrays heeft, zijn er twee resultaten voor IDL gemaakt. Een versie gebruikt de syntax-vereenvoudiging (wildcards, bijvoorbeeld arrayVar[*,*,*,*] = 20), en de andere gebruikt een manier met for-loops. Alle arrays hebben 4 dimensies met de grootte 128x128x47xn. Hierbij wordt n gebruikt met de waardes 64, 100, 150 en 200. De keuze van de eerste drie dimensies is tot stand gekomen omdat dit een maat is voor een bepaald type dicom-scans. De waarde 64 is het maximaal aantal elementen in de 4e dimensie waarbij het on-the-fly converteren van de array geen storende factor is voor de eindgebruiker binnen IDL (zie hoofdstuk 1 van deze bijlage).

	IDL	IDL Loops	C#	Java
Alloceren n=64	5,809000020	5,809000020	0,010014400	11,114000000
Vullen n=64	16,563000240		23,543854400	5,077000000
Totaal:	**22,372000260**		**23,553868800**	**16,191000000**
Alloceren n=100	9,372000000	9,372000000	0,030043200	14,560000000
Vullen n=100	26,270000550	248,124000000	37,033251200	7,872000000
Totaal:	**35,642000550**		**37,063294400**	**22,432000000**
Alloceren n=150	14,530999850	14,530999850	0,030043200	22,196000000
Vullen n=150	45,757000200		55,840294400	12,448000000
Totaal:	**60,288000050**		**55,870337600**	**34,644000000**
Alloceren n=200	19,207999660	19,207999660	0,060086400	56,887000000
Vullen n=200		499,938000000	76,153227200	27,930000000
Totaal:		**519,145999660**	**76,213313600**	**84,817000000**

Als resultaat van deze test kunnen we stellen dat Java het snelst is, t/m een maximale arraygroote van 128x128x47x150 integers.

In de verkregen (ruwe) data bij n=200 is te zien dat na 2 snelle allocaties weer 1 langzame komt bij Java. Aangezien ook gestart wordt met een lang durende allocatie ontstaat het vermoeden dat na 3 array's de garbage collector van Java zijn dienst gaat doen, en voor een vertraging zorgt omdat de geheugenadressen vanuit de JVM (Java Virtual Machine) terug zijn gegeven aan het OS (besturingssysteem). Dit kan weer als gevolg zijn dat Java ineens langzamer is bij een array waarbij n=200.

Java, 128x128x47x200
4,988000000
1,723000000
1,582000000
4,917000000
1,772000000
1,693000000
4,346000000
1,742000000
1,692000000
4,377000000
1,596000000
1,683000000
3,945000000
1,653000000
1,642000000
3,835000000
1,633000000
1,623000000
3,385000000
1,593000000
1,792000000
3,675000000

Een leuk verschil is ook te zien bij het gebruik van for-loops in IDL. Deze blijken een stuk trager dan het gebruik van wildcards (*). Bij n=100 is bij het vullen te zien dat de loops een factor 9 trager zijn dan de wildcards. Wat wel opvalt is dat het vullen van een array bij n=200 niet meer mogelijk is met wildcards. Hier ontstaat een geheugenerror omdat er niet genoeg geheugen beschikbaar is. Bij het gebruik van for-loops is het wel mogelijk n=200 te gebruiken. Waarschijnlijk zal IDL het vullen van (een gedeelte van) een array anders uitvoeren dan in deze test wordt gedaan.
Verder moet bij deze benchmark vermeld worden dat IDL integers van 16-bit gebruikt, en dat C# en Java integers van 32-bit gebruiken. Er is een test gedaan met het alloceren en vullen van long-objecten in IDL (32-bit, n=100). Het vullen duurde 10 seconden langer bij een 32-bit object in IDL. Het alloceren duurde ook 10 seconden langer. In Java is een test gedaan met het gebruik van short objecten (16-bit, n=100). Hierbij duurde het vullen nagenoeg net zo lang als bij een 32-bit array, maar duurde het alloceren 4 seconden korter.
Hierdoor is ook een verschil aan te tonen tussen het alloceren en schrijven van arrays in C# en Java. Java zal eerst alle geheugenplaatsen initialiseren met een bepaalde waarde. Terwijl C# de plaatsen claimt, maar hiermee verder nog niets doet. Dit is af te leiden uit de tabel hierboven. De reserveringstijd bij C# blijft nagenoeg gelijk, terwijl deze bij Java oploopt.

2.4 Conclusie

Op het gebied van functionaliteit zijn de programmeertalen nagenoeg gelijk. Het grootste verschil zit in de (technische) uitvoering van de talen. Hieruit blijkt dat IDL niet de snelste programmeertaal is. De afweging zit in het gebruik voor C# of Java. C# is sneller als er grote arrays aangemaakt moeten worden, bij het werken met de arrays is deze langzamer dan Java. Omdat arrays buiten declaratie ook gebruikt moeten worden, zal Java beter uit de bus komen. Java is in feite trager bij het alloceren van geheugenplaatsen, maar zodra 2x een aanroep wordt gedaan op de gehele array, blijkt deze sneller te zijn.
In principe kan op één benchmark nog geen volledige conclusie getrokken worden, maar aangezien het een basisfunctionaliteit betreft, kan er wel een inschatting gemaakt worden. In dit geval betekend dit dat Java als snelste uit de bus komt, gevolgd door C#.
Toch heeft IDL nog een groot voordeel. Dat is de manier waarop IDL met DICOM bestanden en weergaves wordt omgegaan. IDL bezit namelijk meerdere objecten en functies om met DICOM bestanden te werken. Dit ondersteunen Java en C# niet uit zichzelf. Hiervoor moeten libraries gezocht worden die dergelijke functionaliteit bezitten. Als deze gevonden worden, blijft de vraag hoe efficiënt deze libraries werken. Als deze niet efficiënt geprogrammeerd zijn, kan het alsnog blijken dat IDL sneller is dan Java/C# voor een dergelijke toepassing.

Bijlage C: Functionele en Technische ontwerpen

Functioneel ontwerp iteratie 1

Inleiding

Dit document heeft tot doel de wensen en eisen van de opdrachtgever helder te formuleren, en staat aan de basis voor het technisch ontwerp van het eindproduct.

De belangrijkste reden voor het gebruik van dit document is om, met de opdrachtgever, duidelijk af te spreken wat er wel en niet in de applicatie opgeleverd wordt. Als een hierop volgende fase is afgerond (bijvoorbeeld het technisch ontwerp, of de realisatie), kan men aan de hand van dit document nakijken of alle gevraagde functionaliteit ook verwerkt is.

Aangezien het project initieel als doel heeft om het iMed framework uit te breiden, zullen de use cases beperkt blijven. Dit omdat use cases normaliter gebruikt worden om de interactie met de gebruiker te beschrijven. Als eerste zal er namelijk een standaardset aan functies en GUI elementen gemaakt moeten worden. Hier zullen wel een paar basale use cases bij horen.

Inhoudsopgave

1 Analyse

Op dit moment gebruikt men een framework om het bestaande e.Soft systeem uit te breiden. Dit framework heeft de naam iMed, en bestaat uit een library-set van meerdere klassen. Deze library set zijn losse code-bestanden, en kunnen in een project toegevoegd worden. Op dit moment is het nog niet mogelijk om met het iMed framework 3D weergaves en bewerkingen uit te voeren.

De bedoeling van de uitbreiding is om het huidige 2D model binnen het framework uit te breiden naar een mogelijk 3D model. Hieronder staan de eigenschappen waaraan dit uitgebreide model na iteratie 3 aan moet voldoen:

- De gerealiseerde uitbreidingen moeten een meerwaarde zijn op de bestaande code in het framework
- Het framework moet gemakkelijk zijn om mee te werken c.q. te programmeren
- Het framework moet geen geheugenlekken veroorzaken waardoor een pc kan vastlopen
- De gerealiseerde uitbreidingen zal het mogelijk zijn om eenvoudig VOI's[3] (Volume of Interest) te tekenen.
- Het framework dient op deze VOI's standaardberekeningen uit te kunnen voeren
- De gerealiseerde uitbreiding zal met minimaal een 2D GUI-type[4] opgeleverd moeten worden (namelijk 3 verschillende 2D vensters), en waar mogelijk met een 3D GUI-type[5].

[3] VOI is de afkorting voor een Volume of Interest. Een VOI bestaat uit meerdere ROI's (Region of Interest) die achter elkaar in meerdere planes bevinden. Hierdoor ontstaat een ROI. Aangezien dit uit drie dimensies bestaat, wordt er dan gesproken over een Volume of Interest.

[4] De 2D GUI zal bestaan uit drie verschillende aanzichten op het object, namelijk sagittaal (van de linker zijkant), coronaal (van de voorkant) en transversaal (van de bovenkant).

[5] De 3D GUI-type zal bestaan uit één venster, waarin het object in verschillende richtingen gedraaid kan worden. Hierdoor is er maar één weergave nodig.

2 Programma van Eisen

In dit hoofdstuk worden de wensen en eisen van het programma beschreven, en gecategoriseerd. Allereerst zal er een splitsing worden gemaakt in functionele eisen, en niet-functionele eisen. Hierbij zijn de functionele eisen alles wat hard aantoonbaar is. Dit is de functionaliteit van het programma. De niet-functionele eisen, zijn eisen die niet als functionaliteit aantoonbaar zijn, maar wel invloed hebben op het product.

Het categoriseren van de functionele eisen gebeurt aan de hand van de MoSCoW-methode. Hierbij worden de eisen ingedeeld in Must haves (eisen die in het product moeten), Should haves (eisen die eigenlijk wel in het product zouden moeten, maar een mindere prioriteit hebben), Could haves (eisen die niet in het product hoeven, maar die gemaakt worden als er tijd over is) en Won't haves (eisen die zeker niet in het product komen). Deze methode geeft bij een grote tijdsdruk minder conflicten als er beslist moet worden welke functionaliteit niet opgeleverd kan worden.

2.1 Functionele eisen

In deze paragraaf worden de functionele eisen van iteratie 1 beschreven. Zoals in het PiD beschreven zal iteratie 1 zich voornamelijk richten op het weergeven van de afbeeldingen in 3 verschillende, 2-dimensionale aanzichten. De eisen zijn van toepassing op het functioneren van het systeem, en niet op factoren van het systeem (zoals snelheid en beveiliging).

Must Haves

Het iMed framework dient uitgebreid te worden met ondersteuning voor 3-dimensionale afbeeldingen. Hiermee wordt bedoeld dat de relevante publieke functies/klassen van het iMed framework met een 3D dataset kunnen werken, en hiermee berekeningen kunnen uitvoeren.

De 2-dimensionale GUI dient gemaakt te worden. Hierbij zal de gebruiker 3 afbeeldingen binnen de interface te zien krijgen. Namelijk een sagittaal (van de linker zijkant), een coronaal (van de voorkant) en een transversaal (van de bovenkant) aanzicht. Deze drie aanzichten dienen met elkaar in contact te staan zoals later in de use cases zal blijken. Hierdoor zal het mogelijk zijn dat als er een aanpassing in het ene scherm gebeurt, dit effect heeft op de weergave in een ander scherm.

Should Haves

Het zetten van een minimale drempelwaarde. Waardes die zich onder de drempelwaarde begeven, worden niet meer weergegeven in de afbeelding.

Het veranderen van de kleurintensiteit. Hierbij worden alle waardes met een factor vermenigvuldigd. Hierdoor wordt het kleurverschil tussen activiteiten groter. Hierdoor kan er bijvoorbeeld een hogere nauwkeurigheid worden behaald bij het zetten van de drempelwaarde.

Onderzoeken van een 3D assenstelsel en de bijbehorende mogelijkheid tot roteren van een object wat zich bevind in dit 3D assenstelsel.

Could haves

De mogelijkheid om Volume of Interests (VOI's) te tekenen binnen de 2-dimensionale GUI. Hiermee wordt bedoeld dat de gebruiker een vakje binnen de afbeelding kan tekenen. Hierbij kan (door minimaal 2 aanzichten te gebruiken) een kubus gemaakt worden. Het zogenaamde Volume of Interest. In principe is dit voor iteratie 2. De tweede iteratie is namelijk gereserveerd voor het maken van VOI's in verschillende soorten en maten.

Won't Haves

Het gebruik van meerdere tijdsframes om door de tijd verschillende afbeeldingen te bekijken.

2.2 Niet-functionele eisen

Naast alle functionele eisen (wat moet het programma kunnen) zijn er ook eisen die niet direct aan het programma zelf worden gesteld, maar op de werking van het programma van toepassing zijn. Dit zijn de niet-functionele eisen. In dit hoofdstuk zijn deze eisen opgesomd en beschreven.

Code

De applicatie dient te worden geschreven in de programmeertaal IDL en er dient (waar mogelijk) gebruik gemaakt te worden van de aanwezige functies in het framework.
De keuze om het framework initieel in IDL te schrijven is omdat de Siemens e.Soft systemen dit externe IDL programma's op de machine kunnen starten, en dat deze binnen e.Soft zichtbaar zijn. Verder is het een krachtige taal om berekeningen op beelden te kunnen uitvoeren. Voor dit doel wordt de taal ook bij de NASA en in de meteorologie gebruikt.

Alle publieke functies/methoden dienen beschreven te worden met PRE en POST situatie. Hiermee wordt beschreven wat de input en output van een functie zal zijn. Verder zal er in het kort omschreven worden hoe de functie dit precies uitvoert.

Er wordt gewerkt met een SVN server. Dit zorgt ervoor dat iedereen (de opdrachtnemer en de bedrijfsbegeleider/consultant) altijd de actuele versie beschikbaar heeft. Als iemand aan een codefragment heeft gewerkt, zal hij deze op de server plaatsen. Dit moet met goed commentaar gebeuren. Hierdoor kan men nagaan waarom een bepaalde wijziging is gemaakt, en wanneer dat is gebeurt (en door wie).

Hardware

De applicatie zal uiteindelijk op een Siemens e.Soft systeem gaan werken. Deze systemen worden door Siemens zelf geleverd, en waar nodig wordt de hardware uitgebreid. Hierdoor wordt aan het uitgebreide framework de hardware eis gesteld dat het op de bestaande e.Soft systemen moet werken.

Performance

Natuurlijk is het streven om berekeningen zo snel mogelijk uit te voeren, aangezien een systeem anders wordt er door de gebruikers snel een negatieve stempel op geplakt. Maar als dit ten koste gaat van de kwaliteit, zal er toch gekozen worden voor een kwalitatief beter en betrouwbaarder systeem.

Veiligheid

De veiligheid berust hier vooral op privacy. Aangezien het framework wordt gebruikt om uitbreidingen te schrijven op het e.Soft systeem, zal dit punt wegvallen. De privacy van patiëntdata zal worden gewaarborgd door het e.Soft systeem zelf. Wel zal het zo zijn dat de uiteindelijke applicatie, wat met het iMed framework geschreven wordt, niet met patiëntendata moet gaan rommelen.

Taal

De taal voor de grafische kant van de uitbreidingen zal Engels zijn. Ook zullen er medische termen in voorkomen om op één lijn te blijven met de medisch onderlegde eindgebruiker van het e.Soft systeem.

Gebruiksgemak

De eindgebruiker (gericht op de GUI onderdelen) zal een basiskennis hebben van medische apparatuur. Hierdoor zullen schermvormgevingen opgebouwd worden aan hun basiskennis of de algemene standaard. Dit werkt bevorderend voor het gebruiksgemak, omdat de gebruikte systemen een zelfde soort interface hebben en dit het leerproces voor een applicatie verkort.

Betrouwbaarheid

De betrouwbaarheid van de uitbreiding zal hoog moeten zijn. Er worden namelijk allerlei berekeningen en analyses uitgevoerd op basis van de informatie die het framework geeft. Dit betekent dat de performance misschien iets lager wordt, maar dat de kwaliteit/betrouwbaarheid wel hoger is. Dit omdat applicaties die met het uitgebreide framework worden gebouwd, gebruikt kunnen worden om medische beslissingen te nemen. Als er dan structureel een fout zit in het framework kan dit consequenties opleveren voor de behandeling van een patiënt.

Handleidingen

In principe zullen er geen handleidingen voor de eindgebruiker beschreven worden. Dit aangezien het een uitbreiding op het framework is, en geen applicatie op zichzelf. Daarom zal het Technisch Ontwerp als handleiding dienen.

Documentatie

Zoals hierboven gezegd dient het Technisch Ontwerp als documentatie. Samen met dit document en het commentaar in de code. Dit commentaar zal door middel van pre en posts functies omschrijven wat een functie precies uitvoert. Bij uitgebreide functies zullen er binnen de code regels commentaar komen te staan.

3 Use Cases

In dit hoofdstuk staan alle use cases uitgewerkt, in de vorm van de MoSCoW-methode. Deze komen overeen met de eisen in paragraaf 2.1. Hierbij zijn de use cases voor de Won't Haves weg gelaten.

3.1 Must Haves

Naam:	Opbouwen van 2D weergaves
Actor:	Programmeur die het framework gebruikt
Pre:	
Normal flow:	- De actor geeft aan dat hij de 3x 2D weergave wil gebruiken
Resultaat:	- De actor krijgt drie afbeeldingen (sagittaal, coronaal, transversaal) als weergave. Deze worden real time bijgewerkt naar gelang één van de drie vensters veranderd.
Alternatieve flow:	- De actor krijgt de melding dat dit niet mogelijk is, aangezien er geen geldige DICOM file beschikbaar is. - De actor krijgt de melding dat dit niet mogelijk is, aangezien er niet genoeg planes beschikbaar zijn

Naam:	Selecteren van frames
Actor:	Programmeur die het framework gebruikt
Pre:	De 2D weergave is opgebouwd
Normal flow:	- De actor geeft aan welke weergave hij wil gebruiken - De actor geeft aan naar welk plane hij wil zoomen (doormiddel van scrollen met de muis)
Resultaat:	- De actor krijgt als resultaat dat in de opgegeven weergave naar een bepaald plane is gescrolld - In de andere aanzichten verandert de locatie van de lijn als het aanzicht hierin aan te tonen is
Alternatieve flow:	

3.2 Should Haves

Naam:	Zetten van een minimale drempelwaarde
Actor:	Programmeur die het framework gebruikt
Pre:	Er is een 3D beeld gegenereerd, of er zijn 3x 2D beelden gegenereerd
Normal flow:	- De actor geeft aan dat hij de minimale drempelwaarde wil veranderen - De actor verandert de waarde door een getal te veranderen, of door een slider te verplaatsen
Resultaat:	- De actor krijgt een nieuw gegenereerd beeld, waarbij de kleurintensiteit van kleine waardes veranderd kan zijn
Alternatieve flow:	

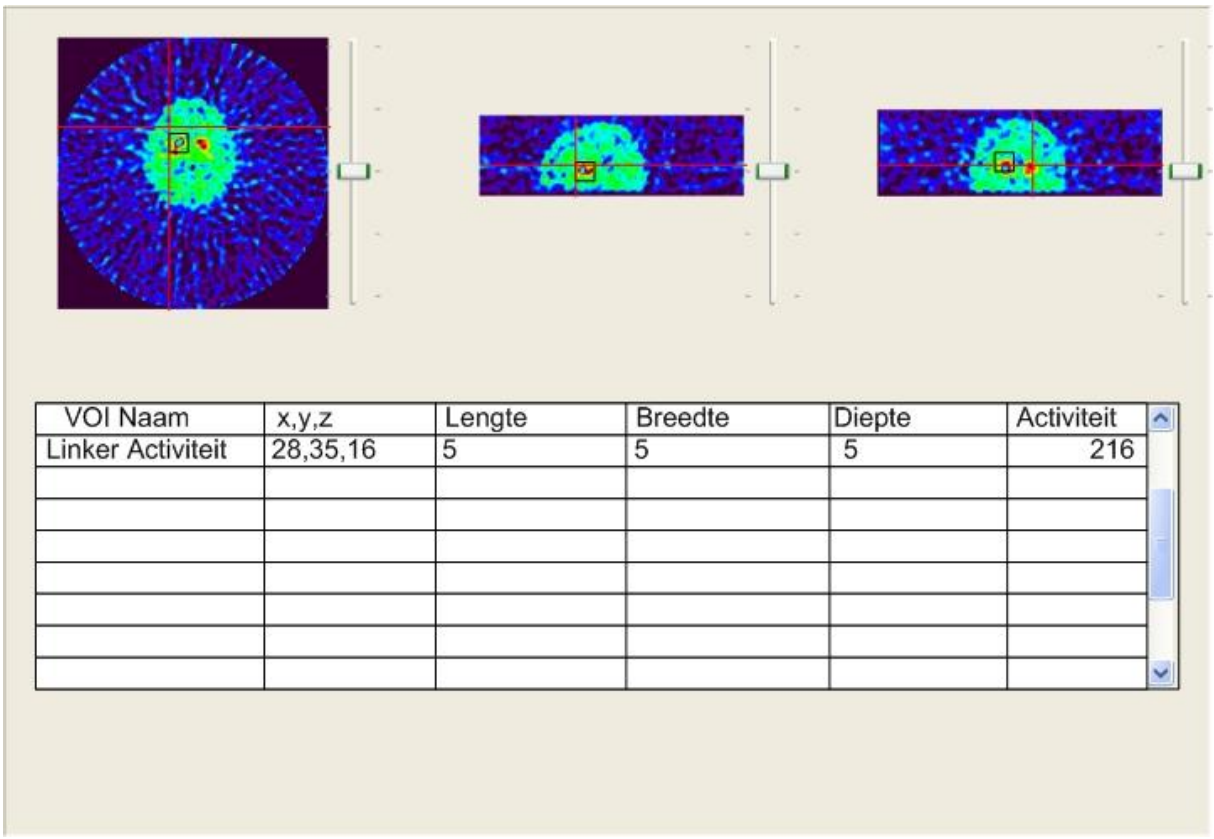
Naam:	Veranderen van kleurintensiteit
Actor:	Programmeur die het framework gebruikt
Pre:	Er is een 3D beeld gegenereerd, of er zijn 3x 2D beelden gegenereerd
Normal flow:	- De actor geeft aan dat hij de kleurintensiteit wil veranderen - De actor verandert versterkingsfactor door een getal te veranderen, of door een slider te verplaatsen
Resultaat:	- De actor krijgt een nieuw gegenereerd beeld, waarbij de kleurintensiteit veranderd kan zijn
Alternatieve flow:	

3.3 Could Haves

Naam:	Tekenen van VOI
Actor:	Programmeur die het framework gebruikt
Pre:	De beelden zijn als 3x 2D gegenereerd
Normal flow:	- De actor geeft aan welke weergave hij wil gebruiken - De actor geeft aan dat hij een ROI wil gaan tekenen - De actor trekt een vrije vorm op een aangegeven positie binnen de weergave - De actor selecteert een andere weergave - De actor trekt in deze weergave de ROI door zodat het een kubus wordt
Resultaat:	- De actor krijgt als resultaat dat er een kubus is gemaakt door middel van lijnen. Deze is in de verschillende weergaves te zien
Alternatieve flow:	

Naam:	Berekenen activiteit binnen VOI
Actor:	Programmeur die het framework gebruikt
Pre:	Er is een 3D ROI getekend
Normal flow:	- De actor geeft aan dat hij de activiteit binnen de ROI wil weten
Resultaat:	- De actor krijgt als resultaat de waarde van de activiteit, gedefinieerd in aantal en in procent ten opzichte van de gehele afbeelding.
Alternatieve flow:	

4 Schermvoorbeeld



VOI Naam	x,y,z	Lengte	Breedte	Diepte	Activiteit
Linker Activiteit	28,35,16	5	5	5	216

In de bovenstaande afbeelding is te zien wat de eerste 2D versie van het framework moet kunnen, als alle must, could en should haves zijn geïmplementeerd. Het moet namelijk mogelijk zijn om dan een VOI te tekenen. Hiervoor zijn minimaal 2 aanzichten nodig. In dit voorbeeld is echter voor drie aanzichten gekozen, aangezien het dan voor de eindgebruiker waarschijnlijk makkelijker is zich te oriënteren.

Verder worden er lijnen in de afbeelding aangegeven. Deze dienen ter verduidelijking waar de weergaves zich bevinden. In de transversale weergave (links) kan men de locatie van de sagittale (midden) en coronale (rechts) weergaves duidelijk maken. In de sagittale weergave worden de transversale en coronale weergave aangeduid, en in de coronale weergave worden de transversale en sagittale weergave aangeduid.

Deze weergaves kunnen verplaatst worden door met de sliders aan de zijkant te bewegen, of met het scrollwiel van de muis. Hierbij zullen de betreffende lijnen ook van plaats verschuiven. Verder is het ook mogelijk om een optie aan te zetten zodat de muis gevolgd wordt binnen het venster, waardoor de andere twee vensters automatisch veranderen.

Onder in de tabel zijn de VOI's beschreven. Deze VOI's worden aangemaakt door een vierkant te tekenen in het aanzicht. Hierdoor verschijnt in het/de ander(e) aanzicht(en) een lijntje wat uitgetrokken kan worden. Als gevolg hiervan ontstaat een kubus. Dit is dan een Volume of Interest waarmee in drie dimensies de activiteit kan worden bepaald.

Technisch ontwerp iteratie 1

Inleiding

Dit document heeft tot doel om een heldere beschrijving te geven over de aanpassingen voor het iMed framework. Dit document zal ook als leidraad dienen bij de daadwerkelijke bouw/aanpassing van het framework.

De belangrijkste reden voor het gebruik van dit document is om, met de bedrijfsbegeleider, duidelijk af te spreken welke aanpassingen er gemaakt worden. Deze aanpassingen zullen er voor dienen om de wensen en eisen uit het Functioneel Ontwerp te realiseren.

Inhoudsopgave

1 Het platform

Het platform waarop het iMed framework wordt gebruikt zijn de Siemens e.Soft workstations. In principe zijn dit normale computers. Hierbij kan gedacht worden aan een computer wat heden ten dage in de computerwinkel te koop is, met gemiddelde specificaties.

Deze computers werken op dit moment met Windows XP als besturingssysteem. Hierop is een applicatie gebouwd door Siemens. Deze applicatie, genaamd e.Soft, verzorgt het aansturen van een PET-scanner, en het genereren en weergeven van de DICOM-images.

Met deze DICOM-images is het mogelijk om, in de Siemens software, analyses uit te voeren. Veel van deze analyses zijn al aanwezig in de applicatie, maar het is ook mogelijk om zelf analyses toe te voegen. Deze kunnen in IDL worden geschreven. Hiervoor is er een Virtual Machine beschikbaar. Deze Virtual Machine wordt bij het Siemens Workstation meegeleverd. Deze Virtual Machine hoort bij IDL, en is speciaal gebouwd om IDL code uit te voeren als een applicatie.

De studie (geschreven in IDL) wordt dan aangeroepen door het e.Soft programma, en zal ook in een venster van het e.Soft programma worden weergegeven.

2 Gekoppelde bestanden en applicaties

Zoals in de vorige paragraaf is uitgelegd, wordt de IDL-studie software door e.Soft gestart. Het e.Soft programma zal dan ook de benodigde files meegeven om de studie uit te voeren.

Zodra een studie is afgerond, is het mogelijk om met het iMed framework screenshots te maken van resultaten. Deze kunnen dan als afbeeldingen bij de studie worden opgeslagen.

Aangezien deze functionaliteit al in het framework aanwezig is, en niet aangepast hoeft te worden (zie Functioneel Ontwerp). Zal dit document hier niet verder op ingaan.

3 Software

Zoals in het Functioneel Ontwerp is uitgelegd, zal de software in IDL worden ontwikkeld. Als ontwikkelomgeving zal hiervoor een standaard computer van het UMC St. Radboud gebruikt worden. Hierop zal IDL 6.3 van ITT VIS geïnstalleerd worden. Deze computer zal dus als ontwikkelomgeving gebruikt worden.

Het huidige iMed framework zal als resource gebruikt worden. Dit omdat de huidige functionaliteit behouden moet blijven, en de functies hiervan wellicht in de uitbreiding gebruikt kunnen worden. Ook zal het nieuwe framework in zijn geheel moeten samenwerken. Het mag niet mogelijk zijn dat door de nieuwe functionaliteit sommige functies niet meer kunnen samenwerken.

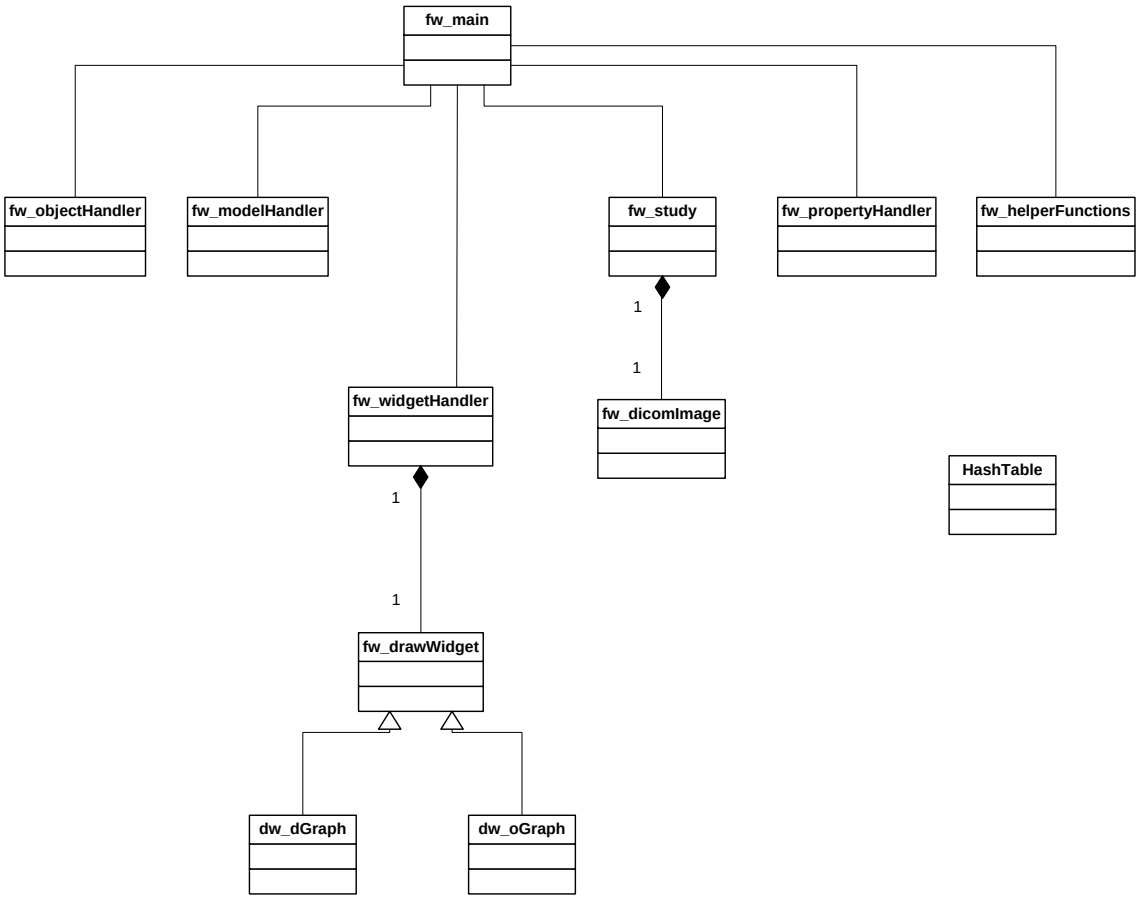
4 Programmastructuur

Dit hoofdstuk is ingedeeld in twee paragrafen. Allereerst zal de huidige structuur worden besproken. Hiervan zijn een klassendiagram en drie sequentiediagrammen gemaakt. Het klassendiagram geeft de globale structuur van de applicatie. Welke objecten er mogelijk aangemaakt worden, en welke verhouding deze tot elkaar hebben. De sequentiediagrammen zijn gemaakt om inzicht te geven in de flow van de applicatie, en hoe de objecten binnen de applicatie gebruikt worden.

4.1 Huidige Situatie

4.1.1 Klassendiagram

Hieronder in het figuur staat het huidige domeinmodel. Dit model is beperkt tot alleen de naam van de klasse, en de relaties van de objecten. Dit is gedaan omdat het klassendiagram niet binnen de afmetingen van dit document zou passen. Onder het figuur staat uitgelegd wat de functie van de verschillende klassen zijn.



Figuur 1: Domeinmodel

fw_main: dit is het basisobject van het framework. Hierin worden de links (pointers) naar de objecten fw_objectHandler, fw_modelHandler, fw_widgetHandler, fw_study, fw_propertyhandler en fw_helperfunctions bijgehouden. Dit object moet binnen de applicatie beschikbaar zijn, om het aanroepen van het framework mogelijk te maken.

fw_modelHandler: In IDL is het mogelijk om gebruik te maken van Object Graphics. Hierbij worden objecten in een virtuele 3D wereld gemaakt, en worden deze via een view gepresenteerd. Fw_modelHandler zorgt ervoor dat objecten aan een model gekoppeld worden, en dat deze worden onderhouden. Deze modellen kunnen vervolgens aan de view van dw_oGraph gekoppeld worden. Dit koppelen gebeurt dan in de widgetHandler.

fw_widgetHandler: Deze klasse beheert de views die gekoppeld worden aan de models uit fw_modelHandler. Als er via fw_widgetHandler een nieuw object wordt aangemaakt (bijvoorbeeld een ROI, een lijn, of dergelijke), wordt deze doorgestuurd naar fw_modelHandler, om dit aan het juiste model te koppelen. Ook regelt deze klasse de communicatie als een ROI op meerdere views zichtbaar moet zijn, en als een nieuwe ROI gemaakt wordt. Deze klasse geeft dan aan de klasse van het type fw_drawWidget door dat er een refresh moet worden uitgevoerd.

fw_drawWidget: Deze klasse is een 1:1 container met het WIDGET_DRAW uit IDL. Deze klasse beheert extra informatie over het widget, en is daardoor binnen het framework altijd aangeroepen zodra er een WIDGET_DRAW nodig is. Verder dient deze klasse als basisklasse voor dw_oGraph en dw_dGraph.

dw_dGraph: Deze klasse erft van fw_drawWidget, en wordt gebruikt voor objecten van het type WIDGET_DRAW, die gebruik maken van direct graphics. Deze modus biedt minder grafische mogelijkheden, maar is wel sneller bij het tekenen van afbeeldingen. Als voorbeeld van de beperkte mogelijkheden is er geen mogelijkheid om eigen onderdelen (bijvoorbeeld ROI's) te tekenen of te bewerken met direct graphics. Hierdoor heeft deze klasse binnen het framework minder functies en mogelijkheden.

dw_oGraph: Deze klasse erft van fw_drawWidget, en wordt gebruikt voor objecten van het type WIDGET_DRAW, die gebruik maken van Object Graphics. Hierbij wordt een model (vanuit fw_modelHandler) gekoppeld aan deze view. Ook worden hier de events van het widget afgehandeld (bijvoorbeeld de muisevents bij het tekenen van een Region of Interest). Het gebruik van object graphics (gekoppeld met een eigen model) brengt wel als nadeel met zich mee dat het meer geheugen en snelheid kost, dan de direct graphics. Daarom dient men een keuzeafweging te maken voor het gebruik van direct of object graphics. Het maken van deze afweging gebeurt op applicatie-niveau.

fw_study: Deze klasse zorgt ervoor dat een (of meerdere) DICOM bestand(en) in de applicatie wordt ingeladen. Aangezien een DICOM bestand meer informatie bevat dan alleen de afbeeldingen, worden deze in de study klasse afgehandeld. Zo kan er uit de headers informatie worden gelezen betreffende de naam van de patiënt, scanner instellingen en andere specifieke informatie betreffende het bestand.

fw_dicomImage: In deze klasse worden de afbeeldingen uit het DICOM bestand bewaard. Verder is het hier mogelijk om een aantal basisfuncties op de afbeeldingen uit te voeren. Deze afbeeldingen zijn in principe gewoon 2 of 3 dimensionale array's met getallen.

fw_propertyHandler: De propertyHandler maakt het mogelijk om d.m.v. keywords (eventueel overeenkomstig met de velden binnen een object) allerlei informatie op te slaan. Als er een veld met dezelfde naam beschikbaar is binnen het object, zal de waarde van de eigenschap (property) in het betreffende veld worden opgeslagen. Is het opgegeven veld niet beschikbaar binnen het object (bijvoorbeeld bij een nieuwe, of tijdelijke property), zal dit als referentie in een globale HashTable worden opgeslagen. Als alle properties gezet en gelezen worden via deze propertyHandler, is het dus mogelijk om in een applicatie (ontwikkeld met het framework) extra informatie op te slaan. Dit bied als voordeel dat het framework flexibel blijft, en dus niet voor iedere applicatie structurele aanpassingen behoeft.

fw_helperFunctions: Deze klasse bestaat uit allerlei hulpfuncties. Dee hulpfuncties maken gebruik van de opbouw van het framework. Hier zit bijvoorbeeld een functie in om een Region of Interest te splitsen.

HashTable: Deze klasse gedraagt zich vergelijkbaar als (bijvoorbeeld) een HashMap in Java. Uitgebreide soorten collecties (zoals HashMap, HashSet, ArrayList, Linked List etc.) zijn namelijk niet beschikbaar in IDL, en dienen zelf geschreven te worden.

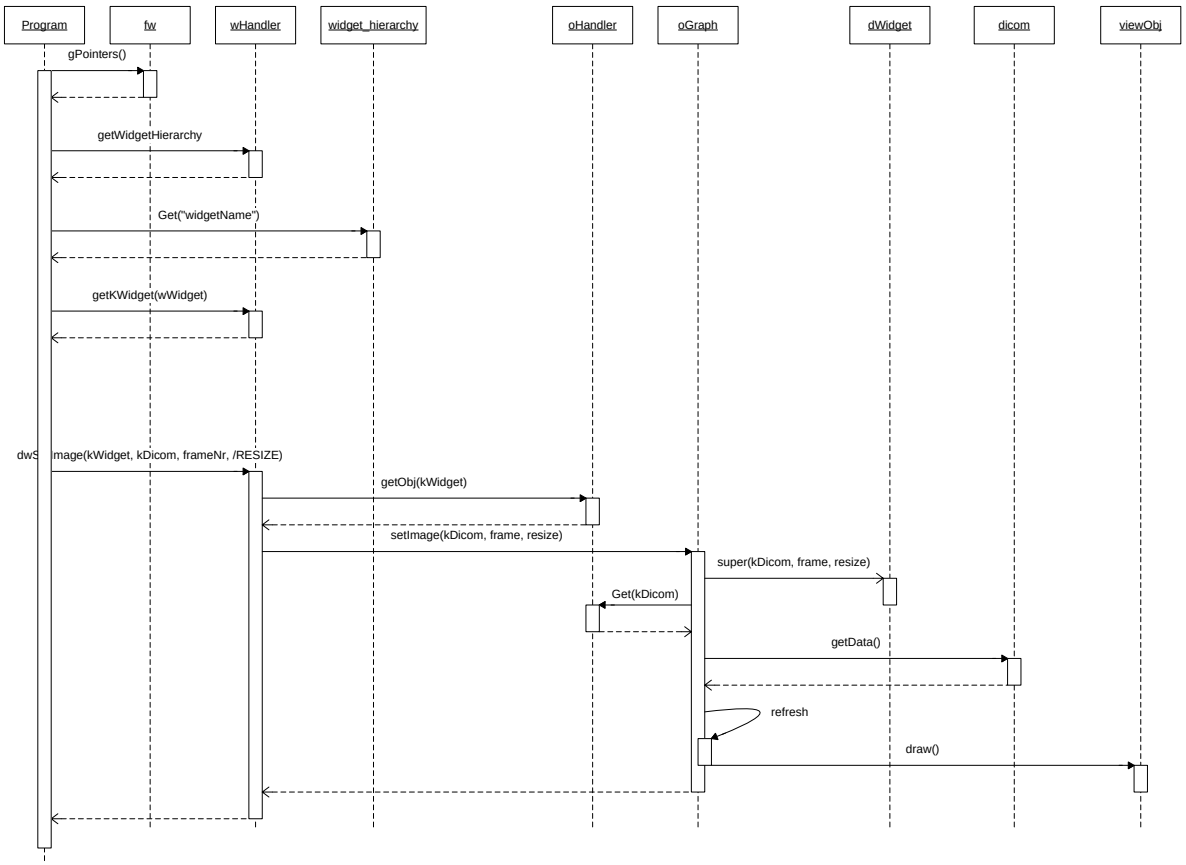
Fw_objectHandler: Deze klasse is de grote container voor het framework. Hierin wordt in een HashTable een sleutel, en de pointer van een object opgeslagen. Dit heeft als voordeel dat objecten altijd terug te vinden zijn, aangezien een hashTable een lijst met sleutels kan teruggeven.

4.1.2 Sequentie Diagrammen

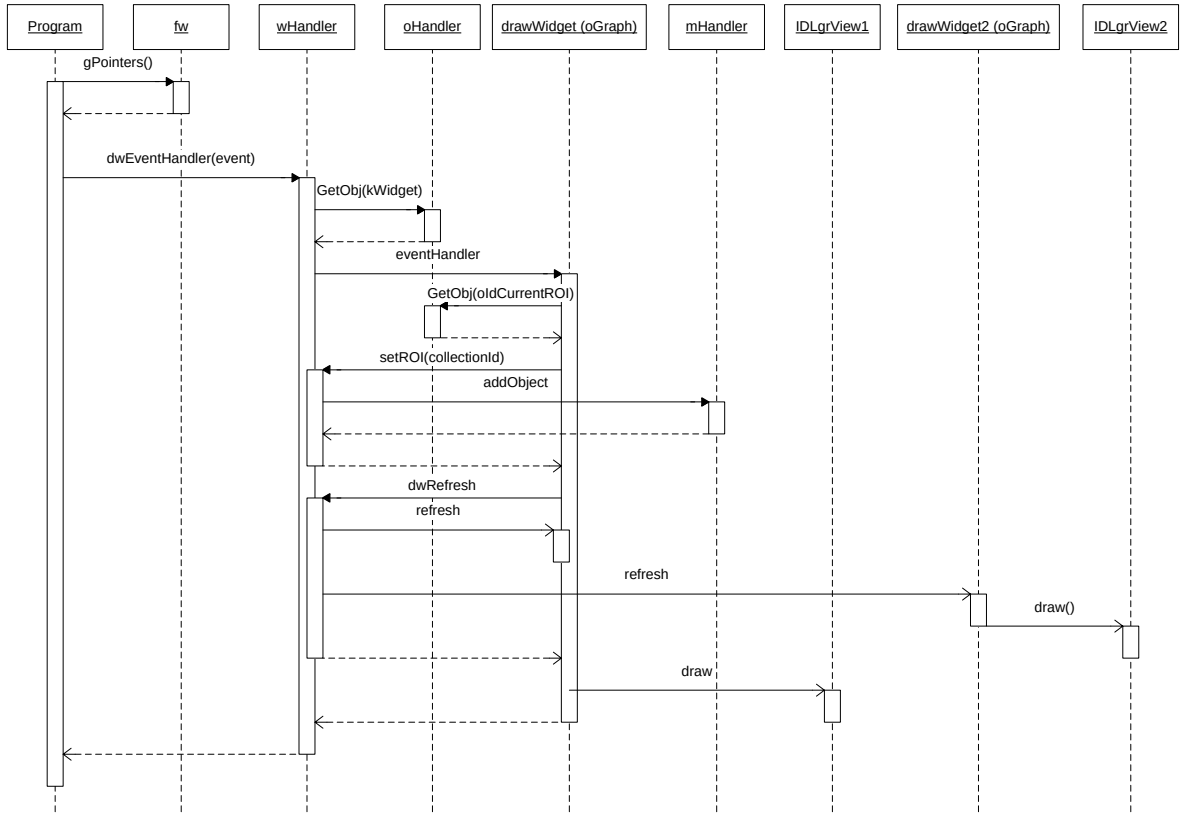
Hieronder staan drie sequentiediagrammen. In het eerste diagram is beschreven hoe het programma wordt doorlopen als er een nieuwe afbeelding via object graphics wordt ingeladen, en hoe deze wordt gekoppeld aan een widget (en model).

In het tweede diagram is beschreven hoe een event tijdens het tekenen van een ROI wordt afgehandeld. In deze sequentie wordt een nieuwe ROI gemaakt, en aan het bestaande model gehangen.

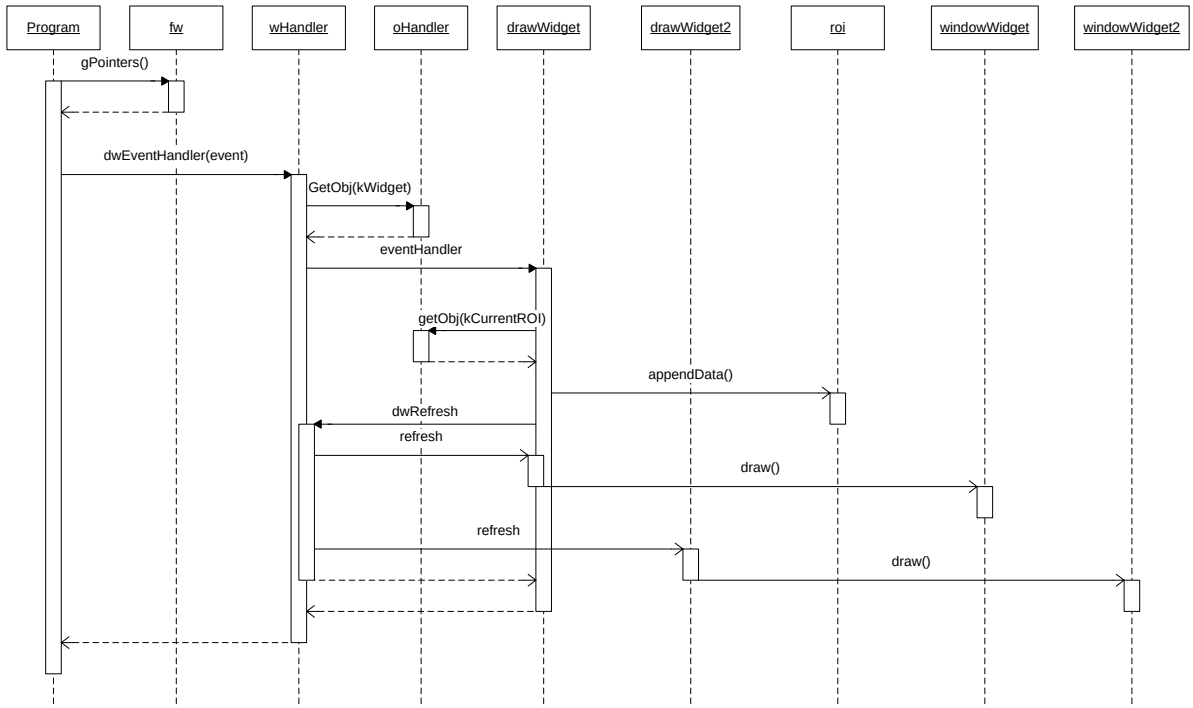
In het derde diagram is beschreven hoe een event tijdens het bijwerken van een ROI wordt afgehandeld. Hiermee wordt bedoeld wat er met het event gebeurt als de gebruiker de muisknop ingedrukt houdt, en met de muis beweegt. Hierbij word een nieuw coördinaat aan de ROI meegegeven.



Figuur 2: Laden van een afbeelding



Figuur 3: drawWidget event, maken van een nieuwe ROI



Figuur 4: drawWidget event, bewerken van een bestaande ROI

4.2 Nieuwe Situatie

Als er wordt gekeken naar het klassendiagram en de sequentiediagrammen valt het meteen op dat in beide situaties de klasse fw_widgetHandler (in de sequentiediagrammen wHandler genoemd) een grote rol speelt. Deze regelt in feite de communicatie richting de drawWidgets (fw_oGraph en fw_dGraph).

Bij de klasse fw_drawWidget is het verstandig om een veld erbij te maken, waarin wordt opgeslagen wat voor type aanzicht het widget heeft (bijvoorbeeld self->Get('sight') EQ "transversal", self->Get('sight') EQ "coronal" of self->Get('sight') EQ "sagittal"), en bij welke widgetcollectie het widget hoort.

Hierdoor weet het fw_widgetHandler object welk type aanzicht wordt weergegeven in het betreffende fw_drawWidget, en kan deze een signaal sturen naar de andere twee fw_drawWidget objecten, zodat deze hun positie-lijnen kunnen aanpassen en zichzelf opnieuw kunnen tekenen.

Door de methode setImage van fw_oGraph of fw_dGraph aan te passen, is het mogelijk dat deze methodes (gebaseerd op het veld self->Get('sight')) een aanroep doen naar de dicom-klasse waarbij ze het aanzicht opgeven. Hierop kan de dicom-klasse de juiste 2D afbeelding teruggeven.

Aangezien setImage van fw_oGraph of fw_dGraph wordt aangeroepen via de widgethandler (wHandler->dwSetImage), kan de widgethandler de andere twee widgets (oGraph clients) aansturen om hun referentielijn van een bepaald aanzicht bij te werken. Met als gevolg dat deze weer een aanroep doen aan de setLine methode om een lijn toe te voegen, of een bestaande lijn aan te passen. In het betreffende oGraph (client-)object kan dan als property worden opgeslagen wat de key van de lijn is. Bijvoorbeeld als property-name wordt dan "coronalline" gezet, en als property-value het objectID van deze lijn.

Aangezien lijnobjecten geselecteerd en verplaatst kunnen worden, zal dit invloed hebben op de andere afbeeldingen. Zodra er een Direct Graphics object wordt verplaatst, wordt de methode moveObj van dw_oGraph aangesproken. Hierin kan dan worden gezet dat er een terugkoppeling nodig is naar de dwSetImage methode van de widgetHandler. Dit omdat de widgetHandler alle andere drawWidgets op de gebruikelijke manier kan aanspreken. Hier dient dan wel rekening gehouden te worden dat een andere lijn (wat het zelfde aanzicht representeert) ook wordt aangepast.

Verder kan in een variabele in fw_drawWidget worden gemaakt voor het minimal threshold level. Door deze variabele aan te passen zal de minimale drempelwaarde in de afbeelding veranderd kunnen worden. Aangezien de key van de huidige afbeelding in het drawWidget wordt bijgehouden, is het mogelijk de afbeelding opnieuw te berekenen zodra deze drempelwaarde wordt aangepast. Natuurlijk is het verstandig om deze variabele in de methode setImage ook mee te nemen. Dezelfde theorie geldt ook voor een multiply-factor binnen een widget. Hierdoor kan een afbeelding "warmer" of "kouder" worden gemaakt.

De uitbreidingen op de methode setImage kunnen dan zo geprogrammeerd worden dat deze variabelen optioneel zijn. Hierdoor wordt de compatibiliteit met eerdere versies van het framework bewaard. Dit bied als voordeel dat oudere applicaties, ontwikkeld met een oudere versie van het iMed framework, ook op de nieuwe versie van het iMed framework kunnen werken.

Pixelarrays en snelheden

In de eerste bijlage van dit document (bijlage A) is een beschrijving van een test met arrays toegevoegd. Hieruit is te concluderen dat het cachen van arrays voordelen heeft m.b.t. het snel weergeven van andere aanzichten. De latency (vertraging) als gevolg van het berekenen van een nieuw aanzicht is hierbij nagenoeg 0. Het grote probleem wat hier bij komt, is dat de cache gemaakt moet worden, en dit tijd kost.

Hierop wordt in bijlage B verder gegaan. Bijlage B bevat de uitkomst van een test met betrekking tot het inlezen van bestanden en het maken van een 4-dimensionale array hiervan. Dit aangezien de bestanden al een array met 3 dimensies bevatten. On the fly (beschreven in bijlage A) is voor de eindgebruiker goed te doen tot en met 64 bestanden. Hierbij zal de eindgebruiker geen vertraging bemerken. Het cachen van één aanzicht duurt 23 seconden bij gebruik van 64 bestanden (127x127x47x64). Aangezien er twee aanzichten gecached worden, zal er 46 seconden nodig zijn om beide aanzichten klaar te maken. Dit is dus veel te lang. Het is daarom toch verstandiger om gebruik te maken van on the fly.

Hiervoor dient de dicom-klasse herschreven te worden. Zowieso aangezien er op dit moment rechtstreeks de pixeldata wordt aangesproken. In de huidige situatie zou er overal (waar nodig) de code moeten staan om de juiste informatie uit de pixelarray te halen. Aangezien dit geen goede oplossing is voor de onderhoudbaarheid van het framework, dient dit ondergebracht te worden in de dicom klasse. De dicom klasse zal dan een 2-dimensionaal array terug geven betreffende het gevraagde plaatje op een opgegeven plane en tijdsframe (3^e en 4^e dimensie).
Op dit moment wordt er binnen het framework met 3-dimensionale arrays gewerkt. Aangezien het mogelijk is dat er overgestapt wordt naar 4-dimensionale arrays, is het verstandig om dit nu te herschrijven zodat de dicom-klasse wordt aangeroepen. Verder is het dan verstandig dat de lokaal uitgevoerde array-mutaties worden ondergebracht in deze dicom-klasse. Dit bied als voordeel dat aanpassingen in het dicom- of bestandssysteem op één plaats worden gedaan.

4.2.1 Sequentie diagram

Hieronder is een sequentie diagram weergegeven voor het veranderen van de afbeelding. Dit sequentiediagram wordt uitgevoerd zodra er een ander plane wordt geselecteerd.

In feite blijven de eerste stappen hetzelfde als in de huidige situatie. Alleen zal in de widgetHandler de methode dwSetImage uitgebreid worden. Bij deze methode zal op het laatst een signaal worden verstuurd naar alle drawWidgets die een ander aanzicht hebben dan degene wat het event opgooit. Dit signaal is dan een aanroep naar een functie wat aangeeft welk type view is aangepast, en dat het aan te roepen drawWidget zijn lijn, gerelateerd aan zijn type weergave, moet aanpassen. Dit zal in werkelijkheid door middel van een loop gebeuren, waarbij over alle aanwezige drawWidgets wordt bepaald of deze een dergelijk signaal nodig hebben.

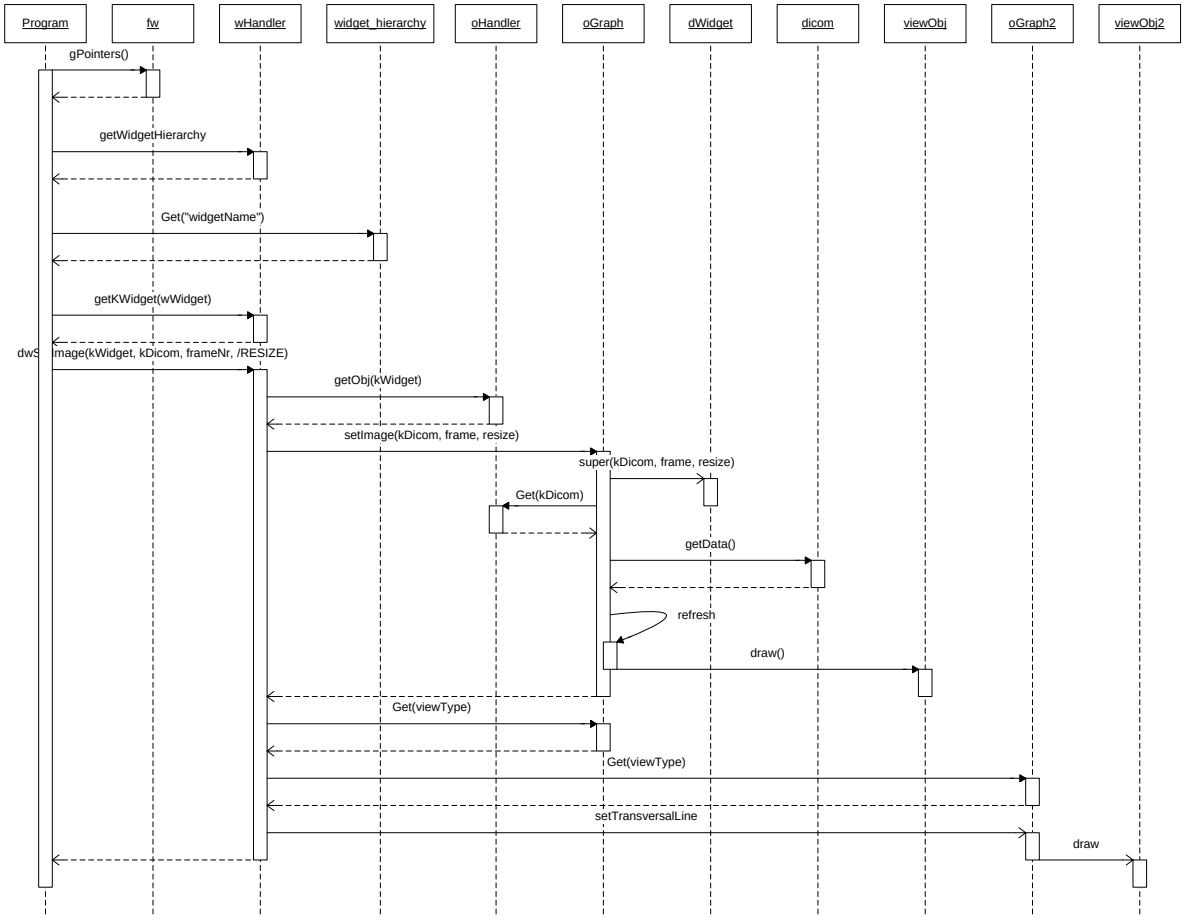


Diagram 4: Laden van een afbeelding in de nieuwe situatie

Functioneel ontwerp iteratie 2

Inleiding

Dit document heeft tot doel de wensen en eisen van de opdrachtgever helder te formuleren, en staat aan de basis voor het technisch ontwerp van het eindproduct.

De belangrijkste reden voor het gebruik van dit document is om met de opdrachtgever duidelijk af te spreken welke functionaliteit er wel en niet wordt gerealiseerd. Als een volgende fase is afgelopen (bijvoorbeeld het technisch ontwerp, of de realisatie), kan er aan de hand van dit document bekeken worden of alle gevraagde functionaliteit ook verwerkt is.

Aangezien het project initieel als doel heeft om het iMed framework uit te breiden, zal het document niet veel eisen bevatten die voor de eindgebruiker van belang zijn. De meeste eisen komen van de programmeurs die gebruik maken van het framework. Deze programmeurs gebruiken het framework om applicaties te ontwikkelen die door het Siemens e.Soft systeem worden aangeroepen. Hiervan stamt ook het doel van dit project, namelijk het uitbreiden van de aanwezige functionaliteit. Met deze uitbreiding zal het voor de programmeurs makkelijker worden om applicaties te schrijven die in drie dimensies berekeningen moet uitvoeren.

Dit functioneel ontwerp dient als uitgangspunt voor de tweede iteratie van dit project. De functionele eisen en Use Cases zijn voor deze iteratie aangepast ten opzichte van de eerste iteratie. Voor de functionele eisen en Use Cases uit iteratie 1 wordt verwezen naar het Functioneel Ontwerp van iteratie 1 (Iteratie 1 – Functioneel Ontwerp / Functioneel Ontwerp v02.docx).

Aan het eind van deze iteratie (iteratie 2), zal dit document gebruikt worden om te bepalen of de gevraagde functionaliteit ook allemaal geïmplementeerd is. Dit kan aan de hand van de use cases. Men zou aan het einde van deze iteratie een kleine testapplicatie kunnen schrijven, waarna de functionaliteit vergeleken kan worden met de use cases. Men kan dan de "normal flow" als acties uitvoeren. Als controle kan dan worden gekeken of het "resultaat" of "alternatieve flow" uit de use case overeenkomt met de actie die het programma uitvoert.

Zo niet, zal dit in de uitloop van iteratie 2 (zie bijlage B van het PiD) worden aangepast. Mocht dit niet haalbaar zijn in de uitloop van iteratie 2, zal dit (na overleg met de opdrachtgever over de prioriteit) verschijnen in het Functioneel Ontwerp van Iteratie 3.

Vanuit de planning is iteratie 3 gericht op het implementeren van een 3D weergave. Als de ontbrekende functionaliteit van belang is voor iteratie 3, zal dit een must-have worden in het Functioneel Ontwerp van iteratie 3.

Inhoudsopgave

1 Analyse

Op dit moment gebruikt men een framework om applicaties te schrijven die door het e.Soft systeem worden aangeroepen. Dit framework heeft de naam iMed, en bestaat uit een library-set van meerdere klassen. Deze library set zijn losse code-bestanden, en kunnen in een project toegevoegd worden. In iteratie 1 is het mogelijk gemaakt om een 2D GUI-type[6] te creëren.

De bedoeling van de framework-uitbreiding (iteratie 2) is om het huidige 2D model binnen het framework uit te breiden naar een 3D model. Hieronder staan de eigenschappen waaraan dit uitgebreide model na iteratie 3 aan moet voldoen:

- De gerealiseerde uitbreidingen moeten een meerwaarde zijn op de bestaande code in het framework. Dat wil zeggen dat de extra functies die zijn gemaakt, ook gebruiksvriendelijk zijn voor de programmeur, en dat deze ook nieuwe mogelijkheden biedt voor de eindgebruiker.
- Het framework moet gemakkelijk zijn om mee te werken c.q. te programmeren
- Het framework moet geen geheugenlekken veroorzaken waardoor een pc kan vastlopen
- De gerealiseerde uitbreidingen zal het mogelijk maken om eenvoudig VOI's[7] te tekenen.
- Het framework dient op deze VOI's standaardberekeningen uit te kunnen voeren. Deze standaardberekeningen zijn bijvoorbeeld het berekenen van het aantal counts binnen een oppervlak/volume, het uitrekenen van het aantal Bq/ml, of het bepalen van de pixelinhoud. Complexe berekeningen dienen in de applicaties verwerkt te worden. Dit wordt gedaan om het framework open te houden voor nieuwe ontwikkelingen.
- De gerealiseerde uitbreiding zal met minimaal een 2D GUI-type opgeleverd moeten worden (namelijk 3 verschillende 2D vensters), en waar mogelijk met een 3D GUI-type[8].

[6] De 2D GUI zal bestaan uit drie verschillende aanzichten op het object, namelijk sagittaal (van de linker zijkant), coronaal (van de voorkant) en transversaal (van de bovenkant).

[7] VOI is de afkorting voor een Volume of Interest. Een VOI bestaat uit meerdere ROI's (Region of Interest) die achter elkaar in meerdere planes bevinden. Hierdoor ontstaat een ROI. Aangezien dit uit drie dimensies bestaat, wordt er dan gesproken over een Volume of Interest.

[8] De 3D GUI-type zal bestaan uit één venster, waarin het object in verschillende richtingen gedraaid kan worden. Hierdoor is er maar één weergave nodig.

2 Programma van Eisen

In dit hoofdstuk worden de wensen en eisen van het programma beschreven en gecategoriseerd. Allereerst zal er een splitsing worden gemaakt in functionele eisen en niet-functionele eisen. In de functionele eisen staat de functionaliteit van het systeem beschreven (functies die de toekomstige gebruiker met het systeem kan uitvoeren). De niet-functionele eisen zijn eisen die niet als functionaliteit aantoonbaar zijn, maar wel invloed hebben op het product.

Het categoriseren van de functionele eisen gebeurt aan de hand van de MoSCoW-methode. Hierbij worden de eisen ingedeeld in Must haves (eisen die in het product moeten), Should haves (eisen die eigenlijk wel in het product zouden moeten, maar een mindere prioriteit hebben), Could haves (eisen die niet in het product hoeven, maar die gemaakt worden als er tijd over is) en Won't haves (eisen die zeker niet in het product komen). Deze methode geeft bij een grote tijdsdruk minder conflicten als er beslist moet worden welke functionaliteit al dan niet opgeleverd kan worden.

2.1 Functionele eisen

In deze paragraaf worden de functionele eisen van iteratie 2 beschreven. Zoals in het PiD beschreven zal iteratie 2 zich voornamelijk richten op het tekenen van regions en VOI's over 3 dimensies, zogenaamde Volume of Interests. De eisen zijn van toepassing op het functioneren van het systeem, en niet op factoren van het systeem (zoals snelheid en beveiliging).

Must Haves

Het achterliggend model dient uitgebreid te worden zodat er kan worden omgegaan met Volumes of Interest. Aangezien dit met het Technisch ontwerp te maken heeft, zal hiervan geen use case beschikbaar zijn.

De aanzichten (gemaakt in iteratie 1) zullen een uitbreiding krijgen betreffende het tekenen van 3D-objecten. Deze objecten kunnen een kubus, cilinder of bol zijn.

Deze objecten moeten aangepast kunnen worden betreffende posities en afmetingen binnen de afbeelding. Deze posities/afmetingen moeten door middel van de drie aanzichten aangepast kunnen worden. Dit houdt ook in dat de andere twee aanzichten automatisch bijgewerkt worden zodra een aanzicht wordt aangepast.

Should Haves

Van de nieuwe Volume of Interests zijn er twee 3D weergaves beschikbaar. Namelijk een die met alleen lijnen zichtbaar is, en een andere waarbij er een doorzichtig object wordt gemaakt waardoor de achterliggende pixeldata zichtbaar blijft.

Could haves

Het genereren van een 3D afbeelding waarin het DICOM bestand wordt gebruikt als basis van het 3D beeld.

Het weergeven van de Volume of Interests in een 3D weergave op twee manieren. Namelijk met lijnen (de randen van de VOI), of als doorzichtige vlakken.

Won't Haves

Het gebruik van meerdere tijdsframes om door de tijd verschillende afbeeldingen te bekijken.

2.2 Niet-functionele eisen

Naast alle functionele eisen (wat moet het programma kunnen) zijn er ook eisen die niet direct aan het programma zelf worden gesteld, maar op de werking van het programma van toepassing zijn. Dit zijn de niet-functionele eisen. In dit hoofdstuk zijn deze eisen opgesomd en beschreven.

Code

De applicatie dient te worden geschreven in de programmeertaal IDL en er dient (waar mogelijk) gebruik gemaakt te worden van de aanwezige functies in het framework.
De keuze om het framework initieel in IDL te schrijven is omdat de Siemens e.Soft systemen externe IDL programma's op de machine kunnen starten, en dat deze binnen e.Soft zichtbaar zijn. Hierdoor ontstaat er een vlekkeloze integratie en lijkt het voor de eindgebruiker alsof de applicatie bij het e.Soft systeem zelf hoort. Verder is het een krachtige taal om berekeningen op beelden te kunnen uitvoeren. Voor dit doel wordt de taal ook bij de NASA en in de meteorologie gebruikt.

Alle publieke functies/methoden dienen beschreven te worden met PRE en POST situatie. Hiermee wordt beschreven wat de input en output van een functie zal zijn. Verder zal er in het kort omschreven worden hoe de functie dit precies uitvoert.

Er wordt gewerkt met een SVN server. Dit zorgt ervoor dat iedereen (de opdrachtnemer en de bedrijfsbegeleider/consultant) altijd de actuele versie beschikbaar heeft. Als iemand aan een codefragment heeft gewerkt, zal hij deze op de server plaatsen. Dit moet met goed commentaar gebeuren. Hierdoor kan men nagaan waarom een bepaalde wijziging is gemaakt, en wanneer dat is gebeurt (en door wie).

Hardware

De applicatie zal uiteindelijk op een Siemens e.Soft systeem gaan werken. Deze systemen worden door Siemens zelf geleverd, en waar nodig wordt de hardware uitgebreid. Hierdoor wordt aan het uitgebreide framework de hardware eis gesteld dat het op de bestaande e.Soft systemen moet werken. Wel speelt de beeldresolutie een punt bij de e.Soft systemen. Dit omdat er een screenshot wordt gemaakt van de applicatie, en deze gebaseerd is op een bepaalde ruimte binnen het beeldscherm.

Performance

Natuurlijk is het streven om berekeningen zo snel mogelijk uit te voeren, anders wordt er door de gebruikers snel een negatieve stempel op geplakt en is het in (klinische) praktijk minder bruikbaar. Als een systeem namelijk lang werkt heeft, zal er gezocht worden naar alternatieven die sneller werken, of zullen taken verplaatst worden naar het einde van de dag. Dit kan weer als gevolg hebben dat het stellen van een diagnose langer duurt, dit is dan ook minder prettig voor de patiënt. Van de andere kant is het wel zo dat een systeem kwalitatief goed en betrouwbaar moet zijn. De keuze is dan om een middenweg te vinden die kwalitatief goed is, maar ook redelijk snel is.

Veiligheid

De veiligheid berust hier vooral op privacy. Aangezien het framework wordt gebruikt om uitbreidingen te schrijven op het e.Soft systeem, zal dit punt wegvallen. De privacy van patiëntdata zal worden gewaarborgd door het e.Soft systeem zelf. Wel zal het natuurlijk zo zijn dat de uiteindelijke applicatie, wat met het iMed framework geschreven wordt, niet met patiëntendata moet gaan "rommelen".

Taal

De taal voor de grafische kant van de uitbreidingen zal Engels zijn. Ook zullen er medische termen in voorkomen om op één lijn te blijven met de medisch onderlegde eindgebruiker van het e.Soft systeem.

Gebruiksgemak

De eindgebruiker (gericht op de GUI onderdelen) zal een basiskennis hebben van medische apparatuur. Hierdoor zullen schermvormgevingen gebaseerd worden op hun kennis of ervaring. Dit werkt bevorderend voor het gebruiksgemak, omdat de gebruikte systemen een zelfde soort interface hebben en dit het leerproces voor een applicatie verkort.

Betrouwbaarheid

De betrouwbaarheid van de uitbreiding zal hoog moeten zijn. Er worden namelijk allerlei berekeningen en analyses uitgevoerd op basis van de informatie die het framework geeft. Dit betekent dat de performance misschien iets lager wordt, maar dat de kwaliteit/betrouwbaarheid wel hoger is. Dit is belangrijk omdat applicaties die met het uitgebreide framework worden gebouwd, gebruikt kunnen worden om medische beslissingen te nemen. Als er dan structureel een fout zit in het framework kan dit consequenties opleveren voor de diagnose of behandeling van een patiënt.

Handleidingen

In principe zullen er geen handleidingen voor de eindgebruiker geschreven worden. Dit aangezien het een uitbreiding op het framework is, en geen applicatie op zichzelf. Daarom zal het Technisch Ontwerp als handleiding dienen.

Documentatie

Zoals hierboven geschreven dient het Technisch Ontwerp als documentatie, samen met dit document en het commentaar in de code. Dit commentaar zal door middel van pre en post omschrijvingen bij functies gebeuren. Bij uitgebreide functies zullen er binnen de code regels commentaar komen te staan.

3 Use Cases

In dit hoofdstuk staan alle use cases uitgewerkt, in de vorm van de MoSCoW-methode. Deze komen overeen met de eisen in paragraaf 2.1. Hierbij zijn de use cases voor de Won't Haves weg gelaten. Aangezien deze componenten allemaal betrekking hebben op applicaties die met het iMed framework gemaakt kunnen worden, wordt er gesproken over de gebruiker van e.Soft als Actor. Als men de programmeur als actor zou gebruiken, zou dit het doel van het project mislopen, aangezien er functionaliteit gebouwd wordt, waardoor de programmeur minder werk heeft met stukken code die toch altijd hetzelfde zijn. Het is daarbij belangrijk om te documenteren wat de functionaliteit hiervan is voor de eindgebruiker. Hoe de technische implementatie hiervan gaat, is af te leiden uit het Technisch Ontwerp.

3.1 Must Haves

Naam:	Tekenen van VOI
Actor:	De gebruiker van e.Soft
Pre:	De beelden zijn als 3x 2D gegenereerd
Normal flow:	- De actor geeft aan welke weergave hij wil gebruiken - De actor geeft aan dat hij een ROI wil gaan tekenen - De actor tekent een gesloten polygon binnen de weergave - De actor selecteert een andere weergave - De actor trekt in deze weergave de ROI door zodat de dimensie die niet beschikbaar was, wordt aangepast.
Resultaat:	- De actor krijgt als resultaat dat er een polygon is getekend, en dat deze in de andere aanzichten is bijgewerkt.
Alternatieve flow:	

Naam:	Berekenen activiteit binnen VOI
Actor:	De gebruiker van e.Soft
Pre:	Er is een VOI getekend
Normal flow:	- De actor geeft aan dat hij de activiteit binnen de ROI wil weten
Resultaat:	- De actor krijgt als resultaat de waarde van de activiteit, gedefinieerd in counts en eventueel in Bq/mL
Alternatieve flow:	

Naam:	Schalen van een VOI
Actor:	De gebruiker van e.Soft
Pre:	Er is een VOI getekend
Normal flow:	- De actor geeft aan welke VOI hij wil aanpassen - De actor geeft aan hoe groot de twee dimensies moeten worden. Dit doet hij/zij door het object met de muis aan te passen.
Resultaat:	- De actor krijgt als resultaat dat de VOI geschaald is naar de waarde die hij heeft aangegeven, en dat eventuele statistiek en andere aanzichten zijn bijgewerkt.
Alternatieve flow:	

Naam:	Roteren van een VOI
Actor:	De gebruiker van e.Soft
Pre:	Er is een VOI getekend
Normal flow:	- De actor geeft aan welke VOI hij wil aanpassen - De actor geeft hoeveel graden hij de VOI wil roteren in de twee richtingen die mogelijk zijn in het aanzicht
Resultaat:	- De actor krijgt als resultaat dat de VOI geroteerd is naar de waarde die hij heeft aangegeven, en dat eventuele statistiek en andere aanzichten zijn bijgewerkt.
Alternatieve flow:	

Naam:	Verplaatsen van een VOI
Actor:	De gebruiker van e.Soft
Pre:	Er is een VOI getekend
Normal flow:	- De actor geeft aan welke VOI hij wil verplaatsen - De actor geeft aan waar hij de VOI naartoe wil verplaatsen in de richtingen die mogelijk zijn in het betreffende aanzicht.
Resultaat:	- De actor krijgt als resultaat dat de VOI verplaatst is naar de waarde die hij heeft aangegeven, en dat eventuele statistiek en andere aanzichten zijn bijgewerkt.
Alternatieve flow:	

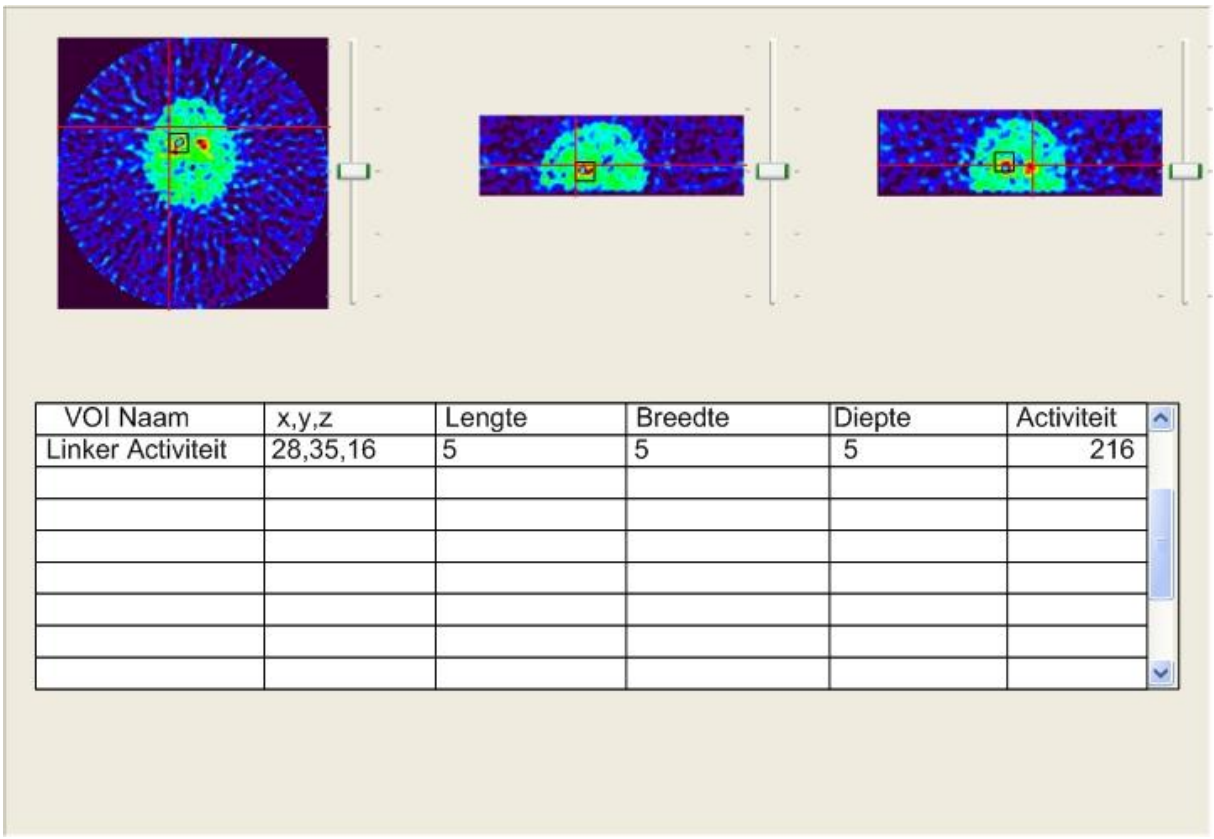
3.2 Should Haves

Naam:	Weergavetype van VOI
Actor:	De gebruiker van e.Soft
Pre:	Er is een ROI getekend
Normal flow:	- De actor geeft aan welke VOI hij wil aanpassen - De actor geeft aan of hij alleen de lijnen wil zien, of dat hij de VOI als een gevuld vlak wil zien. Dit vlak zal een bepaald transparantie-niveau bevatten.
Resultaat:	- De actor krijgt als resultaat dat de VOI in het opgegeven weergavetype wordt weergegeven
Alternatieve flow:	

3.3 Could Haves

Naam:	Genereren van een 3D weergave
Actor:	De gebruiker van e.Soft
Pre:	Er is een DICOM bestand beschikbaar binnen het programma/framework
Normal flow:	- De actor geeft aan dat deze een 3D beeld wil zien
Resultaat:	- De actor krijgt als resultaat dat er een 3D beeld wordt gegenereerd op basis van de DICOM informatie - In het gegenereerd beeld wordt eventueel een VOI weergegeven.
Alternatieve flow:	

4 Schermvoorbeeld



VOI Naam	x,y,z	Lengte	Breedte	Diepte	Activiteit
Linker Activiteit	28,35,16	5	5	5	216

In de bovenstaande afbeelding is te zien wat de eerste 2D versie van het framework moet kunnen, als alle must en could haves zijn geïmplementeerd. Het moet namelijk mogelijk zijn om dan een VOI te tekenen. Hiervoor zijn minimaal 2 aanzichten nodig. In dit voorbeeld is echter voor drie aanzichten gekozen, aangezien het dan voor de eindgebruiker waarschijnlijk makkelijker is om zichzelf te oriënteren.

Verder worden er lijnen in de afbeelding aangegeven. Deze dienen ter verduidelijking betreffende de posities van de weergaves binnen 3 dimensies. In de transversale weergave (links) kan men aan de hand van de lijnen de locatie van de sagittale (midden) en coronale (rechts) weergaves duidelijk maken. In de sagittale weergave worden de transversale en coronale weergave aangeduid, en in de coronale weergave worden de transversale en sagittale weergave aangeduid.

Deze weergaves kunnen verplaatst worden door met de sliders aan de zijkant te bewegen, of met het scrollwiel van de muis. Hierbij zullen de betreffende lijnen ook van plaats verschuiven. Verder is het ook mogelijk om een optie aan te zetten zodat de muis gevolgd wordt binnen het venster, waardoor de andere twee vensters automatisch veranderen.

Onder in de tabel zijn de VOI's beschreven. Deze VOI's worden aangemaakt door een vierkant te tekenen in het aanzicht. Hierdoor verschijnt in het/de ander(e) aanzicht(en) een lijntje dat uitgetrokken kan worden. Als gevolg hiervan ontstaat een kubus. Dit is dan een Volume of Interest waarmee in drie dimensies de activiteit kan worden bepaald.

Technisch ontwerp iteratie 2

Inleiding

Dit document heeft tot doel een heldere beschrijving te geven van de aanpassingen voor het iMed framework. Dit document zal ook als leidraad dienen bij de daadwerkelijke bouw/aanpassing van het framework. In dit document komt naar voren waarom sommige keuzes betreffende architectuur/programmatuur gemaakt zijn.

De belangrijkste reden voor het gebruik van dit document is om, met de bedrijfsbegeleider, duidelijk af te spreken welke aanpassingen/uitbreidingen er gemaakt worden. Deze aanpassingen/uitbreidingen zullen resulteren tot een realisatie van de wensen en eisen uit het functioneel ontwerp. Naderhand zal ook met het Functioneel Ontwerp worden vergeleken of de aanpassingen daadwerkelijk zorgen voor de nieuwe functionaliteit.

Inhoudsopgave

1 Het platform

Het platform waarop het iMed framework wordt gebruikt zijn de Siemens e.Soft workstations. In principe zijn dit normale computers. Hierbij kan gedacht worden aan een computer wat heden ten dage in de computerwinkel te koop is. Mocht het zo zijn dat de specificaties niet voldoende zijn, zal dit bij Siemens worden aangekaart, en zullen zij ervoor zorgen dat de pc (waar nodig) een upgrade krijgt.

Deze computers werken op dit moment met Windows XP als besturingssysteem. Hierop wordt een applicatie gedraaid die ontwikkeld is door Siemens. Deze applicatie, genaamd e.Soft, verzorgt het aansturen van een scanner, en het verwerken en weergeven van de DICOM-images.

Met deze DICOM-images is het mogelijk om, in de Siemens software, analyses uit te voeren. Veel van deze analyses zijn al aanwezig in de applicatie, maar het is ook mogelijk om zelf analyses toe te voegen. Deze kunnen in IDL worden geschreven. Hiervoor levert Siemens de Virtual Machine van IDL mee. Door het gebruik van een Virtual Machine is het mogelijk om dezelfde bytecode uit te voeren op verschillende besturingssystemen. De Virtual Machine vertaalt namelijk de bytecode naar machinecode.

De applicatie (geschreven in IDL) wordt dan aangeroepen door het e.Soft programma. Deze wordt dan schermvullend boven op het e.Soft programma gedraaid. Zodra deze wordt afgesloten worden er screenshots genomen van de resultaten, en worden deze (via een nieuw DICOM-bestand) terug gegeven aan het e.Soft programma.

2 Gekoppelde bestanden en applicaties

Zoals in de vorige paragraaf is uitgelegd, wordt de IDL software door e.Soft gestart. Het e.Soft programma zal dan ook de benodigde files meegeven om de studie uit te voeren.

Zodra een studie is afgerond, is het mogelijk om met het iMed framework screenshots te maken van resultaten. Deze kunnen dan als afbeeldingen bij de studie worden opgeslagen.

Aangezien deze functionaliteit al in het framework aanwezig is en niet aangepast hoeft te worden (zie Functioneel Ontwerp), zal dit document hier niet verder op ingaan.

3 Software

Zoals in het Functioneel Ontwerp is uitgelegd, zal de software in IDL worden ontwikkeld. Als ontwikkelomgeving zal hiervoor een standaard computer van het UMC St. Radboud gebruikt worden. Hierop zal IDL 6.3 van ITT VIS geïnstalleerd worden. Deze computer zal dus als ontwikkelomgeving gebruikt worden.

Het huidige iMed framework zal als resource gebruikt worden. Dit omdat de huidige functionaliteit behouden moet blijven, en de functies hiervan wellicht in de uitbreiding gebruikt kunnen worden. Ook zal het nieuwe framework in zijn geheel moeten samenwerken. Het mag niet mogelijk zijn dat door de nieuwe functionaliteit sommige functies niet meer kunnen samenwerken.

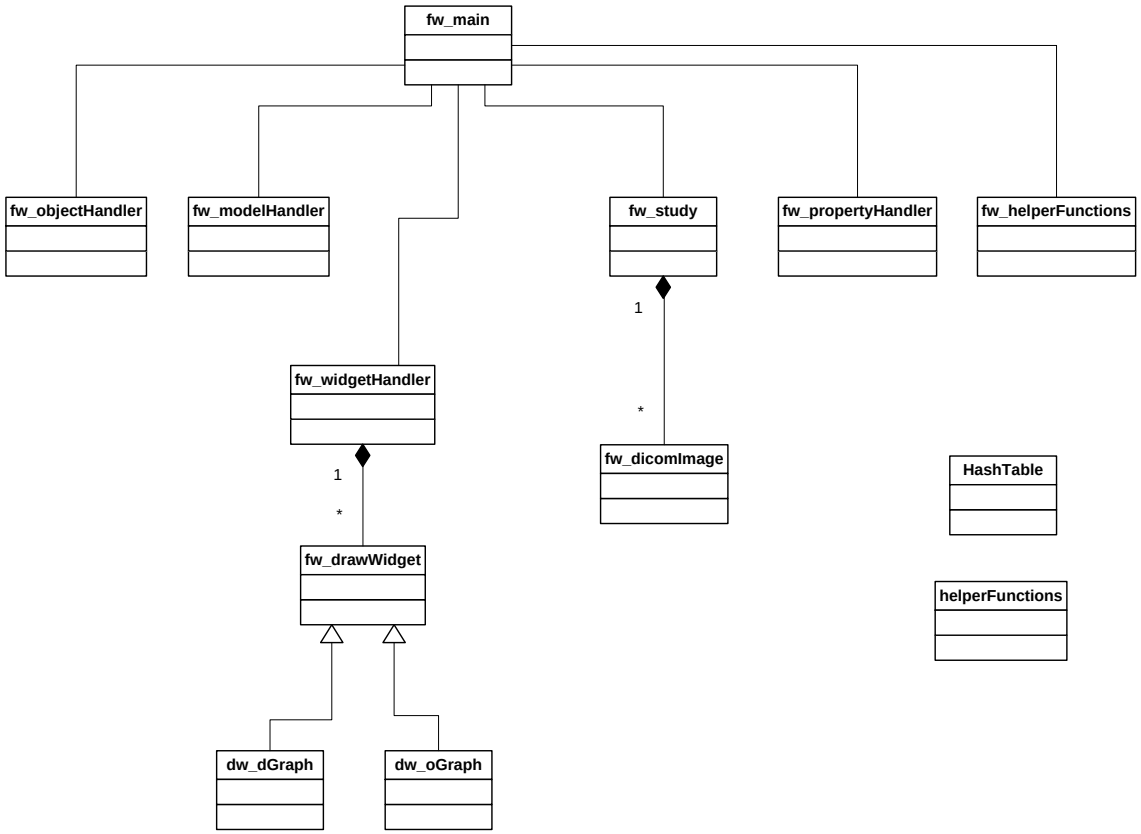
4 Programmastructuur

Dit hoofdstuk is ingedeeld in twee paragrafen. Allereerst zal de huidige structuur worden besproken. Hiertoe zijn een klassendiagram en drie sequentiediagrammen gemaakt. Het klassendiagram geeft de globale structuur van de applicatie aan, en aat zien welke objecten er mogelijk aangemaakt worden, en welke verhouding deze tot elkaar hebben. De sequentiediagrammen zijn gemaakt om inzicht te geven in de flow van de applicatie, en hoe de objecten binnen de applicatie gebruikt worden.

4.1 Huidige Situatie

4.1.1 Klassendiagram

Hieronder in het figuur staat het huidige domeinmodel. Dit model is beperkt tot alleen de naam van de klasse, en de relaties van de objecten. Dit is gedaan omdat het klassendiagram niet binnen de afmetingen van dit document zou passen. Onder het figuur staat uitgelegd wat de functie van de verschillende klassen is.



Figuur 1: Domeinmodel

fw_main: dit is het basisobject van het framework. Hierin worden de links (pointers) naar de objecten fw_objectHandler, fw_modelHandler, fw_widgetHandler, fw_study, fw_propertyhandler en fw_helperfunctions bijgehouden. Dit object moet binnen de applicatie beschikbaar zijn, om het aanroepen van het framework mogelijk te maken.

fw_modelHandler: In IDL is het mogelijk om gebruik te maken van Object Graphics. Hierbij worden objecten in een virtuele 3D wereld gemaakt, en worden deze via een view gepresenteerd. Fw_modelHandler zorgt ervoor dat objecten aan een model gekoppeld worden, en dat deze worden onderhouden. Deze modellen kunnen vervolgens aan de view van dw_oGraph gekoppeld worden. Dit koppelen gebeurt dan in de widgetHandler.

fw_widgetHandler: Deze klasse beheert de widgets en views die gekoppeld worden aan de models uit fw_modelHandler. Als er via fw_widgetHandler een nieuw object wordt aangemaakt (bijvoorbeeld een ROI, een lijn, of dergelijke), wordt deze doorgestuurd naar fw_modelHandler, om dit aan het juiste model te koppelen. Ook regelt deze klasse de communicatie als een ROI op meerdere views zichtbaar moet zijn, en als een nieuwe ROI gemaakt wordt. Deze klasse geeft dan aan de klasse van het type fw_drawWidget door dat er een refresh moet worden uitgevoerd.

fw_drawWidget: Deze klasse is de parent van de klassen dw_dGraph en dw_oGraph. De functionaliteit van deze klasse beperkt zich tot een aantal functiedeclaraties wel dan niet met een body. Verder worden hier variabele opgeslagen (via de propertyHandler) die van toepassing zijn op beide type drawWidgets (dw_dGraph en dw_oGraph).

dw_dGraph: Deze klasse erft van fw_drawWidget, en wordt gebruikt voor objecten van het type WIDGET_DRAW, die gebruik maken van direct graphics. Deze modus biedt minder grafische mogelijkheden, maar is wel sneller bij het tekenen van afbeeldingen. Als voorbeeld van de beperkte mogelijkheden is er geen mogelijkheid om eigen onderdelen (bijvoorbeeld ROI's) te tekenen of te bewerken met direct graphics. Hierdoor heeft deze klasse binnen het framework minder functies en mogelijkheden.

dw_oGraph: Deze klasse erft van fw_drawWidget, en wordt gebruikt voor objecten van het type WIDGET_DRAW, die gebruik maken van Object Graphics. Hierbij wordt een model (vanuit fw_modelHandler) gekoppeld aan deze view. Ook worden hier de events van het widget afgehandeld (bijvoorbeeld de muisevents bij het tekenen van een Region of Interest). Het gebruik van object graphics (gekoppeld met een eigen model) brengt wel als nadeel met zich mee dat het meer geheugen en snelheid kost, dan de direct graphics. Daarom dient men een afweging te maken voor het gebruik van direct of object graphics. Het maken van deze afweging gebeurt op applicatie-niveau.

fw_study: Deze klasse zorgt ervoor dat een (of meerdere) DICOM bestand(en) in de applicatie wordt ingeladen. Aangezien een DICOM bestand meer informatie bevat dan alleen de afbeeldingen, worden deze in de study klasse afgehandeld.

fw_dicomImage: In deze klasse worden de afbeeldingen uit het DICOM bestand bewaard. Verder is het hier mogelijk om een aantal basisfuncties op de afbeeldingen uit te voeren. Deze afbeeldingen zijn in principe gewoon 2 of 3 dimensionale array's met getallen. Ook kan er uit de headers informatie worden gelezen betreffende de naam van de patiënt, scanner instellingen en andere specifieke informatie betreffende het bestand.

fw_propertyHandler: De propertyHandler maakt het mogelijk om d.m.v. keywords (eventueel overeenkomstig met de velden binnen een object) allerlei informatie op te slaan. Als er een veld met dezelfde naam beschikbaar is binnen het object, zal de waarde van de eigenschap (property) in het betreffende veld worden opgeslagen. Is het opgegeven veld niet beschikbaar binnen het object (bijvoorbeeld bij een nieuwe, of tijdelijke property), zal dit als referentie in een globale HashTable worden opgeslagen. Als alle properties gezet en gelezen worden via deze propertyHandler, is het dus mogelijk om in een applicatie (ontwikkeld met het framework) extra informatie op te slaan. Dit bied als voordeel dat het framework flexibel blijft, en dus niet voor iedere applicatie structurele aanpassingen behoeft.

Verder bied deze klasse de mogelijkheid om deze propertys te importeren/exporteren van/naar een XML bestand. Hierdoor wordt het mogelijk om dynamische applicatie-informatie op te slaan zodat de informatie hergebruikt kan worden. Ook nadat het programma opnieuw wordt gestart.

fw_helperFunctions: Deze klasse bestaat uit allerlei hulpfuncties. De hulpfuncties maken gebruik van de opbouw van het framework. Hier zit bijvoorbeeld een functie in om een Region of Interest te splitsen.

HashTable: IDL maakt geen gebruik van collections. Collections zijn objecten die datasets op verschillende manieren kunnen beheren. Voor verschillende type datasets zijn verschillende type collection objecten beschikbaar. Een voorbeeld hiervan is een HashTable (of HashMap). Hierbij wordt een key gekoppeld aan een bepaalde waarde. De key wordt dan gekoppeld aan een adres in het geheugen, waardoor zeer snel de waarde bij de gegeven key opgehaald kan worden.

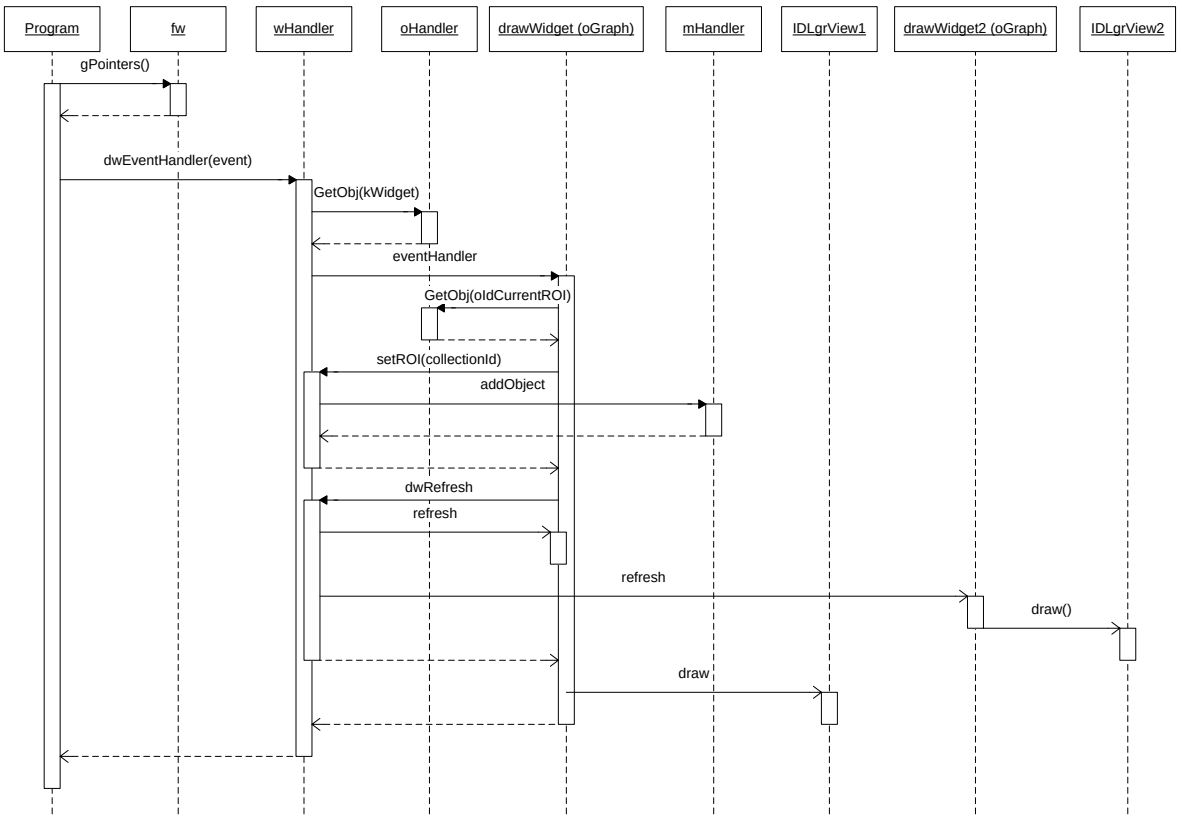
Fw_objectHandler: Deze klasse is de grote container voor het framework. Hierin wordt in een HashTable een sleutel, en de pointer van elk object opgeslagen. Dit heeft als voordeel dat objecten

altijd overal in de code terug te vinden zijn, aangezien een hashTable een lijst met sleutels kan teruggeven.

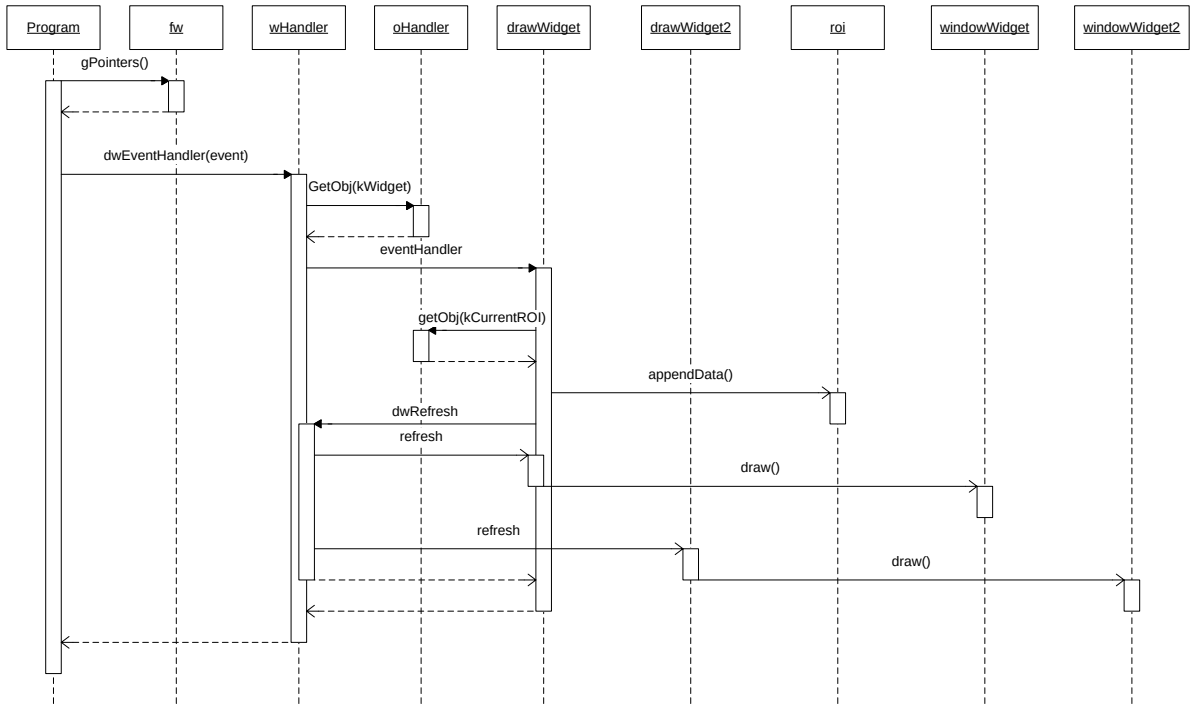
4.1.2 Sequentie Diagrammen

In het eerste diagram is beschreven hoe een event tijdens het tekenen van een ROI wordt afgehandeld. In deze sequentie wordt een nieuwe ROI gemaakt, en aan het bestaande model gekoppeld.

In het tweede diagram is beschreven hoe een event tijdens het bijwerken van een ROI wordt afgehandeld. Hiermee wordt bedoeld wat er met het event gebeurt als de gebruiker de muisknop ingedrukt houdt, en met de muis beweegt. Hierbij wordt een nieuw coördinaat aan de ROI meegegeven.



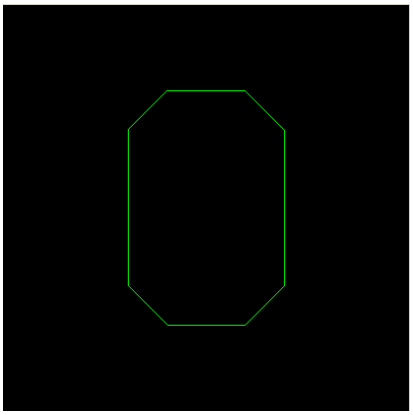
Figuur 2: drawWidget event, maken van een nieuwe ROI



Figuur 3: drawWidget event, bewerken van een bestaande ROI

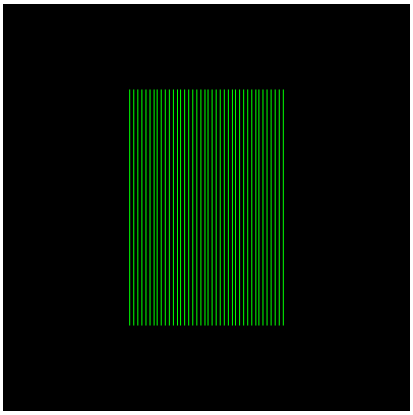
4.2 Nieuwe Situatie

Voor de tweede iteratie zullen een aantal uitbreidingen op bestaande functies en objecten geschreven worden. De uitbreidingen zullen er voor zorgen dat het mogelijk wordt drie-dimensionaal regions (zgn. VOI's) te gebruiken binnen het framework. Uit vooronderzoek van deze iteratie is gebleken dat het mogelijk is om hiervoor de functionaliteit van de klasse IDLanROIGroup te gebruiken. In deze klasse kunnen meerdere ROI's opgeslagen worden, en is er direct functionaliteit beschikbaar die van toepassing is bij het rekenen met VOI's. Zo kan er worden gecontrolleerd of punten binnen of buiten een ROI/VOI vallen, en kan er een masker worden opgevraagd van een bepaalde VOI. Met dit masker kan weer gerekend worden bij inhoud/oppervlaktemetingen.



Figuur 4: Vooraanzicht ROIGroup

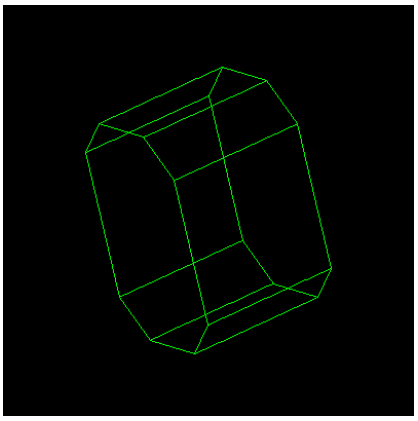
In theorie is een VOI feitelijk meerdere ROI's achter elkaar. Zo wordt een IDLgrROIGroup ook getekend. Deze IDLgrROIGroup erft van IDLanROIGroup, en heeft als extra functionaliteit dat het een Atom Graphics Object is. Dit houdt in dat deze kan worden gebruikt bij het tekenen van dit object bij een Object Graphics modus. Object Graphics is een keuzemogelijkheid bij het maken van afbeeldingen binnen IDL. Bij Object Graphics wordt er een virtuele omgeving gemaakt waarin de objecten (lijnen, afbeeldingen, etc.) in een 3D omgeving worden geplaatst. Deze objecten worden dan aan modellen gekoppeld, waarna de modellen geroteerd kunnen worden. Uiteindelijk ontstaat er een boomstructuur van modellen (waaraan objecten zijn gekoppeld). Door het hoogste model in de boomstructuur te



Figuur 5: Zijaanzicht ROIGroup

verplaatsen/roteren, krijgt de gebruiker het idee dat deze door de virtuele omgeving "loopt". Het nadeel van de IDLgrROIGroup is dat deze alle ROI's apart tekent. Zodra dit object van de zijkant wordt bekeken (lees: het model wordt 90 graden gedraaid), worden er allemaal streepjes weergegeven. Ieder streepje stelt dan de rand van een ROI voor.

De oplossing hiervoor is om van IDLanROIGroup de functie computeMesh aan te roepen. Deze functie geeft alle knooppunten van alle ROI's die in de klasse worden bijgehouden. Deze knooppunten zijn zo geordend dat de punten van één ROI allemaal achter elkaar staan in een array. Doordat deze punten gegroepeerd zijn, mogelijk een functie te schrijven die een array maakt met definities van verbindingen. Hiermee is ook mogelijk alleen de buitenkant (de randen) met elkaar te verbinden, en de punten binnen de ROIGroup niet te verbinden. Door deze knooppunten en verbindingsinformatie aan een IDLgrPolyline mee te geven, is het mogelijk een object te tekenen wat alleen de randen van de ROIGroup laat zien.



Figuur 6: IDLgrPolyline op basis van ROIGroup waarbij de punten aan de buitenkant met elkaar zijn verbonden

4.3 Schalen van een IDLanROIGroup

Het schalen van een IDLanROIGroup heeft een aantal nadelen als dit gebeurt in de planes-richting. In deze richting worden namelijk niet de planes vergroot, maar worden ze op een andere positie gezet. Deze nieuwe positie is dan de factor van de voorgaande positie ten opzichte van de oorsprong van het object. Dit heeft als effect dat de functie computeMask tussen twee planes de waarde 0 terug geeft. Dit effect treedt op omdat de planes verplaatst worden. In ons geval is dat niet gewenst. Het schalen van een VOI houdt immers in dat deze groter of kleiner wordt gemaakt, en dat deze geen hiaten vormt.

Als gevolg hiervan ontstaat een probleem bij het gebruik van IDLanROIGroup. Hierbij wordt er niet meer naar waarheid bepaald of een punt binnen, of buiten de ROIGroup valt.

Voorbeeld

Als voorbeeld nemen we twee vlakken. Vlak 1 ligt op 0 in de z-richting, terwijl vlak 2 op 10 ligt. Vlak 1 heeft een hoogte van 10, terwijl vlak 2 een hoogte van 4 heeft. Zodra er een containsPoints wordt uitgevoerd op z-richting 4 en hoogte 10, zal de functie teruggeven dat deze zich binnen het vlak bevind, terwijl deze er buiten ligt. Wordt er containsPoints uitgevoerd op z-richting 6 en hoogte 10, zal de functie teruggeven dat deze zich buiten het vlak bevind. De conclusie hieruit is dat de functie containsPoints op zoek gaat naar de dichtst bijzijnde ROI, en bepaalt daarop of deze zich binnen of buiten het "vlak" van het plane bevind.

Een ander probleem door het schalen in de planes-richting ontstaat bij het opvragen van het masker. Aangezien de punten nu verder uit elkaar liggen (en er een tussenopening is), zullen er 0-waardes op plekken in de array ontstaan die eigenlijk binnen de VOI liggen. Dit terwijl het wenselijk is dat als de VOI wordt uitgetrokken, het niet zo mag zijn dat er open plekken ontstaan.

Beide resultaten, naar aanleiding van het schalen, zijn niet wenselijk voor het framework. Er zal daarom gekozen worden om ROI's toe te voegen (voor elke nieuwe z-waarde) of te verwijderen als er een schaling in de planes-richting wordt toegepast.

4.4 Realisatie nieuwe situatie

Bij de realisatie van het 3D regions tekenen zijn een aantal factoren anders. Allereerst de koppeling van het model met de GUI-objecten. Bij 2D regions kan hier heel gemakkelijk gebruik worden gemaakt van het IDLgrRoi object. IDLgrRoi is een Atomic Graphic Object. Dit houdt in dat deze getekend kan worden bij een drawWidget dat werkt met Object Graphics. Verder biedt het functies die in een model (model binnen het model-view-controller principe) zouden staan (bijvoorbeeld containsPoints om te kijken of een punt binnen of buiten de ROI valt).

Binnen IDL is de klasse IDLanROIGroup beschikbaar die modelberekeningen kan uitvoeren. Dit object heeft ook een variant (IDLgrRoiGroup) die erft van IDLanROIGroup. Dit object (IDLgrRoiGroup) is ook weer een Atomic Graphic Object, en kan een visuele presentatie van de ROI weergeven. Het nadeel van dit object is dat de weergave vanuit de zijkant alleen streepjes zijn (zie figuur 5). Deze streepjes zijn de randen van de IDLanROI (of IDLgrROI) objecten. Bij de huidige implementatie is deze weergave niet acceptabel, aangezien alleen de randen van de gehele IDLanRoiGroup zichtbaar moeten zijn in de andere aanzichten. Door alleen randen zichtbaar te maken ontstaat voor de eindgebruiker het idee dat het één volume is waarmee hij aan het werk is, en niet allerlei losse plakjes.

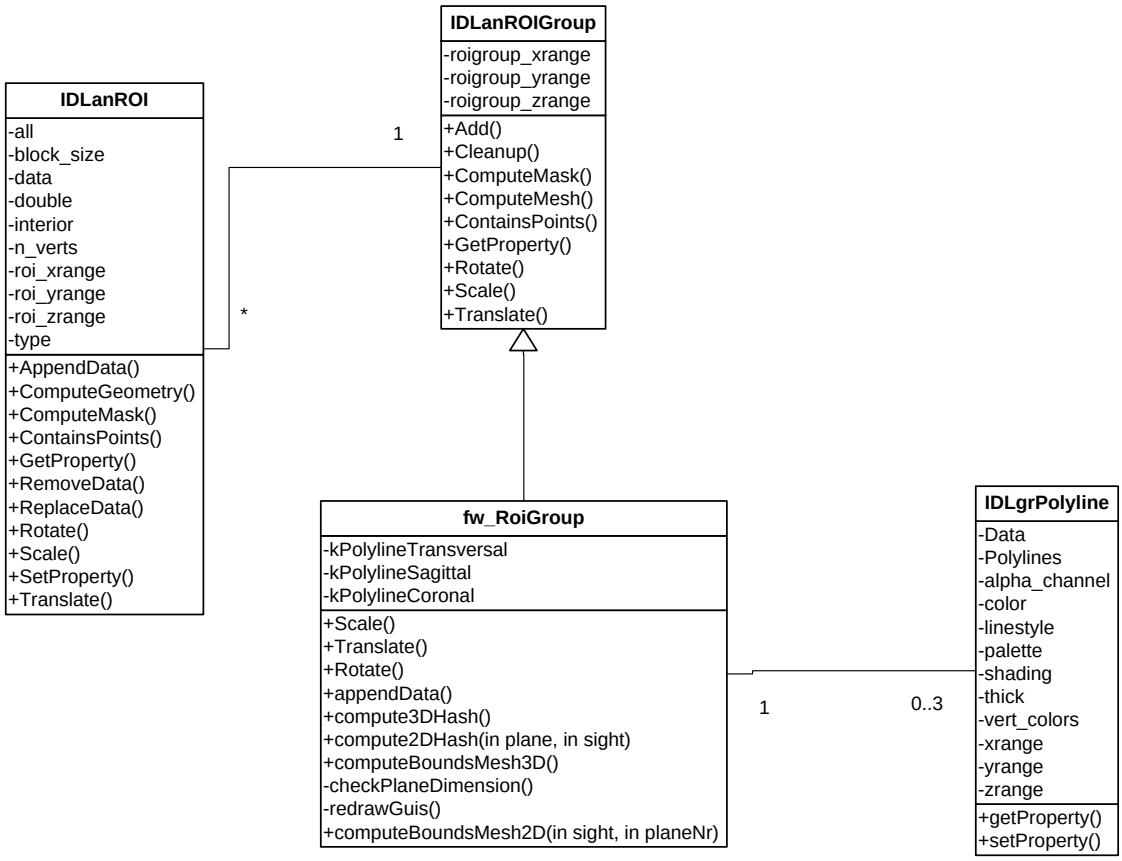
In principe zijn er dus twee objecten nodig. Een IDLanROIGroup met daarin meerdere IDLanROI objecten, en een IDLgrPolyline. Hierbij zal het IDLanROIGroup object zorgen voor de modelberekeningen, terwijl het IDLgrPolyline object zorgt voor de weergave. Dit zorgt ervoor dat er twee objecten bijgehouden moeten worden, namelijk het IDLanROIGroup object, en het IDLgrPolyline object.

De beste oplossing voor dit probleem lijkt om een nieuwe klasse te maken die overerft van IDLanROIGroup. De overerving van IDLanROIGroup krijgt dan de naam fw_RoiGroup. De klassen fw_RoiGroup en IDLgrPolyline zullen dan bi-directioneel aan elkaar gekoppeld zijn. Hierdoor zal er nooit een object "kwijt" raken in het framework. De informatie wordt dan opgehaald vanuit het fw_RoiGroup object. Bijvoorbeeld als er punten worden toegevoegd zal via IDLgrPolyline het fw_roiGroup object worden opgehaald. Het roiGroup object zal bepalen in welk plane (in feite welk IDLanROI object) de data zal worden toegevoegd. Hierna zal deze een update sturen naar alle bekende polyline objecten, zodat deze hun beeld kunnen bijwerken.

Het bovenstaande is een voorbeeld hoe 3D ROI's geïmplementeerd worden, zonder dat er veel aanpassingen aan het bestaande framework worden uitgevoerd. Er is ook gekeken naar de mogelijkheid om één object te maken dat van IDLanROIGroup en IDLgrPolyline overerft, maar hierbij zal er per widget een nieuw object gemaakt moeten worden, aangezien het aanzicht anders is. Hierdoor zijn er 3 verschillende VOI's voor hetzelfde doel nodig, en ontstaat er een grotere kans dat er dataverschillen ontstaan als de drie objecten individueel worden aangepast.

4.5 Uitbreiding klassendiagram

In onderstaande figuur (figuur 4) is de uitbreiding op het klassendiagram weergegeven. De objecten die met "IDL" beginnen zijn objecten die beschikbaar zijn in de programmeertaal. De objecten die met "fw_" beginnen zijn objecten die in het framework gebruikt worden.



Figuur 7: Uitbreiding klassendiagram

De klasse fw_RoiGroup wordt een overerving van IDLanROIGroup. Hierdoor krijgt deze de gewenste functionaliteit voor wat betreft het creëren en bijhouden van ROI objecten. Ook zal deze bijhouden welke polylines een aanzicht van de RoiGroup weergeven, zodat deze worden bijgewerkt zodra de RoiGroup wordt aangepast. Tevens zal deze klasse bij zijn destructie de gekoppelde IDLgrPolyline objecten verwijderen uit de drawWidgets en het geheugen.

4.5.1 Beschrijving van fw_RoiGroup

Er zal één klasse worden toegevoegd aan het framework, namelijk fw_RoiGroup. In het bovenstaande stuk is beschreven wat de globale functionaliteit van deze klassen zijn. In deze sub-paragraaf wordt beschreven wat de methodes van deze klassen als gedrag/functionaliteit hebben.

Methode:	fw_RoiGroup::Scale
Gedrag / Functionaliteit:	Het schalen van de RoiGroup, waarbij zo nodig ROI's worden verwijderd of toegevoegd. Als laatste zal er en sein worden gegeven naar alle IDLgrPolyline objecten dat deze de weergave moeten aanpassen.
Pre:	Vector [x,y,z]: De vector die bepaald wat geschaald moet worden in de drie dimensies (x, y en z). Double factor: De factor van de schaling. 1 betekent geen schaling, 2 betekent een verdubbeling, 0,5 betekent een halvering
Post:	Het object is geschaald, en de IDLgrPolyline objecten zijn bijgewerkt. Er is géén returnwaarde.

Methode:	fw_RoiGroup::Translate
Gedrag / Functionaliteit:	Het verplaatsen van de RoiGroup. Als laatste zal er en sein worden gegeven naar alle IDLgrPolyline objecten dat deze de weergave moeten aanpassen.
Pre:	Vector [x,y,z]: Een vector waarin staat hoeveel pixels/planes een object verplaatst moet worden.
Post:	Het object is verplaatst, en de IDLgrPolyline objecten zijn bijgewerkt zodat deze (waar nodig) een nieuwe locatie hebben gekregen.. Er is géén returnwaarde.

Methode:	fw_RoiGroup::Rotate
Gedrag / Functionaliteit:	Het roteren van de RoiGroup. Als laatste zal er en sein worden gegeven naar alle IDLgrPolyline objecten dat deze de weergave moeten aanpassen.
Pre:	Vector [x,y,z]: Een vector waarin staat hoeveel pixels/planes een object geroteerd moet worden. Double degrees: het aantal graden dat de RoiGroup gedraaid moet worden
Post:	Het object is gedraaid, en de IDLgrPolyline objecten zijn bijgewerkt zodat de knooppunten (waar nodig) een nieuwe locatie hebben gekregen. Er is géén returnwaarde.

Methode:	fw_RoiGroup::AppendData
Gedrag / Functionaliteit:	Deze methode zal een punt toevoegen aan de RoiGroup. Specifiek zal dit op een IDLanRoi object gebeuren. Welk IDLanRoi object binnen de RoiGroup zal zijn, hangt af van de richting waarin de planes zijn gemaakt.
Pre:	Vector [x,y,z]: Een vector waarin de x,y,z locatie van het nieuwe punt staat
Post:	Het punt is toegevoegd aan een (nieuw) IDLanRoi object, en de IDLgrPolyline objecten zijn bijgewerkt.

Methode:	fw_RoiGroup::Compute3DMask
Gedrag / Functionaliteit:	Deze methode zal op basis van de aanwezige punten een 3D mask teruggeven. Deze array zal dezelfde afmetingen hebben als de totale x,y,z van de DICOM afbeeldingen, waarbij waardes binnen de ROIGroup een waarde 255 hebben, en buiten de ROIGroup de waarde 0.
Pre:	Er zijn geen parameters bij de aanroep van deze functie.
Post:	Een 3D array waarin staat welke punten binnen de ROIGroup vallen (waarde 255) en welke punten buiten de ROIGroup vallen (0)

Methode:	fw_RoiGroup::Compute2DMask
Gedrag / Functionaliteit:	Deze methode zal op basis van de aanwezige punten een 2D mask teruggeven. Deze hash is een 2D array, waarbij de afmetingen dezelfde zijn als de afmetingen die bij het aanzicht horen. Als een pixel/punt binnen de RoiGroup valt, zal deze in de array de waarde 255 krijgen. Als een pixel/punt buiten de RoiGroup valt, zal deze de waarde 0 krijgen.
Pre:	String sight: een string die in hoofdletters de waarde "transversal", "sagittal" of "coronal" bevat. Integer planeNr: het nummer van het plane waarop de return-array wordt gebaseerd (planeNr >= 0)
Post:	Een 2D array waarin staat welke punten binnen de ROIGroup vallen (waarde 255) en welke punten buiten de ROIGroup vallen (0) bij een bepaald aanzicht op een bepaald planeNr.

Methode:	fw_RoiGroup::ComputeBoundsMesh3D
Gedrag / Functionaliteit:	Deze methode zal berekenen welke punten aan de rand van de RoiGroup liggen. Zo ja zal deze in een array worden opgenomen die alle rand-punten bevat. Ook zal er een array worden gemaakt met punten die aan elkaar verbonden moeten worden. Deze beide array's zullen in een structure worden teruggegeven.
Pre:	Er zijn geen parameters bij de aanroep van deze functie.
Post:	Een structure waarin twee arrays zijn verwerkt. De eerste array (<structure>.vertices) bevat alle knooppunten, de tweede array (<structure>.connectivity) bevat alle verbindingen.

Methode:	fw_RoiGroup::checkPlaneDimension
Gedrag / Functionaliteit:	Deze methode zal op basis van de aanwezige IDLanRoi objecten bepalen in welke richting deze zijn gemaakt.
Pre:	Er zijn geen parameters bij de aanroep van deze functie
Post:	Een integer waarin wordt aangegeven welke dimensie de planes zijn. 0 betekent planes in de x-richting, 1 betekent planes in de y-richting, 2 betekent planes in de z-richting.

Methode:	fw_RoiGroup::redrawGuis
Gedrag / Functionaliteit:	Deze methode zal de GUI's die bekend zijn bij RoiGroup een sein sturen om te updaten. Hierbij zullen de knooppunten en verbindingen worden meegegeven.
Pre:	Er zijn geen parameters bij de aanroep van deze functie
Post:	Er is een sein gestuurd naar de VoiPolyline objecten om hun aanzicht aan te passen.

Methode:	fw_RoiGroup::ComputeBoundsMesh2D
Gedrag / Functionaliteit:	Deze methode zal op basis van aanzicht en plane nummer een structure teruggeven waarin staat welke punten getekend moeten worden, en op welke manier deze met elkaar verbonden moeten worden.
Pre:	String sight: een string die in hoofdletters de waarde "transversal", "sagittal" of "coronal" bevat. Integer planeNr: het nummer van het plane waarop de return-array wordt gebaseerd (planeNr >= 0)
Post:	Een structure waarin twee arrays zijn verwerkt. De eerste array (<structure>.vertices) bevat alle knooppunten die in het betreffende plane bestaan. De tweede array (<structure>.connectivity) bevat de informatie over het verbinden van deze knooppunten.

Functioneel ontwerp iteratie 3

Inleiding

Dit document heeft tot doel de wensen en eisen van de opdrachtgever helder te formuleren, en staat aan de basis voor het technisch ontwerp van het eindproduct.

De belangrijkste reden voor het gebruik van dit document is om met de opdrachtgever duidelijk af te spreken welke functionaliteit er wel en niet wordt gerealiseerd. Als een volgende fase is afgelopen (bijvoorbeeld het technisch ontwerp, of de realisatie), kan er aan de hand van dit document bekeken worden of alle gevraagde functionaliteit ook verwerkt is.

Aangezien het project initieel als doel heeft om het iMed framework uit te breiden, zal het document niet veel eisen bevatten die voor de eindgebruiker van belang zijn. De meeste eisen komen van de programmeurs die gebruik maken van het framework. Deze programmeurs gebruiken het framework om applicaties te ontwikkelen die door het Siemens e.Soft systeem worden aangeroepen. Hiervan stamt ook het doel van dit project, namelijk het uitbreiden van de aanwezige functionaliteit. Met deze uitbreiding zal het voor de programmeurs makkelijker worden om applicaties te schrijven die in drie dimensies berekeningen moet uitvoeren.

Dit functioneel ontwerp dient als uitgangspunt voor de tweede iteratie van dit project. De functionele eisen en Use Cases zijn voor deze iteratie aangepast ten opzichte van de tweede iteratie. Voor de functionele eisen en Use Cases uit iteratie 2 wordt verwezen naar het Functioneel Ontwerp van iteratie 2 (Iteratie 2 – Functioneel Ontwerp / Functioneel Ontwerp v02.docx).

Aan het eind van deze iteratie (iteratie 3), zal dit document gebruikt worden om te bepalen of de gevraagde functionaliteit ook allemaal geïmplementeerd is. Dit kan aan de hand van de use cases. Men zou aan het einde van deze iteratie een kleine testapplicatie kunnen schrijven, waarna de functionaliteit vergeleken kan worden met de use cases. Men kan dan de "normal flow" als acties uitvoeren. Als controle kan dan worden gekeken of het "resultaat" of "alternatieve flow" uit de use case overeenkomt met de actie die het programma uitvoert.

Zo niet, zal dit in de uitloop van iteratie 3 (zie bijlage B van het PiD) worden aangepast. Mocht dit niet haalbaar zijn in de uitloop van iteratie 3, zal er met de opdrachtgever overlegd worden hoe dit wordt opgelost.

Inhoudsopgave

1 Analyse

Op dit moment gebruikt men een framework om applicaties te schrijven die door het e.Soft systeem worden aangeroepen. Dit framework heeft de naam iMed, en bestaat uit een library-set van meerdere klassen. Deze library set zijn losse code-bestanden, en kunnen in een project toegevoegd worden. In iteratie 1 is het mogelijk gemaakt om een 2D GUI-type[9] te creëren. Iteratie 2 is hiermee verder gegaan. Tijdens iteratie 2 is het mogelijk gemaakt om in de gecreëerde weergave een Volume of Interest te tekenen (kortweg VOI). Dit is een Region of Interest, maar dan in 3 dimensies.

De bedoeling van de framework-uitbreiding (in iteratie 3) is om de gebouwde functionaliteit van iteratie 1 en 2 uit te breiden. In iteratie 3 zal een weergave ontwikkeld worden waarin de data en de VOI's in 3 dimensies weergegeven kunnen worden binnen één tekenvenster. Dit is tevens de laatste iteratie van dit project. Hieronder staan de eigenschappen waaraan dit uitgebreide model na iteratie 3 (aan het einde van dit project) aan moet voldoen:

- De gerealiseerde uitbreidingen moeten een meerwaarde zijn op de bestaande code in het framework. Dat wil zeggen dat de extra functies die zijn gemaakt, ook gebruiksvriendelijk zijn voor de programmeur, en dat deze ook nieuwe mogelijkheden biedt voor de eindgebruiker.
- Het framework moet gemakkelijk zijn om mee te werken c.q. te programmeren
- Het framework moet geen geheugenlekken veroorzaken waardoor een pc kan vastlopen
- De gerealiseerde uitbreidingen zal het mogelijk maken om eenvoudig VOI's[10] te tekenen.
- Het framework dient op deze VOI's standaardberekeningen uit te kunnen voeren. Deze standaardberekeningen zijn bijvoorbeeld het berekenen van het aantal counts binnen een oppervlak/volume, het uitrekenen van het aantal Bq/ml, of het bepalen van de pixelinhoud. Complexe berekeningen dienen in de applicaties verwerkt te worden. Dit wordt gedaan om het framework open te houden voor nieuwe ontwikkelingen.
- De gerealiseerde uitbreiding zal met minimaal een 2D GUI-type opgeleverd moeten worden (namelijk 3 verschillende 2D vensters), en waar mogelijk met een 3D GUI-type[11].

[9] De 2D GUI zal bestaan uit drie verschillende aanzichten op het object, namelijk sagittaal (van de linker zijkant), coronaal (van de voorkant) en transversaal (van de bovenkant).

[10] VOI is de afkorting voor een Volume of Interest. Een VOI bestaat uit meerdere ROI's (Region of Interest) die achter elkaar in meerdere planes bevinden. Hierdoor ontstaat een ROI. Aangezien dit uit drie dimensies bestaat, wordt er dan gesproken over een Volume of Interest.

[11] De 3D GUI-type zal bestaan uit één venster, waarin het object in verschillende richtingen gedraaid kan worden. Hierdoor is er maar één weergave nodig.

2 Programma van Eisen

In dit hoofdstuk worden de wensen en eisen van het programma beschreven en gecategoriseerd. Allereerst zal er een splitsing worden gemaakt in functionele eisen en niet-functionele eisen. In de functionele eisen staat de functionaliteit van het systeem beschreven (functies die de toekomstige gebruiker met het systeem kan uitvoeren). De niet-functionele eisen zijn eisen die niet als functionaliteit aantoonbaar zijn, maar wel invloed hebben op het product.

Het categoriseren van de functionele eisen gebeurt aan de hand van de MoSCoW-methode. Hierbij worden de eisen ingedeeld in Must haves (eisen die in het product moeten), Should haves (eisen die eigenlijk wel in het product zouden moeten, maar een mindere prioriteit hebben), Could haves (eisen die niet in het product hoeven, maar die gemaakt worden als er tijd over is) en Won't haves (eisen die zeker niet in het product komen). Deze methode geeft bij een grote tijdsdruk minder conflicten als er beslist moet worden welke functionaliteit al dan niet opgeleverd kan worden.

2.1 Functionele eisen

In deze paragraaf worden de functionele eisen van iteratie 3 beschreven. Zoals in het PiD beschreven zal iteratie 3 zich voornamelijk richten op het 3-dimensionaal weergeven van dicom-data en VOI's. De eisen zijn van toepassing op het functioneren van het systeem, en niet op factoren van het systeem (zoals snelheid en beveiliging).

Must Haves

Er zal een uitbreiding gemaakt worden op de weergavemodule van het framework. Hierbij moet het mogelijk zijn om de dicom-data en VOI-data weer te geven in een projectie die voor de eindgebruiker de indruk wekt dat het 3D is.

Deze representatie kan door de eindgebruiker worden gedraaid of verplaatst in een gewenst aanzicht.

Should Haves

Verder kan een programmeur de weergavemodule zo bewerken, dat het mogelijk is om de kleuren en het doorlatingsvermogen van de voxels te bepalen. Deze kan hij dan (waar nodig) verwerken in de uiteindelijke applicatie die wordt ontwikkeld met het framework.

Could haves

Er kan nog een grafische component gemaakt worden waarin de intensiteit-sliders zijn verwerkt. Dit zodat de programmeur dit niet meer hoeft te doen.

Won't Haves

Het gebruik van meerdere tijdsframes om door de tijd verschillende afbeeldingen te bekijken. Dit is ook niet gerealiseerd in de voorgaande iteraties.

2.2 Niet-functionele eisen

Naast alle functionele eisen (wat moet het programma kunnen) zijn er ook eisen die niet direct aan het programma zelf worden gesteld, maar op de werking van het programma van toepassing zijn. Dit zijn de niet-functionele eisen. In dit hoofdstuk zijn deze eisen opgesomd en beschreven.

Code

De applicatie dient te worden geschreven in de programmeertaal IDL en er dient (waar mogelijk) gebruik gemaakt te worden van de aanwezige functies in het framework.
De keuze om het framework initieel in IDL te schrijven is omdat de Siemens e.Soft systemen externe IDL programma's op de machine kunnen starten, en dat deze binnen e.Soft zichtbaar zijn. Hierdoor ontstaat er een integratie en lijkt het voor de eindgebruiker alsof de applicatie bij het e.Soft systeem hoort. Verder is het een krachtige taal om berekeningen op beelden te kunnen uitvoeren. Voor dit doel wordt de taal ook bij de NASA en in de meteorologie gebruikt.

Alle publieke functies/methoden dienen beschreven te worden met PRE en POST situatie. Hiermee wordt beschreven wat de input en output van een functie zal zijn. Verder zal er in het kort omschreven worden hoe de functie dit precies uitvoert.

Er wordt gewerkt met een SVN server. Dit zorgt ervoor dat iedereen (de opdrachtnemer en de bedrijfsbegeleider/consultant) altijd de actuele versie beschikbaar heeft. Als iemand aan een codefragment heeft gewerkt, zal hij deze op de server plaatsen. Dit moet met goed commentaar gebeuren. Hierdoor kan men nagaan waarom een bepaalde wijziging is gemaakt, en wanneer dat is gebeurt (en door wie).

Hardware

De applicatie zal uiteindelijk op een Siemens e.Soft systeem gaan werken. Deze systemen worden door Siemens zelf geleverd, en waar nodig wordt de hardware uitgebreid. Hierdoor wordt aan het uitgebreide framework de hardware eis gesteld dat het op de bestaande e.Soft systemen moet werken. Wel speelt de beeldresolutie een punt bij de e.Soft systemen. Dit omdat er een screenshot wordt gemaakt van de applicatie, en deze gebaseerd is op een bepaalde ruimte binnen het beeldscherm.

Performance

Natuurlijk is het streven om berekeningen zo snel mogelijk uit te voeren, anders wordt er door de gebruikers snel een negatieve stempel op geplakt en is het in (klinische) praktijk minder bruikbaar. Als een systeem namelijk lang werkt heeft, zal er gezocht worden naar alternatieven die sneller werken, of zullen taken verplaatst worden naar het einde van de dag. Dit kan weer als gevolg hebben dat het stellen van een diagnose langer duurt, dit is dan ook minder prettig voor de patiënt. Van de andere kant is het wel zo dat een systeem kwalitatief goed en betrouwbaar moet zijn. De keuze is dan om een middenweg te vinden die kwalitatief goed is, maar ook redelijk snel is.

Veiligheid

De veiligheid berust hier vooral op privacy. Aangezien het framework wordt gebruikt om uitbreidingen te schrijven op het e.Soft systeem, zal dit punt wegvallen. De privacy van patiëntdata zal worden gewaarborgd door het e.Soft systeem zelf. Wel zal het natuurlijk zo zijn dat de uiteindelijke applicatie, wat met het iMed framework geschreven wordt, niet met patiëntendata moet gaan "rommelen".

Taal

De taal voor de grafische kant van de uitbreidingen zal Engels zijn. Ook zullen er medische termen in voorkomen om op één lijn te blijven met de medisch onderlegde eindgebruiker van het e.Soft systeem.

Gebruiksgemak

De eindgebruiker (gericht op de GUI onderdelen) zal een basiskennis hebben van medische apparatuur. Hierdoor zullen schermvormgevingen gebaseerd worden op hun kennis of ervaring. Dit werkt bevorderend voor het gebruiksgemak, omdat de gebruikte systemen een zelfde soort interface hebben en dit het leerproces voor een applicatie verkort.

Betrouwbaarheid

De betrouwbaarheid van de uitbreiding zal hoog moeten zijn. Er worden namelijk allerlei berekeningen en analyses uitgevoerd op basis van de informatie die het framework geeft. Dit betekent dat de performance misschien iets lager wordt, maar dat de kwaliteit/betrouwbaarheid wel hoger is. Dit is belangrijk omdat applicaties die met het uitgebreide framework worden gebouwd, gebruikt kunnen worden om medische beslissingen te nemen. Als er dan structureel een fout zit in het framework kan dit consequenties opleveren voor de diagnose of behandeling van een patiënt.

Handleidingen

In principe zullen er geen handleidingen voor de eindgebruiker geschreven worden. Dit aangezien het een uitbreiding op het framework is, en geen applicatie op zichzelf. Daarom zal het Technisch Ontwerp als handleiding dienen.

Documentatie

Zoals hierboven geschreven dient het Technisch Ontwerp als documentatie, samen met dit document en het commentaar in de code. Dit commentaar zal door middel van pre en post omschrijvingen bij functies gebeuren. Bij uitgebreide functies zullen er binnen de code regels commentaar komen te staan.

3 Use Cases

In dit hoofdstuk staan alle use cases uitgewerkt, in de vorm van de MoSCoW-methode. Deze komen overeen met de eisen in paragraaf 2.1. Hierbij zijn de use cases voor de Won't Haves weg gelaten. Aangezien deze componenten allemaal betrekking hebben op applicaties die met het iMed framework gemaakt kunnen worden, wordt er gesproken over de gebruiker van e.Soft als Actor. Als men de programmeur als actor zou gebruiken, zou dit het doel van het project mislopen, aangezien er functionaliteit gebouwd wordt, waardoor de programmeur minder werk heeft met stukken code die toch altijd hetzelfde zijn. Het is daarbij belangrijk om te documenteren wat de functionaliteit hiervan is voor de eindgebruiker. Hoe de technische implementatie hiervan gaat, is af te leiden uit het Technisch Ontwerp.

3.1 Must Haves

Naam:	Weergeven van de DICOM data in 3D
Actor:	De gebruiker van e.Soft
Pre:	De DICOM data is beschikbaar in de applicatie
Normal flow:	- De actor geeft aan dat hij de DICOM data in een 3D venster wil weergeven.
Resultaat:	- De actor krijgt als resultaat dat er een 3D venster wordt opgebouwd waarin de DICOM data wordt weergegeven met een standaard instelling betreffende kleuren en doorlaatbaarheid.
Alternatieve flow:	- Het aantal planes is te weinig om weer te geven in 3D, de gebruiker krijgt hiervan een melding.

Naam:	Weergeven van VOI in 3D venster
Actor:	De gebruiker van e.Soft
Pre:	De weergave van 3D DICOM data is geslaagd
Normal flow:	- De actor geeft aan dat hij de VOI's zichtbaar wil maken in het 3D venster.
Resultaat:	- De actor krijgt als resultaat dat de VOI op de juiste plaats zichtbaar is binnen het 3D venster. Deze VOI zal een bepaalde mate van doorlaatbaarheid hebben om het originele DICOM object zichtbaar te houden.
Alternatieve flow:	

Naam:	Roteren van het weergaveobject
Actor:	De gebruiker van e.Soft
Pre:	De weergave van 3D DICOM data is geslaagd
Normal flow:	- De actor geeft aan dat hij/zij het gehele weergave wil roteren - De actor sleept met de muis over het object, om het weergaveobject te roteren
Resultaat:	- De actor krijgt als resultaat dat de weergave is geroteerd in de richting die hij/zij met de muis heeft aangegeven.
Alternatieve flow:	

Naam:	Verplaatsen van het weergaveobject
Actor:	De gebruiker van e.Soft
Pre:	De weergave van 3D DICOM data is geslaagd
Normal flow:	- De actor geeft aan dat hij/zij de gehele weergave wil roteren - De actor sleept met de muis over het object, om het weergaveobject te verplaatsen.
Resultaat:	- De actor krijgt als resultaat dat de weergave is verplaatst in de richting die hij/zij met de muis heeft aangegeven
Alternatieve flow:	

Naam:	Zoomen op het weergaveobject
Actor:	De gebruiker van e.Soft
Pre:	De weergave van 3D DICOM data is geslaagd
Normal flow:	- De actor geeft aan dat hij/zij op de gehele weergave een zoom wil toepassen. - De actor geeft met de muis aan hoe groot de zoomfactor op de weergave zal zijn.
Resultaat:	- De actor krijgt als resultaat dat de weergave is veranderd. Op de weergave is verder in of uit gezoomd, op basis wat de actor heeft aangegeven.
Alternatieve flow:	

3.2 Should Haves

Naam:	Bepalen minimale grijswaarde
Actor:	De programmeur van de e.Soft uitbreiding
Pre:	De weergave van 3D DICOM data is geslaagd
Normal flow:	- De actor geeft aan dat hij de minimale grijswaarde wil veranderen in waarde x. (x is een vrij te kiezen waarde tussen 0 en 255)
Resultaat:	- De actor krijgt als resultaat dat de weergave is veranderd. De minimale grijswaarde is toegepast, waardoor het beeld lichter of donkerder is gemaakt.
Alternatieve flow:	- De minimale grijswaarde mag niet boven de maximale grijswaarde komen

Naam:	Bepalen maximale grijswaarde
Actor:	De programmeur van de e.Soft uitbreiding
Pre:	De weergave van 3D DICOM data is geslaagd
Normal flow:	- De actor geeft aan dat hij de maximale grijswaarde wil veranderen in waarde x. (x is een vrij te kiezen waarde tussen 0 en 255)
Resultaat:	- De actor krijgt als resultaat dat de weergave is veranderd. De maximale grijswaarde is toegepast, waardoor het beeld lichter of donkerder is gemaakt.
Alternatieve flow:	- De maximale grijswaarde mag niet onder de minimale grijswaarde komen

Naam:	Bepalen minimaal doorlatingsvermogen
Actor:	De programmeur van de e.Soft uitbreiding
Pre:	De weergave van 3D DICOM data is geslaagd
Normal flow:	- De actor geeft aan dat hij het minimaal doorlatingsvermogen wil veranderen in waarde x. (x is een vrij te kiezen waarde tussen 0 en 255)
Resultaat:	- De actor krijgt als resultaat dat de weergave is veranderd. Het minimaal doorlatingsvermogen is toegepast. Hierdoor krijgt de gebruiker het idee dat sommige pixels "doorzichtig" worden, of dat pixels minder "doorzichtig" worden.
Alternatieve flow:	- Het minimaal doorlatingsvermogen mag niet boven het maximaal doorlatingsvermogen komen.

Naam:	Bepalen maximaal doorlatingsvermogen
Actor:	De programmeur van de e.Soft uitbreiding
Pre:	De weergave van 3D DICOM data is geslaagd
Normal flow:	- De actor geeft aan dat hij het maximaal doorlatingsvermogen wil veranderen in waarde x. (x is een vrij te kiezen waarde tussen 0 en 255)
Resultaat:	- De actor krijgt als resultaat dat de weergave is veranderd. Het maximaal doorlatingsvermogen is toegepast. Hierdoor krijgt de gebruiker het idee dat sommige pixels "doorzichtig" worden, of dat pixels minder "doorzichtig" worden.
Alternatieve flow:	- Het maximaal doorlatingsvermogen mag niet onder het minimaal doorlatingsvermogen komen.

3.3 Could Haves

Naam:	Bepalen minimale grijswaarde
Actor:	De gebruiker van e.Soft
Pre:	De weergave van 3D DICOM data is geslaagd
Normal flow:	- De actor past de minimale grijswaarde aan met behulp van een component op het beeldscherm.
Resultaat:	- De actor krijgt als resultaat dat de weergave is veranderd. De minimale grijswaarde is toegepast, waardoor het beeld lichter of donkerder is gemaakt.
Alternatieve flow:	- De minimale grijswaarde mag niet boven de maximale grijswaarde komen. Hiervan krijgt de actor een melding

Naam:	Bepalen maximale grijswaarde
Actor:	De gebruiker van e.Soft
Pre:	De weergave van 3D DICOM data is geslaagd
Normal flow:	- De actor past de maximale grijswaarde aan met behulp van een component op het beeldscherm.
Resultaat:	- De actor krijgt als resultaat dat de weergave is veranderd. De maximale grijswaarde is toegepast, waardoor het beeld lichter of donkerder is gemaakt.
Alternatieve flow:	- De maximale grijswaarde mag niet onder de minimale grijswaarde komen. Hiervan krijgt de actor een melding

Naam:	Bepalen minimaal doorlatingsvermogen
Actor:	De programmeur van de e.Soft uitbreiding
Pre:	De weergave van 3D DICOM data is geslaagd
Normal flow:	- De actor past het minimaal doorlatingsvermogen aan met behulp van een component op het beeldscherm.
Resultaat:	- De actor krijgt als resultaat dat de weergave is veranderd. Het minimaal doorlatingsvermogen is toegepast. Hierdoor krijgt de gebruiker het idee dat sommige pixels "doorzichtig" worden, of dat pixels minder "doorzichtig" worden.
Alternatieve flow:	- Het minimaal doorlatingsvermogen mag niet boven het maximaal doorlatingsvermogen komen. Hiervan krijgt de actor een melding.

Naam:	Bepalen maximaal doorlatingsvermogen
Actor:	De programmeur van de e.Soft uitbreiding
Pre:	De weergave van 3D DICOM data is geslaagd
Normal flow:	- De actor past het maximaal doorlatingsvermogen aan met behulp van een component op het beeldscherm.
Resultaat:	- De actor krijgt als resultaat dat de weergave is veranderd. Het maximaal doorlatingsvermogen is toegepast. Hierdoor krijgt de gebruiker het idee dat sommige pixels "doorzichtig" worden, of dat pixels minder "doorzichtig" worden.
Alternatieve flow:	- Het maximaal doorlatingsvermogen mag niet onder het minimaal doorlatingsvermogen komen. Hiervan krijgt de actor een melding.

Technisch ontwerp iteratie 3

Inleiding

Dit document heeft tot doel een heldere beschrijving te geven van de aanpassingen voor het iMed framework. Dit document zal ook als leidraad dienen bij de daadwerkelijke bouw/aanpassing van het framework. In dit document komt naar voren waarom sommige keuzes betreffende architectuur/programmatuur gemaakt zijn.

De belangrijkste reden voor het gebruik van dit document is om, met de bedrijfsbegeleider, duidelijk af te spreken welke aanpassingen/uitbreidingen er gemaakt worden. Deze aanpassingen/uitbreidingen zullen resulteren tot een realisatie van de wensen en eisen uit het functioneel ontwerp. Naderhand zal ook met het Functioneel Ontwerp worden vergeleken of de aanpassingen daadwerkelijk zorgen voor de nieuwe functionaliteit.

Inhoudsopgave

1 Het platform

Het platform waarop het iMed framework wordt gebruikt zijn de Siemens e.Soft workstations. In principe zijn dit normale computers. Hierbij kan gedacht worden aan een computer wat heden ten dage in de computerwinkel te koop is. Mocht het zo zijn dat de specificaties niet voldoende zijn, zal dit bij Siemens worden aangekaart, en zullen zij ervoor zorgen dat de pc (waar nodig) een upgrade krijgt.

Deze computers werken op dit moment met Windows XP als besturingssysteem. Hierop wordt een applicatie gedraaid die ontwikkeld is door Siemens. Deze applicatie, genaamd e.Soft, verzorgt het aansturen van een scanner, en het verwerken en weergeven van de DICOM-images.

Met deze DICOM-images is het mogelijk om, in de Siemens software, analyses uit te voeren. Veel van deze analyses zijn al aanwezig in de applicatie, maar het is ook mogelijk om zelf analyses toe te voegen. Deze kunnen in IDL worden geschreven. Hiervoor levert Siemens de Virtual Machine van IDL mee. Door het gebruik van een Virtual Machine is het mogelijk om dezelfde bytecode uit te voeren op verschillende besturingssystemen. De Virtual Machine vertaalt namelijk de bytecode naar machinecode.

De applicatie (geschreven in IDL) wordt dan aangeroepen door het e.Soft programma. Deze wordt dan schermvullend boven op het e.Soft programma gedraaid. Zodra deze wordt afgesloten worden er screenshots genomen van de resultaten, en worden deze (via een nieuw DICOM-bestand) terug gegeven aan het e.Soft programma.

2 Gekoppelde bestanden en applicaties

Zoals in de vorige paragraaf is uitgelegd, wordt de IDL software door e.Soft gestart. Het e.Soft programma zal dan ook de benodigde files meegeven om de studie uit te voeren.

Zodra een studie is afgerond, is het mogelijk om met het iMed framework screenshots te maken van resultaten. Deze kunnen dan als afbeeldingen bij de studie worden opgeslagen.

Aangezien deze functionaliteit al in het framework aanwezig is en niet aangepast hoeft te worden (zie Functioneel Ontwerp), zal dit document hier niet verder op ingaan.

3 Software

Zoals in het Functioneel Ontwerp is uitgelegd, zal de software in IDL worden ontwikkeld. Als ontwikkelomgeving zal hiervoor een standaard computer van het UMC St. Radboud gebruikt worden. Hierop zal IDL 6.3 van ITT VIS geïnstalleerd worden. Deze computer zal dus als ontwikkelomgeving gebruikt worden.

Het huidige iMed framework zal als resource gebruikt worden. Dit omdat de huidige functionaliteit behouden moet blijven, en de functies hiervan wellicht in de uitbreiding gebruikt kunnen worden. Ook zal het nieuwe framework in zijn geheel moeten samenwerken. Het mag niet mogelijk zijn dat door de nieuwe functionaliteit sommige functies niet meer kunnen samenwerken.

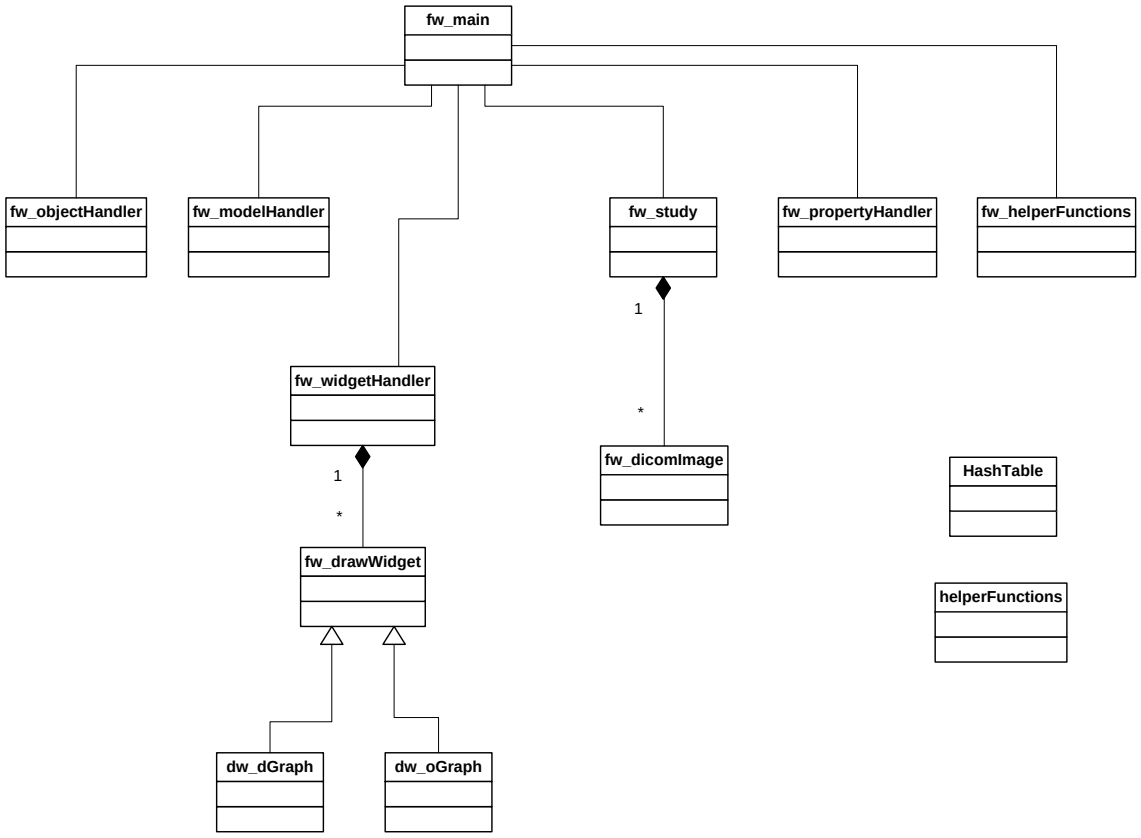
4 Programmastructuur

Dit hoofdstuk is ingedeeld in twee paragrafen. Allereerst zal de huidige structuur worden besproken. Hiertoe zijn een klassendiagram en drie sequentiediagrammen gemaakt. Het klassendiagram geeft de globale structuur van de applicatie aan, en laat zien welke objecten er mogelijk aangemaakt worden, en welke verhouding deze tot elkaar hebben. De sequentiediagrammen zijn gemaakt om inzicht te geven in de flow van de applicatie, en hoe de objecten binnen de applicatie gebruikt worden.

4.1 Huidige Situatie

4.1.1 Klassendiagram

Hieronder in het figuur staat het huidige domeinmodel. Dit model is beperkt tot alleen de naam van de klasse, en de relaties van de objecten. Dit is gedaan omdat het klassendiagram niet binnen de afmetingen van dit document zou passen. Onder het figuur staat uitgelegd wat de functie van de verschillende klassen is.



Figuur 1: Domeinmodel

fw_main: dit is het basisobject van het framework. Hierin worden de links (pointers) naar de objecten fw_objectHandler, fw_modelHandler, fw_widgetHandler, fw_study, fw_propertyhandler en fw_helperfunctions bijgehouden. Dit object moet binnen de applicatie beschikbaar zijn, om het aanroepen van het framework mogelijk te maken.

fw_modelHandler: In IDL is het mogelijk om gebruik te maken van Object Graphics. Hierbij worden objecten in een virtuele 3D wereld gemaakt, en worden deze via een view gepresenteerd. Fw_modelHandler zorgt ervoor dat objecten aan een model gekoppeld worden, en dat deze worden onderhouden. Deze modellen kunnen vervolgens aan de view van dw_oGraph gekoppeld worden. Dit koppelen gebeurt dan in de widgetHandler.

fw_widgetHandler: Deze klasse beheert de widgets en views die gekoppeld worden aan de models uit fw_modelHandler. Als er via fw_widgetHandler een nieuw object wordt aangemaakt (bijvoorbeeld een ROI, een lijn, of dergelijke), wordt deze doorgestuurd naar fw_modelHandler, om dit aan het juiste model te koppelen. Ook regelt deze klasse de communicatie als een ROI op meerdere views zichtbaar moet zijn, en als een nieuwe ROI gemaakt wordt. Deze klasse geeft dan aan de klasse van het type fw_drawWidget door dat er een refresh moet worden uitgevoerd.

fw_drawWidget: Deze klasse is de parent van de klassen dw_dGraph en dw_oGraph. De functionaliteit van deze klasse beperkt zich tot een aantal functiedeclaraties wel dan niet met een body. Verder worden hier variabele opgeslagen (via de propertyHandler) die van toepassing zijn op beide type drawWidgets (dw_dGraph en dw_oGraph).

dw_dGraph: Deze klasse erft van fw_drawWidget, en wordt gebruikt voor objecten van het type WIDGET_DRAW, die gebruik maken van direct graphics. Deze modus biedt minder grafische mogelijkheden, maar is wel sneller bij het tekenen van afbeeldingen. Als voorbeeld van de beperkte mogelijkheden is er geen mogelijkheid om eigen onderdelen (bijvoorbeeld ROI's) te tekenen of te bewerken met direct graphics. Hierdoor heeft deze klasse binnen het framework minder functies en mogelijkheden.

dw_oGraph: Deze klasse erft van fw_drawWidget, en wordt gebruikt voor objecten van het type WIDGET_DRAW, die gebruik maken van Object Graphics. Hierbij wordt een model (vanuit fw_modelHandler) gekoppeld aan deze view. Ook worden hier de events van het widget afgehandeld (bijvoorbeeld de muisevents bij het tekenen van een Region of Interest). Het gebruik van object graphics (gekoppeld met een eigen model) brengt wel als nadeel met zich mee dat het meer geheugen en snelheid kost, dan de direct graphics. Daarom dient men een afweging te maken voor het gebruik van direct of object graphics. Het maken van deze afweging gebeurt op applicatie-niveau.

fw_study: Deze klasse zorgt ervoor dat een (of meerdere) DICOM bestand(en) in de applicatie wordt ingeladen. Aangezien een DICOM bestand meer informatie bevat dan alleen de afbeeldingen, worden deze in de study klasse afgehandeld.

fw_dicomImage: In deze klasse worden de afbeeldingen uit het DICOM bestand bewaard. Verder is het hier mogelijk om een aantal basisfuncties op de afbeeldingen uit te voeren. Deze afbeeldingen zijn in principe gewoon 2 of 3 dimensionale array's met getallen. Ook kan er uit de headers informatie worden gelezen betreffende de naam van de patiënt, scanner instellingen en andere specifieke informatie betreffende het bestand.

fw_propertyHandler: De propertyHandler maakt het mogelijk om d.m.v. keywords (eventueel overeenkomstig met de velden binnen een object) allerlei informatie op te slaan. Als er een veld met dezelfde naam beschikbaar is binnen het object, zal de waarde van de eigenschap (property) in het betreffende veld worden opgeslagen. Is het opgegeven veld niet beschikbaar binnen het object (bijvoorbeeld bij een nieuwe, of tijdelijke property), zal dit als referentie in een globale HashTable worden opgeslagen. Als alle properties gezet en gelezen worden via deze propertyHandler, is het dus mogelijk om in een applicatie (ontwikkeld met het framework) extra informatie op te slaan. Dit bied als voordeel dat het framework flexibel blijft, en dus niet voor iedere applicatie structurele aanpassingen behoeft.

Verder bied deze klasse de mogelijkheid om deze propertys te importeren/exporteren van/naar een XML bestand. Hierdoor wordt het mogelijk om dynamische applicatie-informatie op te slaan zodat de informatie hergebruikt kan worden. Ook nadat het programma opnieuw wordt gestart.

fw_helperFunctions: Deze klasse bestaat uit allerlei hulpfuncties. De hulpfuncties maken gebruik van de opbouw van het framework. Hier zit bijvoorbeeld een functie in om een Region of Interest te splitsen.

HashTable: IDL maakt geen gebruik van collections. Collections zijn objecten die datasets op verschillende manieren kunnen beheren. Voor verschillende type datasets zijn verschillende type collection objecten beschikbaar. Een voorbeeld hiervan is een HashTable (of HashMap). Hierbij wordt een key gekoppeld aan een bepaalde waarde. De key wordt dan gekoppeld aan een adres in het geheugen, waardoor zeer snel de waarde bij de gegeven key opgehaald kan worden.

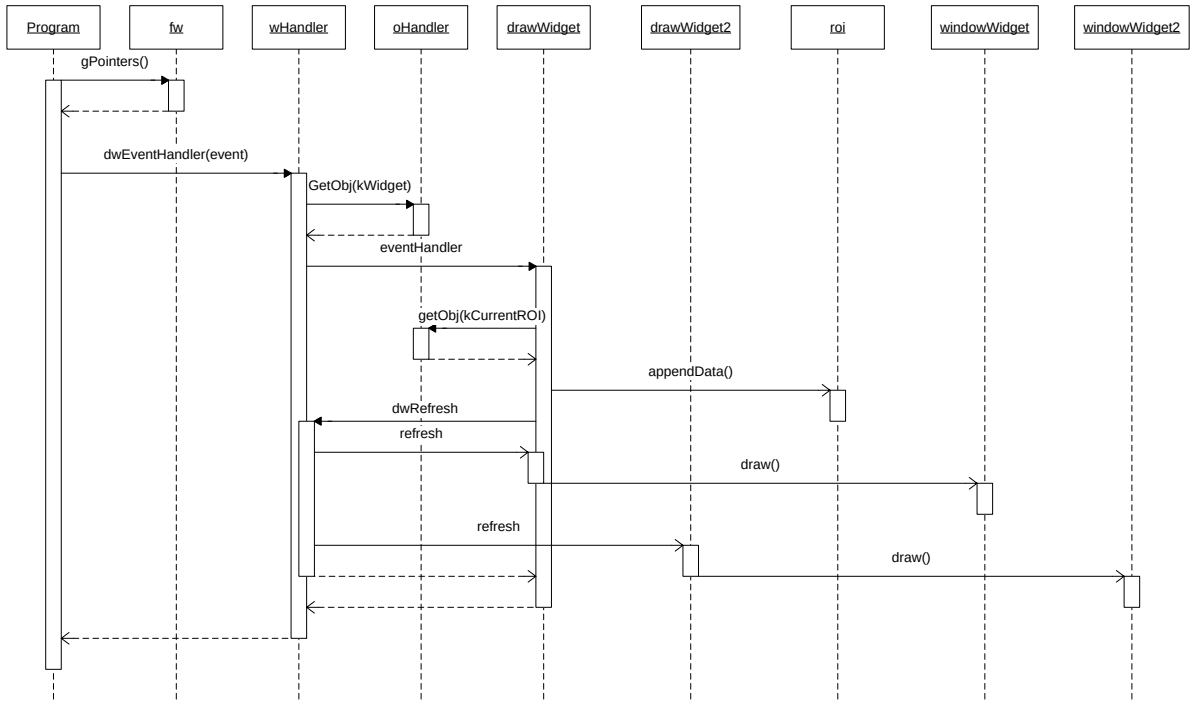
Fw_objectHandler: Deze klasse is de grote container voor het framework. Hierin wordt in een HashTable een sleutel, en de pointer van elk object opgeslagen. Dit heeft als voordeel dat objecten

altijd overal in de code terug te vinden zijn, aangezien een hashTable een lijst met sleutels kan teruggeven.

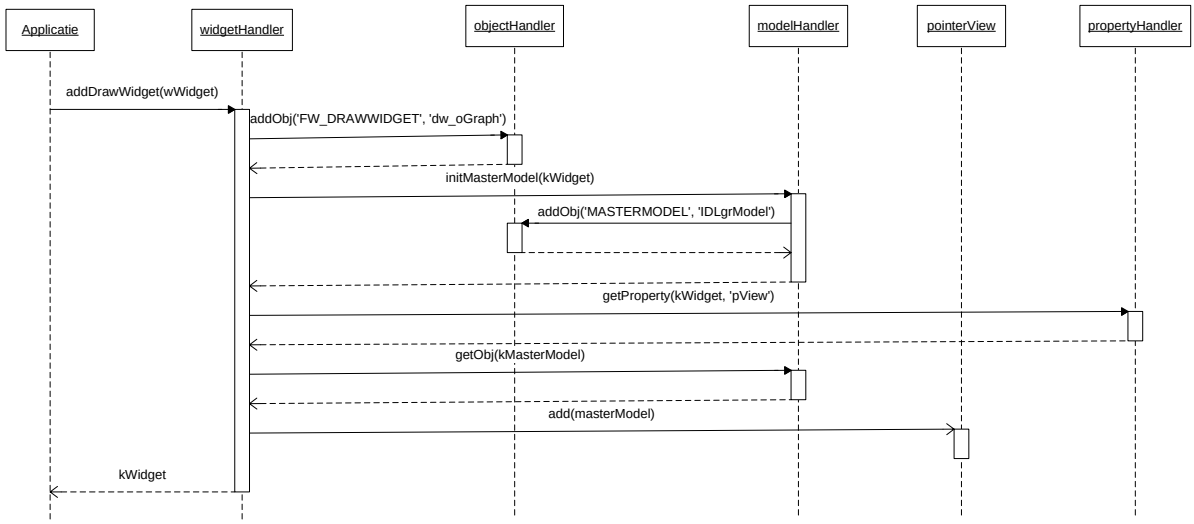
4.1.2 Sequentie Diagrammen

In het eerste diagram is beschreven hoe een event tijdens het bijwerken van een ROI wordt afgehandeld. Hiermee wordt bedoeld wat er met het event gebeurt als de gebruiker de muisknop ingedrukt houdt, en met de muis beweegt. Hierbij wordt een nieuw coördinaat aan de ROI meegegeven.

In het tweede diagram is beschreven hoe een widget, getekend met de grafische editor van IDL, gekoppeld wordt binnen het framework. Hierbij wordt er een view gemaakt die aan het widget wordt gekoppeld. Aan deze view wordt het top-model uit de boomstructuur gehangen. Hierdoor kan de boomstructuur aangepast worden, en het widget worden bijgewerkt als het model wordt aangepast.



Figuur 2: aanpassen ROI

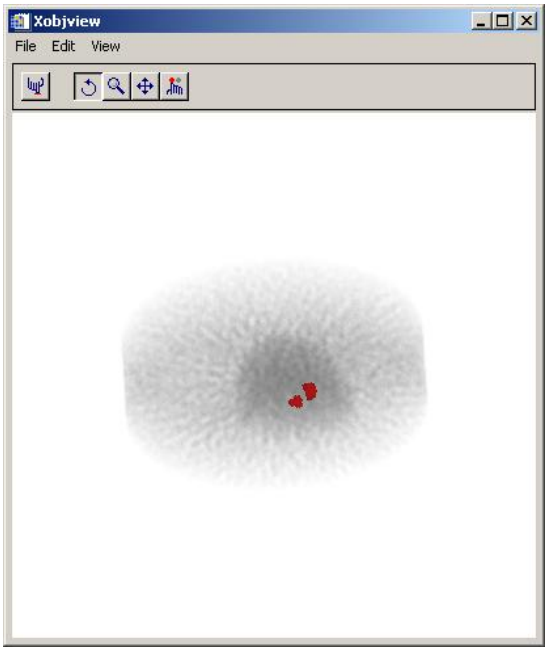


Figuur 3: Koppelen van widget aan framework

4.2 De IDLex objecten

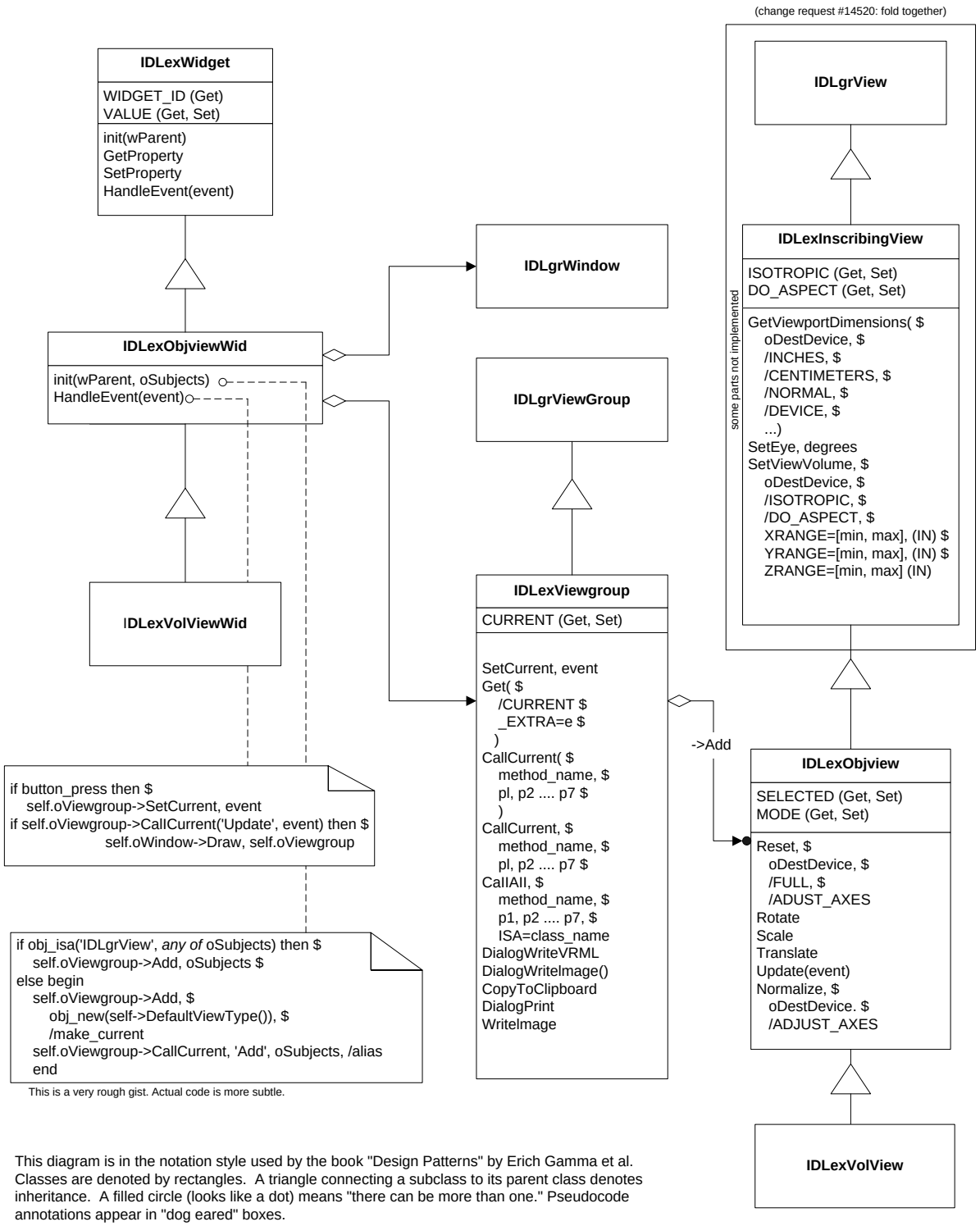
Binnen IDL zijn er een aantal experimentele klassen beschikbaar. Hiervan is de API gedocumenteerd, maar de onderliggende code is niet veel van bekend. De naam van deze experimentele klassen begint met IDLex. Verder zijn deze objecten geschreven in IDL. De broncode van deze objecten is hierdoor beschikbaar in de installatiemap van IDL. Op basis hiervan kan de functionaliteit achterhaald worden, en kan er een klassendiagram gemaakt worden.

Een implementatievoorbeeld van de IDLex objecten is xobjview. Aan dit object kan tijdens de initialisatie een aantal atom graphics objects gekoppeld worden. Deze worden onderliggend aan een model gekoppeld, en worden gecentreerd weergegeven in een nieuw venster. In principe bevat xobjview de functionaliteit die gevraagd wordt voor iteratie 3, maar biedt het niet de flexibiliteit aangezien er geen objecten of eigenschappen on-the-fly toegevoegd, verwijderd of aangepast kunnen worden. Er zal iedere keer een nieuw venster gecreëerd moeten worden. Verder is het mooier om een dergelijk object binnen het huidige widget te gebruiken, en niet in een nieuw venster.



Figuur 4: het xobjview object

Aangezien de functionaliteit wel bruikbaar was, maar er een aantal probleempunten aan zaten, is er uitgezocht hoe het IDLex domeinmodel eruit ziet. Dit model staat op de volgende pagina (figuur 5). Op de pagina erna staat uitgelegd wat de globale functie per object is (bron: http://www.paulsorenson.com/underthehood.html).



Figuur 5: Domeinmodel IDLex objecten

IDLexWidget: Dit is de basisklasse betreffende widgets in het IDLex systeem. Door het gebruik van deze klasse wordt er afgedwongen om een aantal functies te gebruiken.

IDLexObjviewWid: Dit widget wordt gebruikt door xobjview. Deze klasse initialiseert namelijk de menubalk, objectbuttons, en genereert events voor het bijwerken/aanpassen van de weergave.

IDLexVolViewWid: Deze klasse is weer een implementatie op IDLexObjviewWid. Hierbij wordt het mogelijk gemaakt om met extra sliders, buttons, checkboxes , radiobuttons en comboboxen mogelijk gemaakt om de weergave aan te passen.

IDLexViewgroup: Deze klasse is een uitbreiding op de klasse IDLgrViewGroup . Deze klasse houdt bij welk object binnen de container wordt gebruikt. Dit is het current-object waar een aantal functies direct op uitgevoerd kunnen worden, zonder dat het object uit de container gehaald hoeft te worden.

IDLexObjview: Dit object verzorgt de 3D presentatie. Het object erft indirect van IDLgrView en kan daardoor aan een widget_draw gekoppeld worden. Het bezit de extra eigenschappen om een Atom Graphics Object aan een model te koppelen, of om een model te koppelen. Dit model/object wordt dan gecentreerd voor weergave. Het IDLexObjview object kan aangeroepen worden om de afbeelding te roteren, scalen (zoom) of transleren. Dit kan door een directe aanroep op functies, of door gebruik van de event handler.

IDLexInscribingView: Deze klasse is een uitbreiding op IDLgrView. Door het gebruik van deze klasse is het eenvoudiger om eigenschappen van een view aan te passen. Hierin zijn ook allerlei controles gebouwd om te kijken of de waardes die opgegeven worden wel reëel zijn.

4.3 Keuze van IDLex object

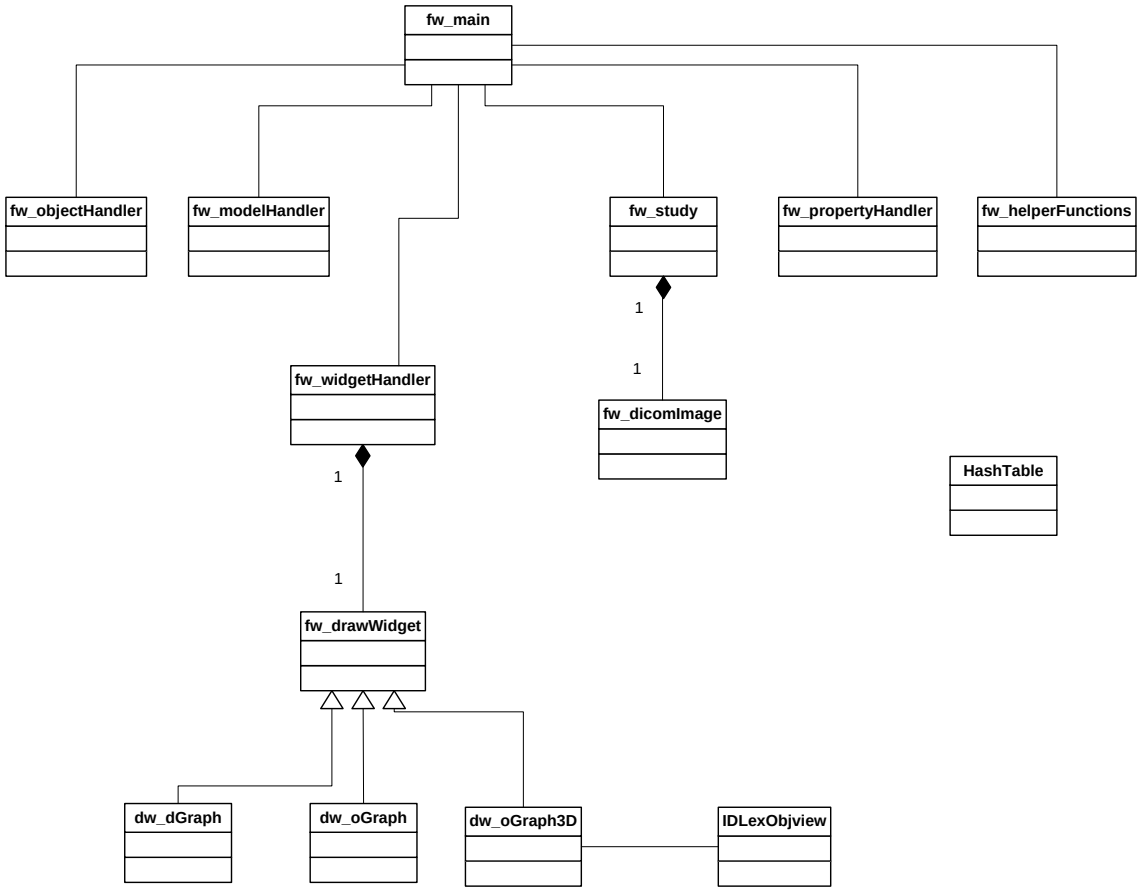
Aangezien er een aantal eisen aan de weergave gesteld zijn, is er een afweging gemaakt welk object uit het IDLex systeem de functionaliteit het beste afspiegelde.
Er is gekozen om IDLexObjview te gebruiken. Deze klasse biedt de functionaliteit om zelf de weergaveobjecten on-the-fly aan te passen. Verder is deze klasse te implementeren binnen het huidige weergavesysteem, aangezien het (indirect) erft van IDLgrView.
Dit komt omdat IDLexObjview de basisfunctionaliteit van de weergave bevat. Alle extra grafische componenten zoals buttons, sliders, etc. zijn in dit object weggelaten. Deze componenten kunnen in het framework of in de GUI editor gebouwd worden, waardoor de 3D weergave flexibel blijft voor gebruik in het framework. Door deze flexibele aanpak wordt het mogelijk koppelingen te leggen tussen verschillende IDLexObjviews, of andere componenten.

4.4 Uitbreiding klassendiagram

Voor het toevoegen van 3D weergave functionaliteit zal het klassendiagram uitgebreid moeten worden. Binnen het framework zijn twee weergavetypes beschikbaar die beide afbeeldingen kunnen presenteren in 2D. Aangezien 3D presentatie van informatie andere functies met zich meebrengt is het verstandig om het framework uit te breiden met een nieuwe klasse. Deze zal zich op hetzelfde niveau bevinden als fw_dGraph en fw_oGraph.

De nieuwe klasse zal de naam fw_oGraph3D krijgen. Het is namelijk een implementatie waarbij gebruik wordt gemaakt van Object Graphics, en zal deze 3D informatie gaan presenteren.

Net zoals fw_oGraph zal fw_oGraph3D een 1:1 container zijn. Het verschil zal in de functies en variabelen die gekoppeld zijn aan fw_oGraph3. Deze zijn namelijk gebaseerd op 3D weergave. Functies voor het tekenen of aanpassen van objecten zullen hierin niet voorkomen. Verder zijn er functies beschikbaar voor het aanpassen van kleurtabellen en doorzichtigheid van pixels binnen een bepaalde range. De beschrijving van de functies staan in de volgende paragraaf.

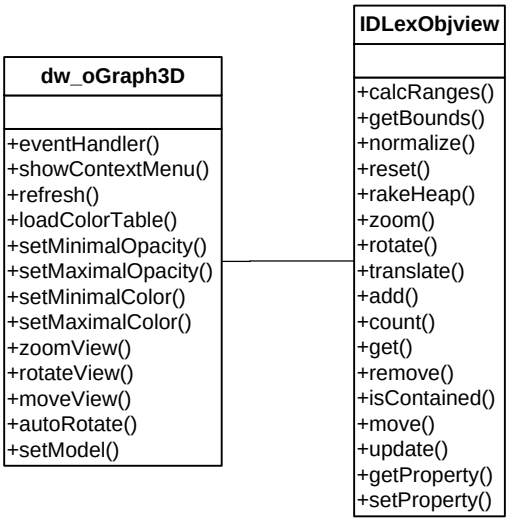


Figuur 6: uitbreiding domeinmodel

4.4.1 Beschrijving van fw_RoiGroup

In deze sub-paragraaf zullen de functies van dw_oGraph3D worden beschreven. Zoals in voorgaande tekst te lezen is deze klassen relevant voor de 3D uitbreiding van het framework.

Op de volgende wordt per functie wordt uitgelegd waarvoor deze dient.



Figuur 7: Uitbreiding klassendiagram

Methode:	Fw_oGraph3D::eventHandler
Gedrag / Functionaliteit:	Deze methode is de eventHandler voor alle muis-acties die plaatsvinden op het oGraph3D object. Deze acties kunnen doorgestuurd worden naar andere functies waar nodig is.
Pre:	Het het object is geïnitialiseerd, en er is een event-structure meegegeven.
Post:	Er is een actie uitgevoerd/doorverwezen op basis van de event-structure.

Methode:	Fw_oGraph3D::showContextMenu
Gedrag / Functionaliteit:	Deze methode zal een menu laten zien binnen het view-object. Deze methode zal worden aangeroepen zodra er met de rechtermuisknop binnen het view-object wordt geklikt. De informatie die in het menu zal staan is relevant voor het bewerken/aanpassen van de weergave.
Pre:	De coördinaten voor het tekenen van het menu zijn verplicht.
Post:	Het menu is boven op het view-object getekend.

Methode:	Fw_oGraph3D::refresh
Gedrag / Functionaliteit:	Deze methode zal aan het huidige window doorgeven dat het opnieuw getekend moet worden. Dit zal als gevolg hebben dat mogelijk aangebrachte wijzigingen in het IDLexObjview object worden weergegeven.
Pre:	Er is een window beschikbaar
Post:	Het window wordt opnieuw getekend

Methode:	Fw_oGraph3D::loadColorTable
Gedrag / Functionaliteit:	Deze methode zal een nieuwe/andere kleurtabel inladen, en toepassen op het IDLexObjview object.
Pre:	Er is een IDLexObjview object aanwezig
Post:	De nieuwe kleurtabel wordt toegepast op het IDLexObjview object, en wordt weergegeven zodra er een refresh wordt uitgevoerd.

Methode:	Fw_oGraph3D::setMinimalOpacity
Gedrag / Functionaliteit:	Deze methode zal de doorlaatbaarheidstabel aanpassen. Onder de minimale opacity zijn pixels zeer doorlaatbaar. Boven deze minimalOpacity zijn pixels zeer ondoorlaatbaar.
Pre:	Er is een IDLexObjview object aanwezig
Post:	De nieuwe doorlaatbaarheid wordt toegepast op het IDLexObjview object, en wordt weergegeven zodra er een refresh wordt uitgevoerd

Methode:	Fw_oGraph3D::setMaximalOpacity
Gedrag / Functionaliteit:	Deze methode zal de doorlaatbaarheidstabel aanpassen. Boven de maximale opacity zijn pixels ondoorlaatbaar. Onder deze maximalOpacity zijn pixels zeer ondoorlaatbaar.
Pre:	Er is een IDLexObjview object aanwezig
Post:	De nieuwe doorlaatbaarheid wordt toegepast op het IDLexObjview object, en wordt weergegeven zodra er een refresh wordt uitgevoerd

Methode:	Fw_oGraph3D::setMaximalOpacity
Gedrag / Functionaliteit:	Deze methode zal de doorlaatbaarheidstabel aanpassen. Boven de maximale opacity zijn pixels ondoorlaatbaar. Onder deze maximalOpacity zijn pixels zeer ondoorlaatbaar.
Pre:	Er is een IDLexObjview object aanwezig
Post:	De nieuwe doorlaatbaarheid wordt toegepast op het IDLexObjview object, en wordt weergegeven zodra er een refresh wordt uitgevoerd

Methode:	Fw_oGraph3D::zoomView
Gedrag / Functionaliteit:	Deze methode zal aan het IDLexObjview doorgeven dat de zoom veranderd is. De methode zal controleren of de opgegeven waarde wel geldig is voor het toepassen van de zoom.
Pre:	Er is een IDLexObjview object aanwezig
Post:	De zoomfactor wordt toegepast op het IDLexObjview object. Deze zoom wordt zichtbaar zodra er een refresh wordt uitgevoerd.

Methode:	Fw_oGraph3D::rotateView
Gedrag / Functionaliteit:	Deze methode zal aan het IDLexObjview doorgeven dat het globale model wordt geroteerd. De methode zal controleren of de opgegeven waarde wel geldig is voor het toepassen van de rotatie.
Pre:	Er is een IDLexObjview object aanwezig
Post:	De rotatie wordt toegepast op het IDLexObjview object. Deze rotatie wordt zichtbaar zodra er een refresh wordt uitgevoerd.

Methode:	Fw_oGraph3D::moveView
Gedrag / Functionaliteit:	Deze methode zal aan het IDLexObjview doorgeven dat het globale model wordt verplaatst. De methode zal controleren of de opgegeven waarde wel geldig is voor het toepassen van de verplaatsing.
Pre:	Er is een IDLexObjview object aanwezig
Post:	De verplaatsing wordt toegepast op het IDLexObjview object. Deze verplaatsing wordt zichtbaar zodra er een refresh wordt uitgevoerd.

Methode:	Fw_oGraph3D::autoRotate
Gedrag / Functionaliteit:	Deze methode zal op basis van het opgegeven tijdsinterval de afbeelding roteren, en een refresh uitvoeren.
Pre:	Er is een IDLexObjview object aanwezig
Post:	Per opgegeven tijdseenheid wordt de afbeelding geroteerd en opnieuw getekend.

Methode:	Fw_oGraph3D::setModel
Gedrag / Functionaliteit:	Deze methode zal het gegeven model met onderliggende objecten koppelen aan het IDLexObjview object, en het mogelijk maken om een weergave hiervan te geven.
Pre:	Er is een IDLexObjview object aanwezig
Post:	Het model is gekoppeld. Zodra er een refresh wordt uitgevoerd, zal het object zichtbaar worden.

Bijlage D: Logboek

Week 1

Deze week was de introductie op de afdeling. Hierbij is mij duidelijk geworden hoe een PET-scanner werkt, en wat hiermee gemeten wordt. Verder is er uitgelegd waarom er eigen software wordt geschreven, en wat de voordelen hiervan zijn. Ook is hierbij uitgelegd op welke manier de eigen geschreven software werkt.
Verder ben ik tijdens deze week ook van start gegaan met de programmeertaal IDL. Hiervan zijn op de afdeling meerdere boeken beschikbaar, maar er zijn twee boeken die de basis van IDL goed uitleggen. Een van de boeken gaat over de GUI, en het andere over de basisprincipes van de programmeertaal.
Allereerst ben ik begonnen om een command-line programma te maken waarbij er standaard acties uitgevoerd worden (for/while/if). Daarna ben ik verder gegaan met het maken van klassen en de overerving hiervan. Dit gaat namelijk anders in zijn werk dan in (bijvoorbeeld) Java.
Ook ben ik gestart met het maken van het Plan van Aanpak. Hierbij heb ik de eerste conceptversie gemaakt.

Week 2

Deze week ben ik vooral bezig geweest met het ophalen en weergeven van DICOM bestanden. In deze bestanden zit heel veel header-informatie opgeslagen in een collectie vergelijkbaar met een HashMap. Verder liggen er meerdere afbeeldingen opgeslagen in één DICOM bestand. Zo is het mogelijk dat deze planes (de afbeeldingen) achter elkaar worden uitgelezen. Dit moet ook straks de mogelijkheid bieden om de afbeelding in 3D te gebruiken.
Ook ben ik bezig geweest met de grafische deel binnen de programmeertaal. Hierbij heb ik gekeken of het mogelijk is om 2D afbeeldingen in te laden, en om deze te roteren. Dit was wel mogelijk, maar dit zal niet goed werken voor 3D rendering. Dit heeft er mee te maken dat een afbeelding een apart object is binnen IDL. Het is dus niet mogelijk om meerdere afbeeldingen achter elkaar te gebruiken voor een zijaanzicht.
Wat betreft de documentatie ben ik bezig geweest met het Plan van Aanpak. Hierop heeft de bedrijfsbegeleider feedback gegeven. Deze heb ik verwerkt, en nogmaals terug gestuurd of het voor hem akkoord is. Ook ben ik met het functioneel ontwerp bezig geweest. Hierin heb ik MoSCoW beschreven wat er die iteratie uitgevoerd wordt, zijn de use cases van de eisen gemaakt, zijn de niet-functionele eisen beschreven, en heb ik een schermvoorbeeld gemaakt.

Week 3

Deze week heb ik 3 dagen gewerkt (2 dagen niet gewerkt i.v.m. carnaval). In deze drie dagen heb ik het functioneel ontwerp afgemaakt, en ben ik voorzichtig begonnen aan het klassendiagram voor in het Technisch Ontwerp. Hierbij heb ik eerst een klein prototype gemaakt waarbij de dicom informatie van 3D naar 2D informatie wordt omgezet, en daardoor verschillende aanzichten verkregen worden. Dit is eigenlijk best redelijk gelukt. Het tekenen van lijntjes hierin heb ik geprobeerd, maar daar had ik een ontwerpfout in gemaakt. Daarom ben ik toen begonnen aan een goed technisch ontwerp, om niet bij extreme programming terecht te komen. Hierdoor zou de kans groter zijn dat er een product ontstaat waarbij mijn begeleiders niet instemmen met de programmeertechniek.
Verder heb ik ook de laatste versie van het framework gekregen. Hiermee ben ik aan een verkenning begonnen, en heb hier een kleine applicatie mee proberen te schrijven. Dit om een beetje kennis te maken hoe het framework precies werkt. Hierbij liep ik tegen de build-volgorde aan. Het blijkt dat deze in de juiste volgorde moet staan, anders worden functies of objecten niet herkend. Zeer belangrijk om mee te nemen voor de verdere uitbreiding van het framework.
Bij het doornemen van het framework kwam ik er achter dat mijn kleine schets van het klassendiagram niet helemaal in het framework zal passen. Met name op het gebied van overerving en objectafhankelijkheid. In het framework worden de onderdelen los gekoppeld. Als men 3 grafische tekenobjecten heeft, wordt het lastig om deze elkaar te laten aanspreken. Dit is wel in hun voorbeeldapplicatie uitgevoerd. Dus dat zal ik begin volgende week uitzoeken, voordat ik een nieuwe versie van het klassendiagram ga maken.
Verder heb ik me nog in een onderdeel verdiept met betrekking op de grootte van ROI's. De scanners hebben namelijk een klein blur-effect, waardoor de oppervlakte/inhoud van regions (2D en 3D) andere maten hebben dan dat ze daadwerkelijk zijn. Hierover is een artikel gepubliceerd een aantal jaren geleden, waarbij werd aangetoond dat het wiskundig/statistisch mogelijk is om aan te geven hoe groot de region in werkelijkheid is. Binnen de afdeling willen ze hiermee verder gaan in 3 dimensies. Aangezien de grootte van een weefsel hierbij wordt gebaseerd op meerdere iteraties (en dit niet beschikbaar is in het huidige framework), zal hiervoor een nieuwe applicatie geschreven moeten worden.

Week 4

In deze week heb ik het functioneel ontwerp met mijn begeleiders besproken, en heb ik naar aanleiding hiervan aanpassingen gemaakt. Deze aanpassingen waren vooral gericht om dingen duidelijker en uitgebreider te beschrijven.
Verder ben ik vooral bezig geweest met het uitzoeken van het framework. Hierbij heb ik een aantal sequentiediagrammen gemaakt die betrekking hebben op de huidige functionaliteit. Na het maken van deze diagrammen ben ik verder gegaan met het technisch ontwerp voor iteratie 1. Hiervoor waren de sequentie diagrammen zeer nuttig. In het technisch ontwerp ben ik verder gegaan vanuit het huidige klassendiagram en sequentiediagram. Door deze goed te analyseren heb ik een mogelijke implementatie voor 2x 3D beelden gemaakt op dit klassendiagram, en heb ik hier ook sequentiediagrammen voor toegevoegd. Dit heb ik op het laatst van de week al ingeleverd bij de bedrijfsbegeleiders. Volgens de planning ligt dit een week voor, maar aangezien het een voorstel is, kan het goed zijn dat dit ontwerp deze week nog aangepast wordt.
Verder is afgelopen vrijdag het eerste bedrijfsbezoek van de docentbegeleider geweest. Over het plan van aanpak (PiD) waren niet veel opmerkingen. Wel moet ik op een paar plaatsen het communicatieplan en de planning aanpassen. Dit met betrekking tot de conceptversies van het afstudeerverslag en de eindpresentatie.

Week 5

In deze week ben ik vooral bezig geweest met het onderzoeken naar een aantal punten in IDL. Aangezien de dataset groot kan worden, kan het zo zijn dat er geheugen-errors ontstaan, of dat het inefficient gaat werken. Er wordt namelijk gebruik gemaakt van een 3 of 4-dimensionale array. In IDL is het gebruik van wildcards hierbij best handig, maar kan wel voor een tijdsverlies leiden. Ik ben er namelijk achter gekomen dat als de wildcards als laatste dimensies staan, dit langzamer gaat, dan dat de wildcards als eerste dimensie staan ([*,*,4] werkt sneller dan [4,*,*]). Aangezien er mogelijk met een 4D gewerkt gaat worden heb ik ook onderzocht hoe groot de dataset mocht zijn, zonder dat de eindgebruiker daar last van krijgt (doordat hij moet wachten totdat een plaatje uit deze dataset is geconstrueerd). Verder heb ik met mijn begeleiders het Technisch Ontwerp doorgesproken. Over het algemeen had ik door hoe het framework in elkaar steekt, en was de opzet voor de 3x 2D weergave wel okee. Alleen zaten hier nog wat kleine haken en ogen aan, die ik moest bijwerken.
Verder ben ik met het afstudeerverslag bezig geweest. Zo heb ik de bedrijfsorganisatie en opdracht voor een groot deel beschreven. De eerste conceptversie hiervan moet rond 27 maart af zijn, dat moet lukken.

Week 6

In het begin van deze week ben ik bezig geweest met het uitzoeken wat de snelste methode is voor het laden van bestanden, en wat de mogelijheden van multithreading binnen IDL zijn.
De uitslag van het laden van bestanden vond ik best interessant. De methode om DICOM bestanden te openen had langer tijd nodig dan de methode om bestanden op byte-niveau te openen. Op zich is dit nog verklaarbaar. Het uitlezen van het bestand via de dicom klasse ging namelijk sneller dan op byte-niveau. Dit komt omdat de DICOM klasse weet welke bytes deze moet uitlezen, terwijl op byte-niveau alles opgehaald wordt.
Verder ben ik gaan kijken wat de mogelijkheden van multithreading binnen IDL zijn. Ik kwam er op uit dat er een "thread-pool" wordt gebruikt, die IDL zelf beheert. IDL gaat namelijk kijken of een berekening lang gaat duren, en of deze veilig over meerdere processoren verdeeld kan worden. De programmeur hoeft alleen aan te geven hoe groot de brokken computaties per processor mogen zijn. Dit is in mijn toepassing niet handig. Wat ik namelijk in gedachten had was om een DICOM-service te maken. Deze zou dan niet alle bestanden (4^e dimensie) hoeven inladen, maar een aantal. Om de x aantal seconden zou er dan gecontrolleerd kunnen worden wat de huidige waarde van de 4^e dimensie is. Als deze tegen de rand van het gecachde array is, kan de service zelf (op de achtergrond) een nieuwe cache maken waarbij de huidige 4^e dimensie in het midden zit.
Deze oplossing is dus niet mogelijk in IDL. IDL heeft namelijk maar één event-que waarin alles wordt afgehandeld. Multithreading wordt alleen toegepast bij zware berekeningen. Dit is dus zeer jammer. Wat wel mogelijk is, is om een extra programma (en dus ook een extra IDL session) op de achtergrond te draaien, en om hier de informatie aan op te vragen. Dit moet besproken worden in hoe verre dat wenselijk is.
Op het laatst van de week was ik zo ver dat ik de drie weergaves zichtbaar had, en de referentielijnen in de andere afbeeldingen werden bijgewerkt. Ook heb ik het toen mogelijk gemaakt om de andere lijnen te verplaatsen, en zo het aanzicht in de andere weergaves te veranderen. Verder is er een minimale drempel geïmplementeerd. Hierdoor kan er achtergrond/ruis worden weggefilterd.

Week 7

In het begin van deze week ben ik bezig geweest met het bugfixen van iteratie 1. Er zat bijvoorbeeld een kleine bug dat het scrollen door een afbeelding buiten de bounds van de array kwam, en daardoor crashte. Deze zijn nu opgelost.

Verder ben ik begonnen met uitzoeken hoe een ROI in 3D gedefinieerd moet worden (VOI) binnen het programma. Allereerst was het idee om de IDLgrROI klasse zelf uit te breiden, zodat deze ook in de z-richting gebruikt kan worden. Het bleek dat de source van IDLgrROI niet beschikbaar was, en dat daardoor het uitbreiden van deze klasse niet verstandig zou zijn. Zelf ben ik namelijk niet bekend met het precieze interne gedrag van deze klasse, en bij uitbreiding zou er dus grote kans zijn op het ontstaan van bugs.

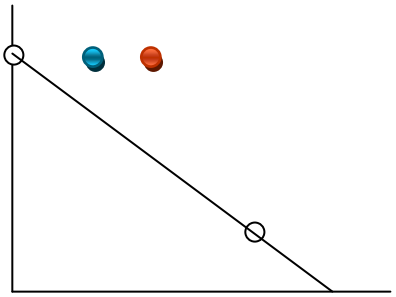
Ik ben toen verder gegaan met zoeken binnen nieuwsgroepen en fora op het 3D gebruik van regions. Hiermee kwam ik uit bij de klasse IDLgrROIGroup. Deze klasse kan meerdere ROI's als één groep beschouwen, en er ook functies van ROI's in 3D bewerkstelligen (resizen, roteren, een mask opvragen, etc.). Het enigste nadeel hieraan was dat vanuit een niet-transversaal aanzicht geen "rectangle" zichtbaar was, maar allemaal verticale lijntjes. Dit zijn namelijk de randen van de ROI's.

Dit is op te lossen door een mash te maken van alle ROI's (via de ROIGroup) en deze in een polygon object te zetten. Dit geeft als voordeel dat er een kubus wordt gegenereerd, maar dan met twee open vlakken. Namelijk de voor en achterkant van het object.

Week 8

Deze week ben ik begonnen met iteratie 2. Allereerst ben ik verder gegaan met controleren of de functies van een ROIGroup ook echt goed werken. Hier kwam ik tegen een probleem aan. ROIGroup heeft een scale functie. Hierbij dacht ik eerst dat het mogelijk was om twee ROI's te maken, en deze te scalen, zodat er een kubus ontstaat. Dit werkt ook, alleen onstaat er dan voor mijn gebruik een probleem. Namelijk dat de containsPoints functie, en de computeMask functie waardes gaan genereren die niet kloppen. containsPoints gaat dan kijken bij welk vlak het punt het dichtst in de buurt ligt, en gaat van dat vlak bekijken of het punt daarin ligt. Dit heeft als gevolg dat als er 5 ROI's zijn gecreëerd, die allemaal 10 punten van elkaar af liggen (dus 0, 10, 20, 30, 40) dat er waardes uit komen die niet waar kunnen zijn.

Bijvoorbeeld als punt 0 een hoogte 10 heeft, en punt 1 (dus op 10) een hoogte 4 heeft, zal containsPoints op hoogte 10, diepte 4 zeggen dat het punt binnen de ROI's valt, terwijl bij hoogte 10, diepte 6 zal zeggen dat deze buiten de ROI's valt. Dit is in de afbeelding hiernaast te zien. De blauwe punt is true, en valt volgens containsPoints binnen de ROI, terwijl de rode punt false is (diepte 6) en buiten de ROI valt.



De functie computeMask had ook nog een verhaal bij het scalen. Deze gaat de ROI's verder uit elkaar zetten, maar vult deze niet op. Dus als er een ROIgroup gemaakt wordt met z-waardes 1,2,3 en 4 zal er niets aan de hand zijn. Maar zodra deze 1,2,3 en 4 met een factor 10 worden gescaled, worden de waardes 10, 20, 30 en 40. Alle waardes die hiertussen liggen (21….29, 31….39 etc.) zullen een mask krijgen die 0 is, waardoor er problemen ontstaan in berekeningen.

Beide afwijkingen hebben gevolgen op berekeningen. Hierdoor kan het bijvoorbeeld zijn dat de uitslag lager of hoger is dan wat werkelijk het geval is. Dit heeft op zijn beurt weer gevolgen voor de behandeling van een patiënt (wel/geen operatie, wel/geen behandeling, etc.).

Hierdoor is besloten om bij het scalen de ROIGroup aan te vullen of te verkleinen met ROI's in de z-richting.

Week 9

Deze week is het functioneel ontwerp van iteratie 2 afgerond, en is een eerste versie van het technisch ontwerp gemaakt.

Begin van deze week had ik een voorbeeld gemaakt waarbij de mousehandler flink was aangepast om het mogelijk te maken om met 2 objecten VOI's bij te houden. Dit was volgens Laurens niet echt de bedoeling, aangezien de manier waarop een 2D region wordt getekend gelijk moet zijn aan de manier waarop een 3D region wordt gemaakt. Dit om de begrijpbaarheid van de code hoog te houden.

Daarom ben ik met een technisch ontwerp gekomen waarbij er van bestaande IDL klassen wordt geërfd. Hierbij wordt er een model bijgehouden wat de coördinaten van het VOI object bijhoudt, en wordt er per aanzicht een apart object getekend. Deze objecten zijn zo met elkaar verbonden dat het model alleen aan te roepen is via de aanzichtobjecten. Deze sturen dus ook de informatie door naar het model object. Hierdoor wordt het modelobject aangepast, en zal er een sein terug worden gestuurd naar de gekoppelde aanzichtobjecten dat deze zichzelf moeten bijwerken. Dit heeft als voordeel dat het framework denkt dat deze met één object te maken heeft, terwijl erachter wordt geregeld dat meerdere objecten betrokken zijn bij het betreffende object. Door namen van methodes zo op dezelfde manier te formuleren als dat ze gebruikt worden in het 2D model, zullen er niet veel aanpassingen nodig zijn in het framework. Hierdoor zullen er minder aanpassingen in het bestaande framework nodig zijn. Dit komt weer ten goede voor het backwards compatible houden van de code.

Week 10

Deze week ben ik bezig geweest om het functioneel ontwerp af te ronden. Hierover heb ik nog een gesprek met mijn begeleiders gehad. Zoals een paar weken geleden beschreven, is het niet mogelijk om multithreading toe te passen binnen IDL. Daarom heb ik een klein onderzoek gedaan naar de voor- en nadelen van IDL t.o.v. 3^{e} generatie programmeertalen. Als 3^{e} generatie programmeertalen heb ik Java en C# gebruikt. De redenen voor deze keuze staan in het betreffende document (of in het eindverslag).

Het resultaat van dit onderzoek was niet naar mijn eigen verwachting. Het bleek dat Java het snelst van de 3 was, en IDL het traagst. C# zat hier tussenin. Dit onderzoek was op basis van array's. Hierbij werden array's aangemaakt, gevuld, en uitgelezen.

Op twee andere punten is IDL wel sterker. Standaard beschikt het over functionaliteit voor het werken met DICOM bestanden, en heeft het een eenvoudige syntax betreffende arrays. Hierdoor worden de performancevoordelen gedeeltelijk teniet gedaan. Voor de afstudeerstage is dit verder niet relevant. Maar voor verdere ontwikkeling van het framework, en voor mijn persoonlijke kennis van de verschillende programmeertalen wel.

Ook heb ik het met mijn begeleiders gehad over het functioneel ontwerp. Ik was van mening een façade te maken die communiceert met een andere klasse. Zij vonden dit eigenlijk niet netjes, omdat deze façade ook de weergave van het object verzorgde. Een punt waar ikzelf nog niet bij stil had gestaan. Daarom is er voor gekozen om de façade uit het model te laten, en het framework op een aantal plaatsen (licht) uit te breiden.

Week 11

Deze week ben ik bezig geweest met de implementatie van iteratie 2. Allereerst hebben we maandag (na overleg) een klasse verwijderd uit het Technisch Ontwerp, aangezien deze geen extra functionaliteit bevatte, maar alleen een overerving was.

De klasse die erbij is gekomen draagt zorg voor het maken, bewerken en weergeven van een Volume of Interest. Hiermee ben ik zover gekomen dat ik op een willekeurig aanzicht een figuur kan tekenen, en dat er een lijn ontstaat op een ander aanzicht wat de representatie van de VOI voor dat type

aanzicht representeerd. Het roteren en verplaatsen van dit object is ook al mogelijk gemaakt. Komende week ga ik verder met het schalen van dit object. Hier komt het probleem met het ontbreken van planes aan de orde. Verder wordt dan ook de functie afgeschreven die de mask van het 3d object moet teruggeven. Beide functies zijn al voor een groot deel klaar, maar moeten nog afgemaakt worden.

Week 12

Deze week ben ik in het begin bezig geweest met het afmaken/bugfixen van de functionaliteit van iteratie 2. Hierbij is het mogelijk gemaakt om zelf een VOI te tekenen, en deze in een ander aanzicht uit te breiden. Dit is gelukt voor objecten in vrije vorm, maar ook voor cirkels en vierkanten. Als resultaat is dus ontstaan dat er een object getekend kan worden, en dat deze in de niet-gebruikte dimensie uitgerekt kan worden.

Het enigste probleem zat in het roteren, scalen en verplaatsen van afbeeldingen. Dit ging in eerste instantie niet goed. Later kwam ik er achter dat dit te maken had met het feit dat de arrays in een ander aanzicht eerst geconverteerd moesten worden. Daarna ging het roteren en verplaatsen goed.

Het schalen was meer werk. Allereerst moesten de schaling-vectoren geconverteerd worden. Daarna moest het object geschaald worden. Als deze schaling in de planes-dimensie was, moest er gecontroleerd worden of er roi's bijgemaakt moeten worden. Hier ging het in eerste instantie mis. Dit kwam omdat het schalen er voor zorgt dat er decimale getallen ontstaan. Planes zijn daarintegen zonder decimalen. Hierdoor was het zo dat de lijn in het ander aanzicht zomaar verdween (omdat er geen punten op 16 lagen, maar wel op 16,14 bijvoorbeeld. Daarom is er voor gekozen om na het schalen in de planes-dimensie, de waardes in deze dimensie af te ronden.

Verder kwam ik er achter dat punten niet direct aan elkaar lagen (boven, onder, links of rechts langs het andere punt). Hierdoor verdween ook af en toe een VOI in een ander aanzicht. Dit is opgelost door iedere coördinaat in de afbeelding te controleren of het punt op de rand ligt, tussen twee andere punten in.

Verder ben ik aan het afstudeerverslag bezig geweest. Hier heb ik een stuk over iteratie 2 bij geschreven, en ben ik bezig geweest om de onderzoeksresultaten in het verslag te verwerken. Het hoofdstuk wat gaat over de inhoud is dus sterk uitgebreid.

Week 13

Deze week ben ik begonnen aan iteratie 3. Hierbij heb ik gekeken of ik bestaande objecten/componenten kon gebruiken om de 3D weergave voor DICOM informatie mogelijk te maken. Dit heb ik gedaan door meegeleverde IDL objecten uit elkaar te rafelen. Hierbij bleek xobjview zeer geschikt. Hiervan was het mogelijk de weergave los te halen van de buttons en andere widgets, zodat dit in een eigen widget geplaatst kon worden. Het object wat ik hierbij verkreeg bevatte al de eigenschap om te roteren, transleren, zoomen en het object te centreren. Eigenlijk alles wat voor de 3^{e} iteratie nodig is. Wel moest ik even spelen met de doorlaatbaarheid en het kleurpallet, voordat ik de informatie op de juiste manier kon zien.

Dit heb ik dan ook aan mijn begeleiders in een klein prototype laten zien. Aangezien zij hier zeer tevreden over waren, ben ik verder gegaan met het functioneel ontwerp voor iteratie 3. Hier ben ik dan ook zeer ver mee opgeschoten.

Wat betreft iteratie 2 zit hier nog een klein probleem. Zodra er rotatie plaatsvind in een aanzicht wat de planes laat zien, gaat de representatie de mist in. De data erachter blijft wel goed. Wat er in feite gebeurt is dat de knooppunten in de ROI-planes een andere positie krijgen. Daardoor komen er punten van verschillende ROI's in één aanzicht. Hierdoor komt er een knooppuntenlijst terug die gesorteerd is op planes. Dit geeft een grote rommel aangezien de punten "door elkaar" staan. Ze staan niet in een volgorde waarbij er direct een lijn door de punten getrokken kan worden. Ik heb

geprobeerd hier een methode voor te schrijven die de punten sorteert, maar dit werkt nog niet helemaal.

Verder heb ik het 2e concept van het eindverslag opgestuurd naar mijn docentbegeleider. Aan het begin van de week heb ik een concept naar mijn bedrijfsbegeleiders gestuurd. Hiervan kreeg ik donderdag feedback. Voordat ik vrijdag het concept heb opgestuurd, heb ik deze feedback eerst verwerkt.

Week 14

Allereerst ben ik deze week bezig geweest met het probleem betreffende de rotatie. Hier ben ik een eind mee opgeschoten. Ik heb de weergavefunctie zo aangepast dat de knooppuntenlijst blijft zoals deze is, maar dat de verbindingslijst wordt aangepast. Allereerst wordt er gekeken wat het middelpunt van de knooppunten is. Op basis daarvan wordt bepaald wat de tangens-waardes zijn van de knooppunten t.o.v. het middelpunt. Deze tangens-lijst wordt dan gesorteerd van laag naar hoog. Met deze sortering wordt dan de verbindingslijst opgezet.

Dit werkt redelijk goed. De rotatie is nu mogelijk, en als een object wordt geroteerd, kan het goed zijn dat de vorm van de ROI zichtbaar wordt in het andere aanzicht. Dit maakt het roteren/schalen/verplaatsen van VOI's zeer bruikbaar.

Hiermee is ook het oorspronkelijk probleem opgelost met de weergave bij rotaties. Het enigste mankement wat er nu nog zit, is bij vrije vormen waarbij de tangens gelijk is, maar de afstand anders is. Punten met een korte afstand tot het middelpunt dienen namelijk later in de verbindingslijst te staan dan punten met een lange afstand. Op dit moment komen deze achter elkaar te staan. Dit levert een beeld met allerlei haken op.

Daarom moest er een andere manier worden gezocht voor de weergave van een VOI. Hierbij opperde mijn begeleider het idee om de knooppunt-volgorde van de ROI-plane te gebruiken. Binnen een ROI wordt namelijk een array met alle knooppunten opgeslagen. Deze volgorde is dezelfde als de volgorde van toevoegen. Als er informatie uit meerdere planes wordt gehaald, kan op basis van de array-index van de knooppunten worden bepaald welke punten achter elkaar moeten volgen.

In eerste instantie werkte dit niet. Dit kwam omdat de opvulling van lege ROI-plekken (door scaling) ervoor zorgde dat de volgorde van planes in de container niet klopte. Daardoor was het object in een ander aanzicht weer vertekend. Na het ordenen van de planes was dit wel mogelijk. Het enigste nadeel van deze methode was dat de volgorde zo is opgebouwd, dat er de vorm van een zandloper ontstaan in plaats van een vierkant. Dit komt omdat de twee evenwijdige lijnen op dezelfde y-coördinaat liggen. Na het tekenen van de eerste lijn, zal het knooppunt worden verbonden met het eerste knooppunt van de evenwijdige lijn. Hierdoor ontstaat de eerste diagonaal. De tweede diagonaal ontstaat omdat het laatste punt (onderaan de lijn) wordt verbonden met het eerste punt (bovenaan de evenwijdige lijn).

Dit zou opgelost kunnen worden door halverwege de connectielijst om te draaien, zodat er géén diagonalen worden getekend. Dit is niet mogelijk aangezien er niet vanuit mag worden gegaan dat een type aanzicht altijd hetzelfde blijft. Door rotaties in het niet-vrije-vorm aanzicht is het namelijk mogelijk dat het vrije-vorm aanzicht van het transversaal naar het sagittaal of coronaal aanzicht verplaatst (of een andere combinatie). Als in deze situatie de helft van de connectielijst wordt omgedraaid, zal er een hiaat ontstaan. Tevens zal de eindgebruiker het idee krijgen dat er een scheiding tussen de VOI zit, en dat deze in 2en is gesplitst.

Week 15

Deze week ben ik bezig geweest met de rotatie van de connectielijst. Zie week 14 waarom dit nodig was. Allereerst heb ik een optie geprobeerd waarbij er gezocht werd naar de kortste afstand tussen punten. Dit bleek helaas niet de juiste oplossing aangezien punten even ver van elkaar af kunnen liggen.

Een andere optie die ik heb geprobeerd is het gebruik van het middelpunt van de punten. Daarna heb ik op radiaal gesorteerd, zodat ze allemaal clockwise werden gekoppeld. Dit bleek ook niet goed te werken in situaties met vrije vormen. Hierbij kwam een punt eerder in aanmerking dan gepland, omdat deze over de as van de gemiddelde heen ging.

De laatste optie die ik heb geprobeerd is op basis van hoeken. De 2^e helft van het array dient omgedraaid te worden als het een vierkant betreft. Daarom ben ik bezig geweest om te kijken wat de hoek naar het volgende punt is, en wat de hoek naar het eindpunt is. De grootste hoek is namelijk altijd van toepassing aangezien de kleine hoek bij een vierkant altijd de diagonaal betreft. Bij het teken-plane ligt dat anders. Daar is dit in de meeste gevallen zo. In de gevallen waar dit niet als aanname gebruikt kan worden, kan nog de controle met de afstand worden uitgevoerd. Zodra de afstand het kortst is bij een grote hoek, dàt punt gekozen te worden.

Het enigste probleem wat nu nog optreedt is de situatie waarbij 3 lijnen in plaats van 2 lijnen beschikbaar zijn in een vierkant (zijaanzicht). Hierbij mag er niet op de helft gekeken worden om te draaien, aangezien het volgende punt dan de grootste hoek heeft.

Week 16

Vorige week (week 15) ben ik tussendoor bezig geweest met het Functioneel Ontwerp van iteratie 3. Dit heb ik gemaakt op basis van het FO van iteratie 2. Iteratie 3 was gericht op het weergeven van afbeeldingen in een 3D projectie. Deze moest de (eind)gebruiker kunnen roteren, zoomen, of verplaatsen naar zijn eigen voorkeur.

In het begin van deze week heb ik hiervoor het Technisch Ontwerp afgemaakt. Hierbij werd duidelijk dat er heel veel functionaliteit beschikbaar was in een klasse van de IDLex-set, en dat deze alleen geïmplementeerd hoefde te worden in het framework. Hiervoor had ik in week 13 al een klein prototype gemaakt. Dit prototype heb ik dan uitgewerkt in het functioneel ontwerp.

Aangezien mijn mening was dat dit toch niet veel werk zou zijn, heb ik dit ook geïmplementeerd deze week. Het is nu mogelijk om een VOI in een 2D aanzicht te tekenen/verplaatsen/roteren etc. en dat deze in het 3D aanzicht realtime wordt bijgewerkt.

Verder kwam ik er achter dat het niet moeilijk was om van de 3D weergave een filmpje te maken. Wat er in feite moest gebeuren was de afbeelding in kleine stappen draaien, en van iedere stap een afbeelding maken. Binnen IDL is een klasse beschikbaar waar foto's in gezet kunnen worden, en die hiervan een JPEG-filmpje van maakt. Door de afbeelding iedere keer 5 graden te draaien, en hiervan pixels te nemen, was het ook vrij eenvoudig om een filmpje te maken van het roteren.

Verder heb ik nog eens naar multithreading binnen IDL gekeken. Op internet stond een thread binnen de IDL nieuwsgroep dat dit gedaan kon worden door twee idl processen/applicaties te maken, en deze met elkaar te laten communiceren via sockets. Dit is niet een nette oplossing, maar kan het probleem van grote dicom-sets waarschijnlijk wel oplossen. Hier zal ik komende week overleg over plegen met mijn begeleiders.

Als laatste ben ik nog met de weergave van een VOI in 2D bezig geweest. Nu ben ik aan een ontwerp aan het denken waarbij per punt op de y-as gekeken of er punten beschikbaar zijn voor dat betreffende plane. Hiervan kunnen de buitenste x-waardes opgehaald worden, waardoor er geen gedoe meer ontstaat met hoeken en dergelijke. Volgende week probeer ik dit te implementeren. De pseudo-code hiervoor is wel al geschreven.

Week 17

Deze week ben ik vooral bezig geweest met het afstudeerverslag. Voornamelijk op het gebied van taalgebruik en zinsbouw. Verder was het een korte week aangezien de pinksteren hierin viel, en ik op donderdag naar Amsterdam ben geweest voor een gesprek en uitleg over de master die ik na deze opleiding wil gaan volgen.

Week 18 t/m 20

Aangezien dit stageverslag voor het einde van de stageperiode ingeleverd moet worden, zijn er nog 3 weken die ontbreken. In deze weken zal er onderzocht worden of het mogelijk is om via een aantal andere manieren thread-functionaliteit te creëren, en zal er gestart worden met een applicatie waarin het uitgebreide framework wordt geïmplementeerd.