
BSc-project IN3405



Eindverslag

Projectgroep:

Abdoelrasheed Jarmohamed
Kishen Gajadhar

Commissie:

Job Scheffers (Zorgverzekeraar DSW, begeleider)
B.R. Sodoyer (TU Delft, coördinator / begeleider)

Omgeving:

Zorgverzekeraar DSW
Technische Universiteit Delft
Faculteit EWI Technische Informatica
Variant Softwaretechnologie

Datum: 2 oktober 2008

Status: wachten op acceptatie

Versie: 1.0

©Oktober 2008 DSW Zorgverzekeraar

Alle in dit verslag afgebeelde gegevens, waaronder teksten, foto's, illustraties, grafische materiaal, (handels)namen, logo's, waren- en dienstmerken, zijn in eigendom van zorgverzekeraar DSW en worden beschermd door auteursrecht, merkenrecht en/of enig ander intellectueel eigendomsrecht.

Voorwoord

Voor het behalen van het Bachelordiploma van de opleiding Technische Informatica aan de Technische Universiteit Delft, is het Bachelorproject een belangrijk en verplicht onderdeel. Wij hebben het Bachelorproject gedaan in samenwerking met DSW Zorgverzekeraar in Schiedam. Gedurende drie maanden hebben wij op het kantoor van DSW gewerkt aan het project. Een van de eindproducten is dit eindverslag, waarin wordt beschreven van wat wij precies allemaal hebben gedaan.

Het project bestond uit het ontwikkelen van een image viewer control, een integreerbare tool voor het bekijken van een bitmap op verschillende manieren. De gebruiker moet een willekeurige bitmap op het scherm kunnen vergroten, verkleinen, roteren, inverteren enz. Na het ontwikkelen van deze tool, kan het worden geïntegreerd in de applicaties van DSW bij welke dat nodig zijn.

Doordat er maar twee projectleden zijn, is de samenwerking en communicatie gemakkelijk verlopen. Ook de werksfeer binnen DSW is goed en prettig om in te werken. Dit allemaal heeft bijgedragen dat dit project in zijn geheel een leuke ervaring is geweest. Ook door het werken op het kantoor in plaats van op de faculteit, hebben we een beter beeld gekregen hoe de ICT-wereld in elkaar steekt.

Onze dank gaan uit naar Samira El Haddioui en Job Scheffers, die ons de mogelijkheid hebben gegeven om het Bachelorproject bij DSW Zorgverzekeraar te doen. Ook gaat onze dank uit naar Meneer Scheffers voor de begeleiding op het kantoor en Meneer Sodoyer voor de begeleiding vanaf de TU. Verder willen wij een ieder bedanken die op welke wijze dan ook ons geholpen heeft tijdens dit project.

Schiedam, 30 september 2008

Kishen Gajadhar
Rashied Jarmohamed

Samenvatting

De opdracht voor het project komt van DSW Zorgverzekeraar. Door de vele formulieren, brieven en rekeningen die de DSW ontvangt en verstuurt, worden al deze documenten gescand en opslagen in de een database. De medewerkers van DSW gebruiken een losstaande image viewer om al deze documenten te bekijken wanneer dat nodig is. DSW, dat een eigen ICT-afdeling, heeft wil nu voor de toekomst beschikken over een verbeterde versie van deze image viewer. En deze nieuwe image viewer moet geen applicatie zijn, maar een control (plugin) die kan worden hergebruikt. De image viewer control kan dan worden ingebouwd in de verschillende systemen van DSW waar dat nodig is. Dit zou kunnen gebeuren in een invoersysteem voor nieuwe klanten. Voor de medewerkers is het dan niet meer nodig om twee verschillende applicaties te openen.

Analyse

De oorspronkelijke opdracht van de image viewer control was in drieën verdeeld. Een van de redenen van het opsplitsen is om het ontwikkelen zo overzichtelijk mogelijk te houden en de functionaliteiten van elkaar te scheiden. Na een aantal gesprekken met de opdrachtgevers en eindgebruikers zijn er meerdere malen een aantal wijzigingen in de opdracht gekomen. De veranderingen waren nodig, omdat in het begin van het project nog een aantal feiten onduidelijk waren bij de werkgever.

Uiteindelijk is de control in vieren verdeeld: *Viewer Control*, *Navigatie Control*, *Converter Control* en de *Search Control*.

Met de *Viewer Control* kan de weergave van de getoonde image worden veranderd. De image kan op verschillende manieren worden gezoomd, geroteerd en geïnverteerd. Ook kan de image op vier verschillende fitmodes worden weergegeven: heightmode, widthmode, fullmode en de nonemode.

Met de *Navigatie Control* kan er worden genavigeerd in een bepaalde structuur tussen de verschillende images waaruit een document kan bestaan.

De *Converter Control* zorgt ervoor dat hij de documenten, die hij haalt uit de database, in de juiste structuur zet. En omdat een document niet alleen uit images bestaat, maar ook als bijvoorbeeld een Word-bestand, zal deze control ervoor zorgen dat de juiste programma's extern worden gestart.

Het zoeken van documenten in de database wordt mogelijk gemaakt door de *Search Control*.

Samen met de opdrachtgever is afgesproken om alle controls afzonderlijk van elkaar te analyseren, ontwerpen en implementeren. Dit omdat de grootste prioriteit en de moeilijkste control toch de *Viewer Control* is.

Tijdens de analyse kwamen we erachter dat de ontwikkeling van de *Viewer Control* toch anders verloopt dan menig ander opdracht. De use cases kunnen erg kort zijn, doordat de eindgebruiker maar één knop te drukken in het control om de gewilde resultaat te krijgen. Wij hebben ervoor gekozen om de verschillende mogelijkheden van de *Viewer Control* in één use case te zetten.

De analyse van de *Navigatie Control* is veel eenvoudiger. Het moet ervoor zorgen dat er kan worden genavigeerd tussen de images. De images zitten in een bepaalde structuur. Ze bevinden zich in een attachment. En attachments zitten weer in documenten. Dus als er een aantal documenten wordt gegeven aan de *Navigatie Control*, kan deze navigeren tussen images, attachments en documenten.

Wat de *Converter Control* moet kunnen is een gegeven document uit de database halen, deze in de gewenste structuur zetten en bepaalde attachments extern laten openen.

Aan de *Search Control* zijn we niet aan toe gekomen en hebben we dus ook niet geanalyseerd.

Ontwerp

Het ontwerp van de *Viewer Control* is niet erg ingewikkeld. Het zijn meer de algoritmes die de opdracht erg ingewikkeld maken. Deze user interface bestaat uit een zelfgeschreven picturebox klasse. Hierin bevindt zich de image. Elke keer als de picturebox de opdracht krijgt om de weergave te veranderen van de image, tekent hij zichzelf opnieuw met de gevraagde weergave van de image.

Voor het opnieuw tekenen wordt de nieuwe zoomfactor en nieuwe positie van de image doorgegeven aan de picturebox.

Het ontwerp van de *Navigatie Control* bestaat uit drie klassen. Een attachment klasse, een document klasse en de user interface. In de attachment-klasse zit een array waarin de images zich bevinden. In de document-klasse zit ook een array, maar dan met attachments. In de user interface klasse kan er worden genavigeerd in de structuur van images. De user interface zal bestaan uit de Viewer Control die is toegevoegd en een toolbar met de navigeerbuttons. Aan de hand van de positie van de getoonde image in het structuur zullen een aantal knoppen worden gedisable of onzichtbaar worden gemaakt. Als bijvoorbeeld de laatste image van een document wordt getoond, zal de last image button worden gedisable.

De *Converter Control* bestaat maar uit één klasse. Voor het ophalen van documenten uit de database en extern openen van programma's zal het gebruik maken van een ActiveX control die al gemaakt was voor het gebruik van de oude image viewer.

Implementatie

De implementatie van alle controls is gedaan met behulp van Visual Studio. Het ontwerp van de *Viewer Control* ziet er klein en eenvoudig uit. De implementatiefase daarvoor is daarentegen wel met enige moeite verlopen. De moeilijkheid daarvan zat hem in het maken van de algoritmes voor de verschillende functies. Ook moesten de algoritmes efficiënt zijn, omdat het veranderen van de weergave naar de images zonder flikkering moet gebeuren. Voor een aantal functies waren er verschillende algoritmes geïmplementeerd. Het algoritme dat dan het efficiëntste was, is gekozen voor Viewer Control.

Om te navigeren tussen de images van de documenten in de *Navigatie Control*, is gebruik gemaakt van arrays. De hele structuur van de images is opgebouwd uit arrays. De attachment-klasse heeft een array van images en zo heeft de document-klasse een array van attachments. Met behulp van de properties kunnen de inhoud van de arrays worden opgevraagd.

De implementatie van de *Converter Control* is niet in zijn geheel afgerond. Wat wel is geïmplementeerd is het ophalen van documenten uit de database.

Testen

Elk van de control werd zowel handmatig en automatisch getest. Automatische tests werden in een door visual studio gegenereerde test project gedaan waar voor de handmatige tests voor elk van de control een test applicatie werd gemaakt.

Conclusie

Uiteindelijk werden twee van de vier controls volledig afgerond. Met de derde is een start gemaakt. De implementatie was al begonnen. Omdat voor dit project een compleet werkende *viewer control* als enige **must** werd gezien, kan dit project als succes gezien worden.

Inhoudsopgave

Voorwoord	ii
Samenvatting.....	iii
Inhoudsopgave	v
1 Inleiding.....	6
2 Planning & Organisatie	7
2.1 Communicatie	7
2.2 Documentatie	7
3 Bedrijfsprofiel	8
3.1 Rol van ICT binnen DSW.....	8
3.2 Software Ontwikkeling binnen DSW.....	9
4 Huidige en Toekomstige Situatie	10
5 Het Project	14
5.1 Controls	14
5.2 Ontwikkelfases	15
6 Analyse	17
6.1 Viewer Control.....	17
6.2 Navigatie Control.....	19
6.3 Converter Control	21
7 Ontwerp.....	24
7.1 User Interface	24
7.2 Systeem ontwerp.....	26
7.3 Testen.....	29
8 Implementatie	30
8.1 User Interface	30
8.2 Viewer Control.....	33
8.3 Navigatie Control.....	35
8.4 Converter Control	40
8.5 Tests.....	41
9 Conclusie en Reflecties.....	43
9.1 Conclusies.....	43
9.2 Reflectie en verantwoording	43
Referenties.....	45
Bijlage A: Opdrachts omschrijving	46
Bijlage B: Research verslag	47
Bijlage C: Plan van Aanpak	59
Bijlage D: RAD viewer control.....	75
Bijlage E: RAD navigatie control.....	96
Bijlage F: RAD converter control	106
Bijlage G: SDD viewer control	116
Bijlage H: SDD navigatie control	134
Bijlage I: SDD converter control.....	141
Bijlage J: Acceptatie & Test plan.....	146

1 Inleiding

Dit is het eindverslag van het bachelorproject die we hebben afgerond voor de opleiding Technische Informatica aan de TU Delft. De opdracht voor het project komt niet van de TU, maar van een bedrijf, namelijk DSW Zorgverzekeraar. Een zorgverzekeraar heeft erg veel administratie. Dit allemaal wordt natuurlijk gedaan met verschillende informatiesystemen. Voor het ontwikkelen van deze systemen heeft DSW een eigen ICT-afdeling. Een van de opdrachten voor ICT-afdeling die al langer op planning stond was de ontwikkeling van een image viewer control.

De image viewer control moet een integreerbare tool zijn voor het bekijken van een image op verschillende manieren. De eindgebruiker moet een willekeurig image op het scherm kunnen vergroten, verkleinen, roteren, inverteren enz. Na het ontwikkelen van deze tool, kan het worden geïntegreerd in de applicaties van DSW bij welke dat nodig zijn. Dit zal dan gedaan worden door de medewerkers van de ICT-afdeling.

Bij de oplevering van de image viewer control en de presentatie, wordt ook dit eindverslag afgegeven. Dit verslag kan namelijk ook dienen als referentie voor mogelijk toekomstig onderhoud en/of aanpassingen aan de control door de medewerker van de ICT-afdeling. Elk van de gemaakte keuzes wordt daarom binnen dit eindverslag beschreven. Alle opgeleverde documentatie van de ontwikkelfases bevindt zich in dit eindverslag, zodat het eventueel in de toekomst gebruikt kan worden.

Hoe de organisatie van het project was staat in hoofdstuk 2. In hoofdstuk 3 wordt wat meer over DSW beschreven. Ook wordt er kort beschreven op welke manier de softwareontwikkeling tot stand komt. Waarom DSW een image viewer control wil laten ontwikkelen staat in hoofdstuk 4. Dit wordt uitgelegd door middel van een beschrijving van de huidige situatie en toekomstige situatie. In hoofdstuk 5 wordt uitgelegd hoe de opdracht is verdeeld en welke verantwoordelijkheden elk deel heeft. De analyse van de opdracht staat in hoofdstuk 6. In hoofdstuk 7 bevindt zich het ontwerp. Een implementatie verslag staat in hoofdstuk 8.

2 Planning & Organisatie

Op de eerste dag van onze stage hebben we een introducerend gesprek gehad over de te ontwikkelen software met onze begeleider en opdrachtgever, Job Scheffers. Na het gesprek hebben we direct een uitgebreid plan van aanpak gemaakt. Daarna zijn we begonnen met de twee weken verplichte vooronderzoek. Na het afronden van het verplichte onderzoek zijn we begonnen met het de eigenlijke opdracht in week 3. In deze week hebben we gesprekken gehad met de eindgebruikers en de opdrachtgever. Na deze gesprekken hebben we samen met de opdrachtgever enkele wijzigingen aangebracht in de oorspronkelijke opdracht. De belangrijke punten van deze gesprekken zijn beschreven in de bijlage van het de RAD van de *viewer control*.

We hebben toen ook besloten om de opdracht in vier aparte delen te splitsen. Zodat elk deel apart wordt geanalyseerd, ontworpen en geïmplementeerd. Dit om de gehele opdracht wat overzichtelijker te maken. Ook is dit besloten omdat de werkgever de eerste en belangrijkste deel van de opdracht geheel afgerond wil hebben. Er zou tijd worden verloren als de gehele opdracht van te voren zou worden geanalyseerd en ontworpen, terwijl de belangrijkste deel niet geheel wordt geïmplementeerd door tijdgebrek. Door de verdeling van de opdracht is het plan van aanpak, die we in de eerste week van de stage hebben opgesteld, voor een deel komen te vervallen. Het plan van aanpak is toegevoegd aan dit document als bijlage. Over de onderverdeling van de opdracht en de details hierover staat meer in hoofdstuk 5.

2.1 Communicatie

De gehele stage hebben wij afgewerkt op kantoor van DSW. Wat namelijk ook een eis was van DSW. Met wie wij binnen DSW hebben gecommuniceerd, is vooral de opdrachtgever Job Scheffers. De communicatie gebeurde dus altijd face to face op kantoor. Als iets voor ons niet duidelijk was of als we een aantal suggesties hadden, konden we altijd even stappen naar de heer Scheffers. Maar als er bijvoorbeeld iets over het functioneren van de image viewer control uitgebreid moest worden besproken met de heer Scheffers of andere medewerkers, werd er een afspraak gemaakt. We trokken ons dan terug in een vergaderruimte in het DSW gebouw.

Omdat de projectgroep maar uit twee leden bestond en we samen altijd aanwezig waren op kantoor, is de communicatie onderling gemakkelijk verlopen. Het was dus niet nodig om speciaal een afspraak te maken om iets te bespreken.

En als één van de twee verhinderd werd door ziekte of wat dan ook, kon de andere altijd verder werken, omdat alle documenten centraal op een harde schijf stonden. Op DSW is er namelijk een netwerk harde schijf. Alle documenten en code die werden ontwikkeld, werden ook geplaatst op de netwerk harde schijf.

2.2 Documentatie

Het inleveren van een aantal documenten heeft de TU verplicht gesteld. Voordat er wordt begonnen aan de eigenlijke stageopdracht, moet er twee weken vooronderzoek worden gedaan. Het eerste document dat dus ingeleverd werd, was het onderzoeksverslag. Dit is als bijlage te vinden in dit document.

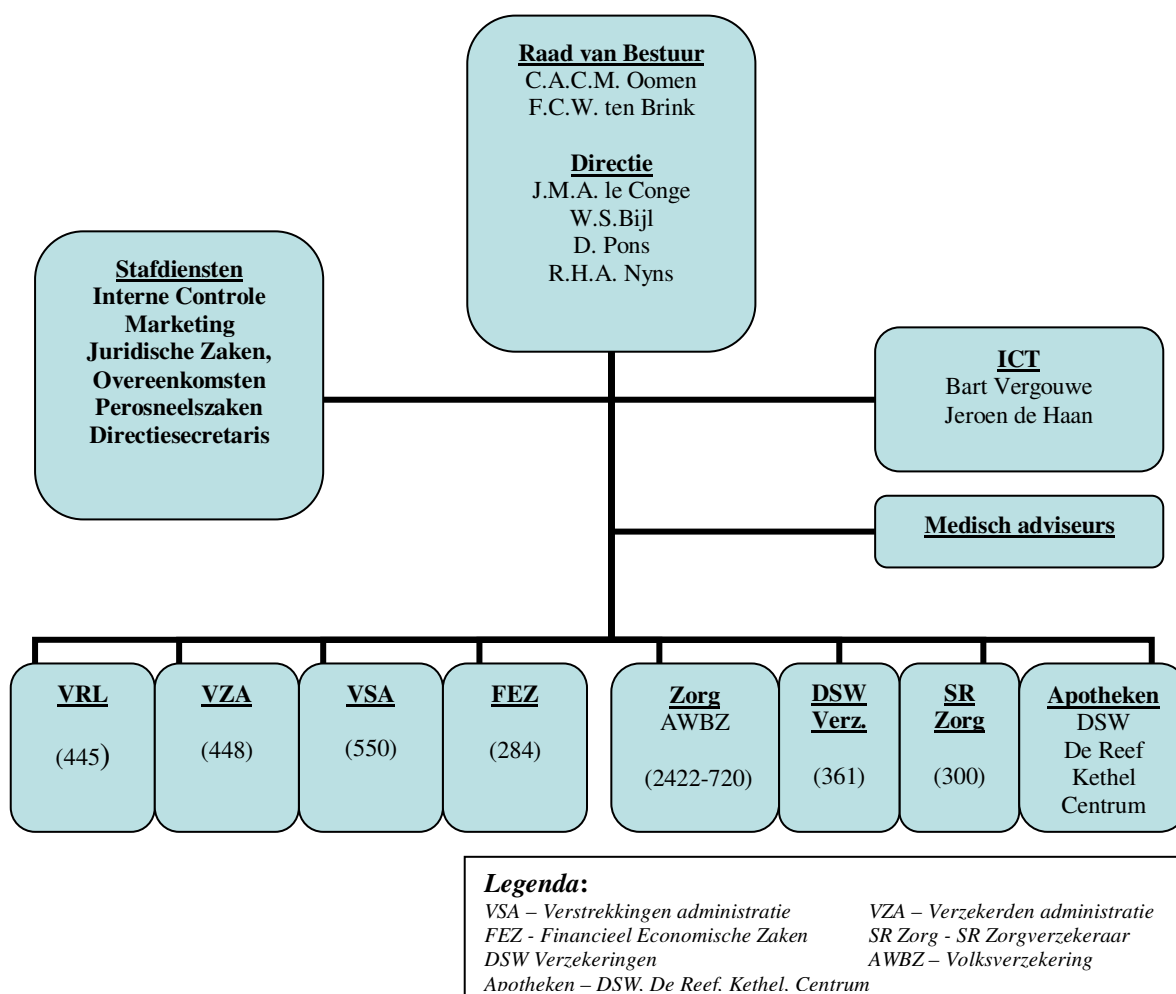
Bij het beginnen aan de stageopdracht er een plan van aanpak opgesteld. Ook dit is als bijlage bijgevoegd aan dit document. Nadat er was afgesproken dat de gehele Image Viewer Control in vieren is gedeeld, is ook afgesproken dat deze allemaal afzonderlijk worden geanalyseerd, ontworpen, geanalyseerd en getest. Dus voor elke deel die is afgerond is er een verslag voor elke fase. Al deze documenten zitten bijgevoegd in dit eindverslag. En tot slot is natuurlijk dit verslag tot stand gekomen, het eindverslag.

3 Bedrijfsprofiel

Sinds januari 2006 zijn de mensen in Nederland wegens de Ziekenfondswet verplicht een ziekenfondsverzekering af te sluiten. Zorgverzekeraar DSW voert als gevolg van deze wet de verplichte ziekenfondsverzekering uit en is marktleider in de regio Delfland Schieland en Westland. Deze zorgverzekeraar heeft ruim 330.000 verzekerden.

Omdat DSW vindt dat iedereen recht heeft op betaalbare zorg van hoge kwaliteit, is zij betrokken bij een groot aantal zorginhoudelijke projecten. Het voornaamste doel van DSW is het bevorderen van een optimale gezondheidszorg voor haar verzekerden.

Binnen DSW zijn er verschillende afdelingen waaronder 'VZA: Verzekerden Administratie' welke direct gebruik zal gaan maken van het te bouwen control. Een overzicht van de organisatie wordt gegeven in *figuur 1*¹.



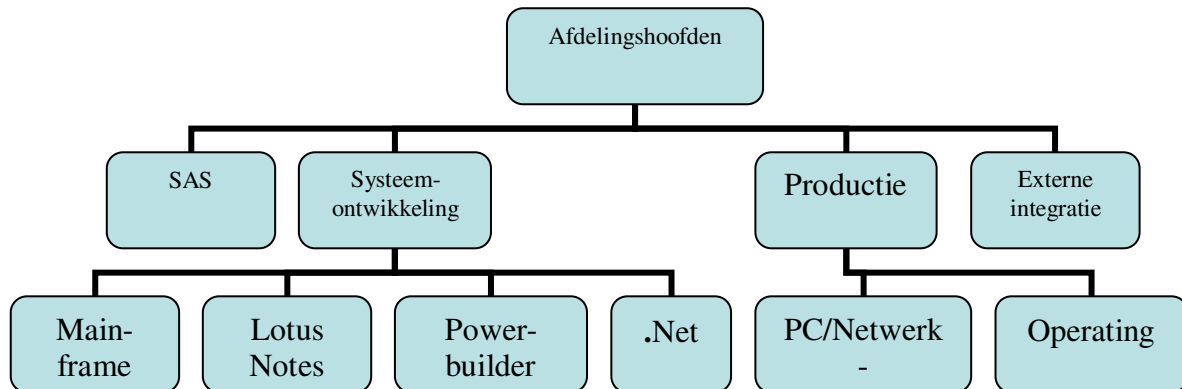
Figuur 1: Organisatie DSW

3.1 Rol van ICT binnen DSW

Om zorg van hoge kwaliteit te kunnen bieden maakt DSW gebruik van allerlei mogelijke middelen op het gebied van automatisering. Hierbij worden de nieuwste technieken gebruikt.

Administratieve verwerkingen worden bijna volledig met de computer gedaan binnen DSW en de afdeling ICT zorgt ervoor dat al deze systemen beschikbaar zijn. Bij het ontwikkelen van systemen

wordt er rekening gehouden met de wensen en kennis van gebruikers. Ontwikkeling vindt hierbij ‘in huis’ plaats door de eigen ontwikkelaars voor de verschillende platformen. Systeem ontwikkeling vindt plaats voor/in Mainframe, Lotus Notes, Powerbuilder en .Net. Dit kan ook gezien worden in *figuur 2* waarin de hiërarchie binnen de afdeling ICT wordt weergegeven¹.



Figuur 2: Organisatie van de ICT-afdeling

3.2 Software Ontwikkeling binnen DSW

Bij het maken van software, wordt binnen DSW *Rational Unified Process-RUP* als methodiek gebruikt. RUP is grofweg een iteratief software ontwikkelingsproces. Door prototypering en terugkoppeling worden requirements voortdurend aangepast. Iteratief ontwikkelen binnen het project d.m.v deelproducten zorgt ervoor dat eventueel veranderende wensen van een klant niet het hele product beïnvloeden maar per deelproduct aangepast kan worden.

RUP is gebaseerd op de volgende principes:

- *Ontwikkel software iteratief (requirements, prototype, feedback & terugkoppeling)*
- *Maak gebruik van component gebaseerde architectuur*
- *Test het systeem*
- *Maak gebruik van versiebeheer tijdens de software ontwikkeling*

DSW schrijft niet strikt voor dat RUP gebruikt moet worden. Er kan ook gekozen worden voor een andere methodiek.

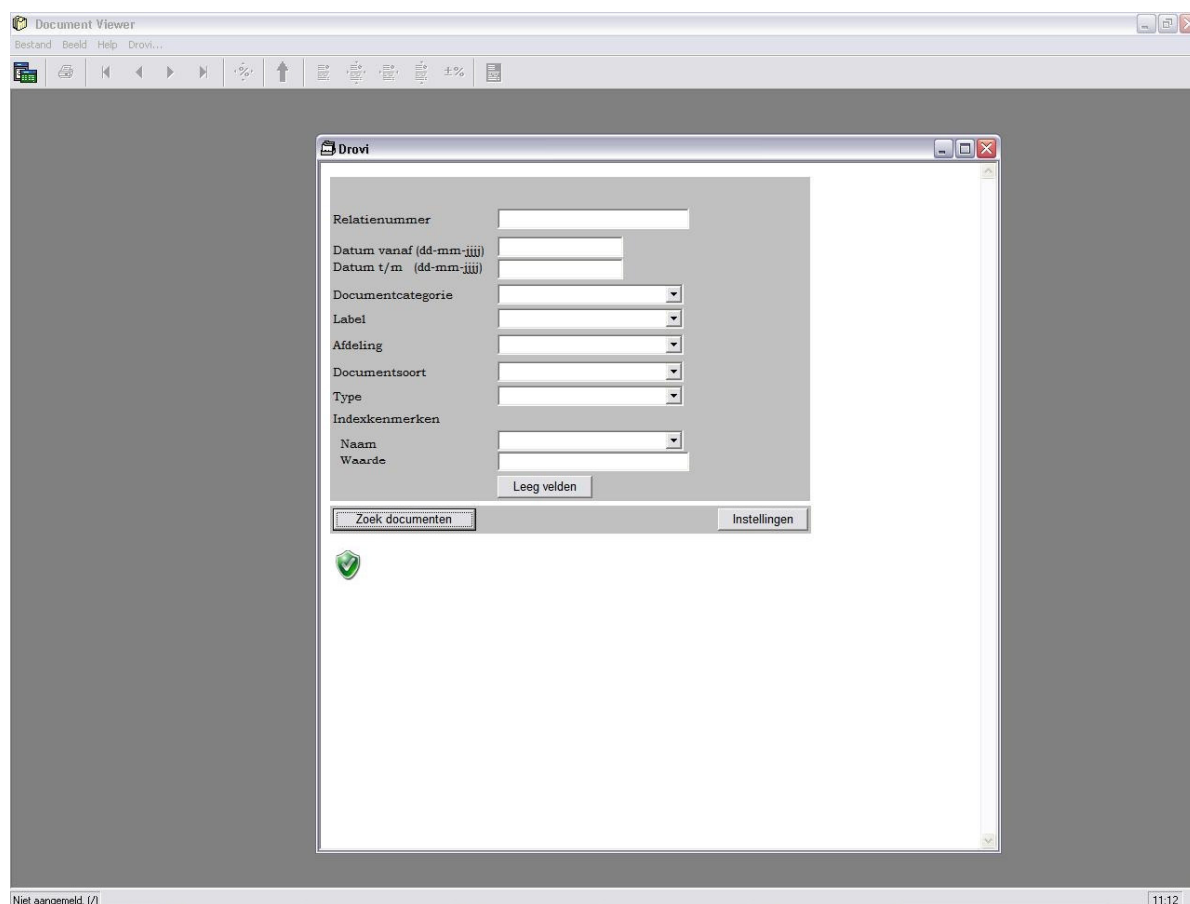
4 Huidige en Toekomstige Situatie

Zoals u hebt gelezen in hoofdstuk 3 heeft DSW een eigen ICT-afdeling waar ze alle systemen ontwikkelen en onderhouden voor de verschillende afdelingen van DSW. De software in dit project zal worden ontwikkeld voor de Verzekerden-Administratie afdeling (VZA) en de Verstrekking-Administratie afdeling (VSA). Deze afdelingen versturen veel post en krijgen ook veel post binnen. Dit zijn meestal brieven van en naar klanten of ziekenhuizen. Vaak zijn dit ook rekeningen van klanten of ziekenhuizen. Al deze documenten moeten door verschillende medewerkers worden bekeken en worden allemaal digitaal opgeslagen, zodat iedere medewerker op de afdelingen alle documenten kan opvragen en inzien. De documenten worden dus gescand en in een database opgeslagen.

Huidige Situatie

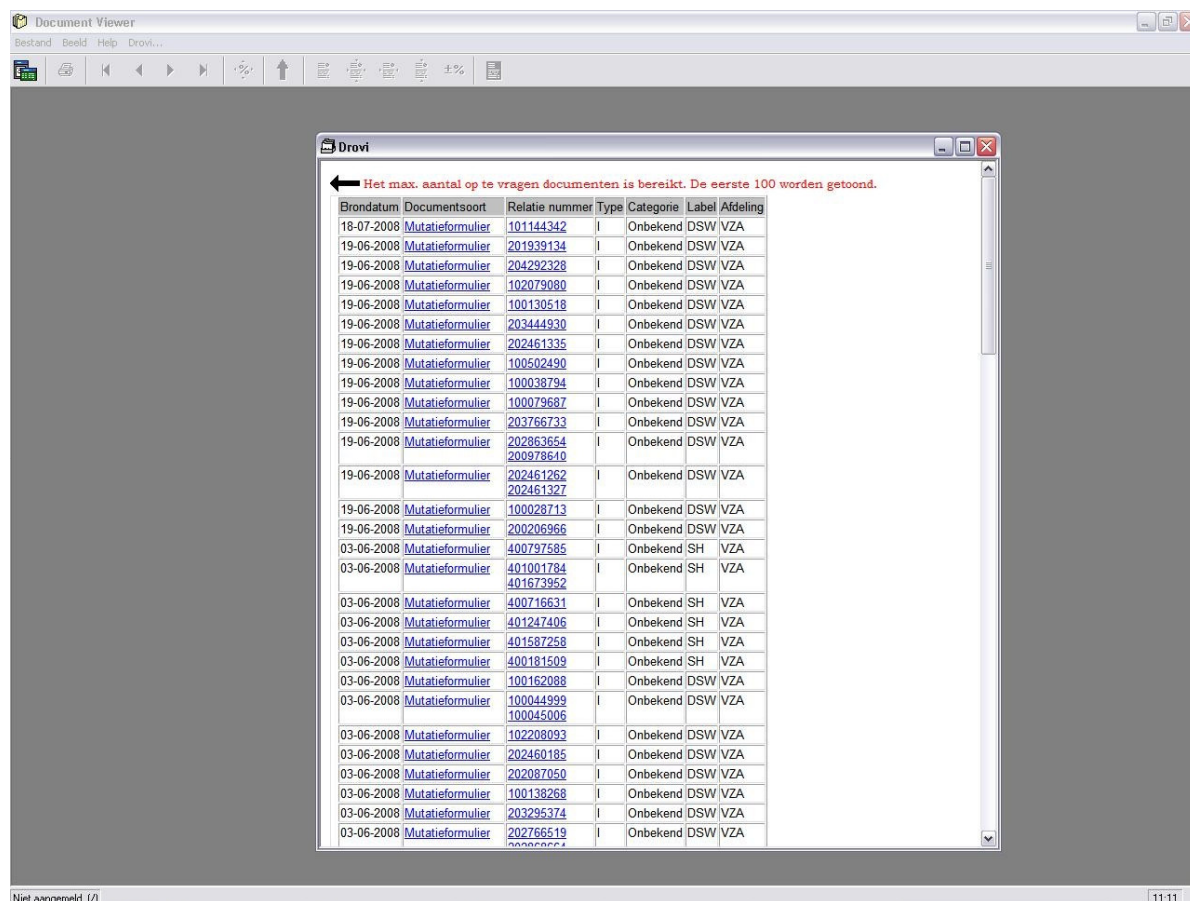
Voor het bekijken van de documenten is een aantal jaren geleden een losstaande image viewer ontwikkeld met de gewenste mogelijkheden. In het menu van de image viewer zit een link naar de "opvraagstelsysteem" voor de documenten. Dit opvraagstelsysteem heet "Drovi".

Als er op de link wordt gedrukt komt er een zoekscherm (versimpelde versie van Internet Explorer) tevoorschijn waarmee met de benodigde zoekcriteria kan gezocht worden naar een bepaald document of een selectie van documenten. Een afbeelding van de huidige image viewer is weergegeven in de onderstaande *figuur 3*.



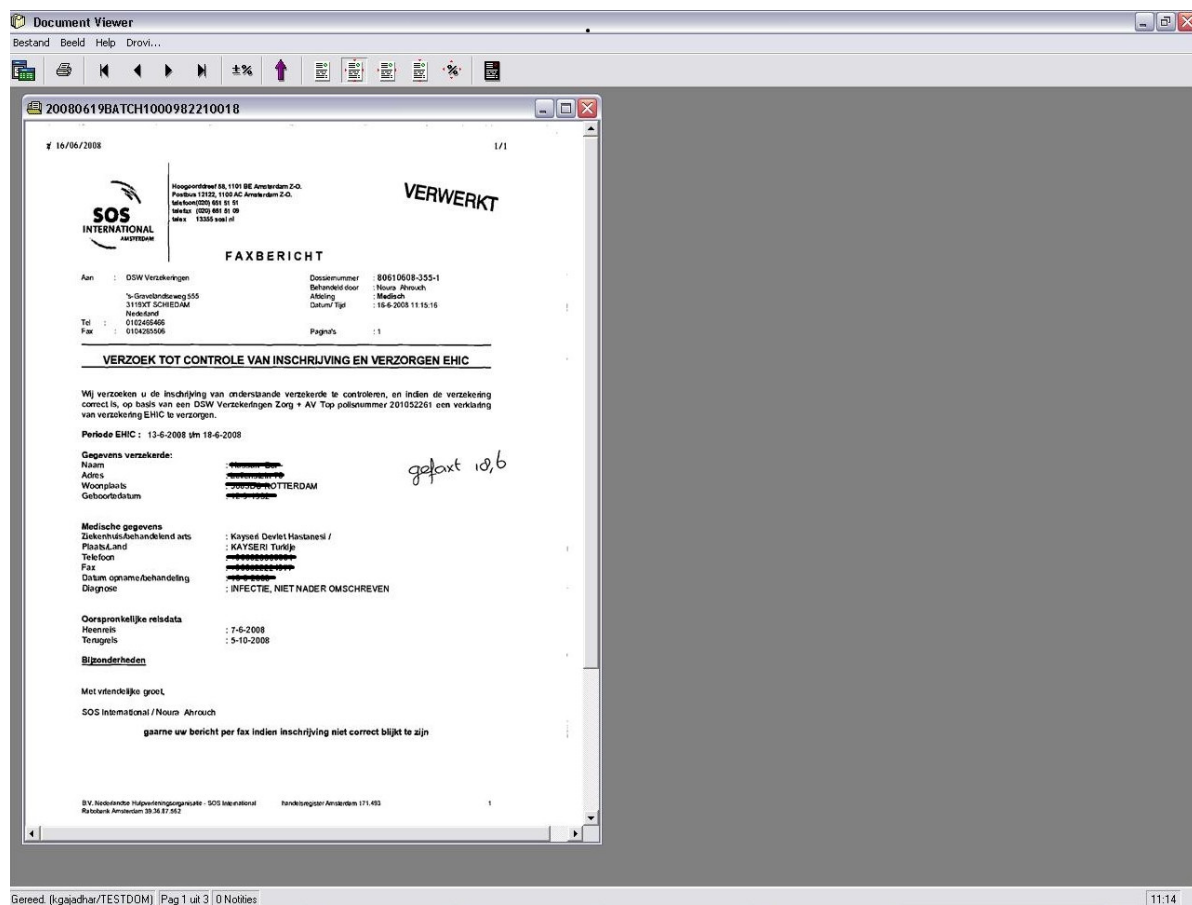
Figuur 3: Huidige losstaande image viewer met het Drovi-zoekscherm

In figuur 3 is ook het zoekscherm van Drovi te zien. Als de criteria zijn ingevoerd in de verschillende velden van het zoekscherm en daarna op de zoekknop wordt gedrukt, stuurt het zoekscherm het verzoek door naar een IIS (Microsoft Internet Information Services of Server). De IIS zorgt ervoor dat er contact wordt gemaakt met de *Drovi-database*. Dit is een database met alle indices en id-nummers van alle documenten die zijn opgeslagen in een andere database genaamd: *InfoImage*. De indices en id-nummers worden dan gestuurd naar het zoekscherm. Dit zoekscherm laat dan een lijst van de gevraagde documenten zien. Deze lijst is te zien in *figuur 4*.



Figuur 4: Huidige losstaande image viewer met een lijst van documenten

Als op een document wordt geklikt, zal de index van het document door worden gegeven naar de image viewer. De image viewer maakt contact met InfoImage en zoekt dan met behulp van de index naar het document en laat het vervolgens zien. In *figuur 5* is te zien hoe een document wordt weergegeven in de huidige image viewer.

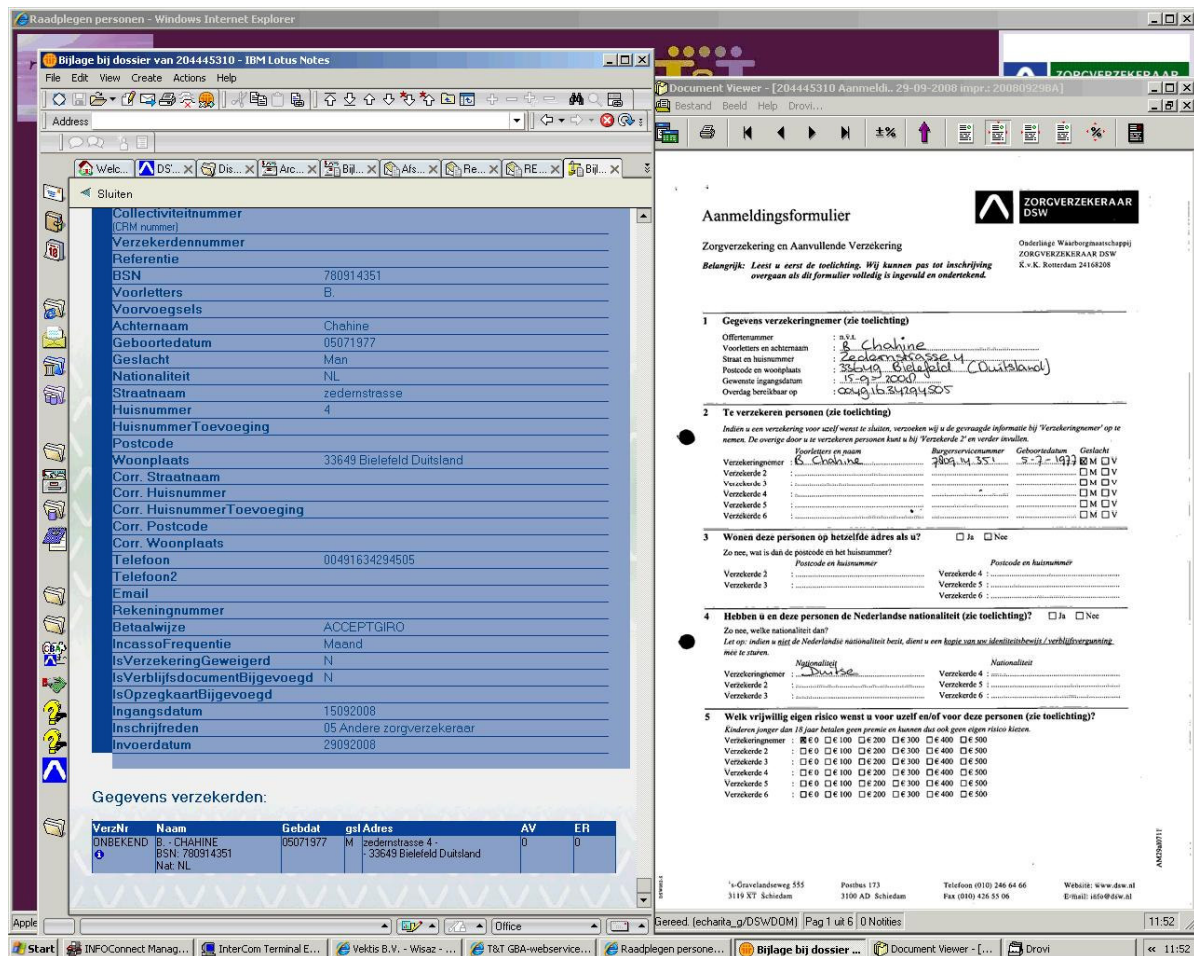


Figuur 5: Huidige losstaande image viewer met een document dat wordt weergegeven

Als een medewerker bijvoorbeeld informatie uit een document wil invoeren in het huidige “invoersysteem” van DSW, opent hij naast het invoersysteem ook nog de image viewer. Daarna zal hij Drove moeten openen, vervolgens zoekt hij de documenten en als laatste wordt het gewilde document getoond in de image viewer. De informatie in de documenten die de medewerker nodig heeft zal hij dan kunnen invoeren in het invoersysteem.

De medewerkers van DSW werken met vele verschillende applicaties. Er zijn applicaties om bijvoorbeeld nieuwe klanten in te voeren, applicaties om het dossier van een klant te beheren en applicaties om de gescande documenten in te voeren in de database. Bij veel van de applicaties hebben de medewerkers informatie nodig uit de documenten. Naast de schermen van het systeem waarmee ze informatie kunnen bewerken, moeten ze ook vaak de image viewer openen om de documenten te bekijken. De medewerkers moeten altijd hun beeldscherm zo indelen dat alles overzichtelijk blijft en de documenten leesbaar zijn in de image viewer. Dit is vaak onhandig, tijdrovend en dus inefficiënt.

In *figuur 6* is een printscreen weergegeven van één van de medewerkers die werkt met de image viewer. Aan de linkerkant wordt het systeem weergegeven om de gegevens van de klanten in te voeren. Aan de rechterkant is de image viewer weergegeven met een aanmeldingsformulier van een klant. Hier is te zien dat met een losstaande image viewer lastiger werken is.



Figuur 6: Gebruik huidige image viewer

Toekomstige Situatie

DSW had al een tijdje besloten dat het in de nabije toekomst af wil van de losstaande image viewer. Het wil nu een image viewer die ingebouwd kan worden in de bestaande schermen van elk systeem waarbij het nodig is om informatie uit de documenten te halen. Hierdoor zullen de medewerkers geen tijd meer verliezen met het telkens opnieuw indelen van hun beeldscherm.

5 Het Project

Het doel van het project is om een usercontrol te ontwikkelen met minimaal de zelfde functionaliteiten als de huidige image viewer. De systemen die ontwikkeld worden voor DSW, worden bijna allemaal ontwikkeld met de programmeertaal C# van het .NET-omgeving. De image viewer control zal dan ook in C# geschreven moeten worden, om het integreren van de control mogelijk en makkelijk te maken.

5.1 Controls

De oorspronkelijke opdracht van de image viewer control was in drieën verdeeld. Een van de redenen van het opsplitsen is om het ontwikkelen zo overzichtelijk mogelijk te houden en de functionaliteiten van elkaar te scheiden. Na een aantal gesprekken met de opdrachtgevers en eindgebruikers zijn er meerdere malen een aantal wijzigingen in de opdracht gekomen.

De reden waarom de opdracht een aantal keer is veranderd, is omdat er bij de opdrachtgever nog een aantal feiten over de oude image viewer nog onduidelijk waren. Hier wordt beschreven wat voor veranderingen allemaal hebben plaatsgevonden tijdens het hele project.

Eerste Verandering

Na dat we de opdrachtbeschrijving hadden gekregen, hebben we gesprekken gevoerd met de eindgebruikers en de opdrachtgever. Tijdens deze eerste gesprekken is de opdracht op een aantal punten gewijzigd en is de Image Viewer Control in vieren gedeeld. Vier verschillende controls met elk zijn verantwoordelijkheid. In het RAD in bijlage C zijn deze gesprekken genotuleerd en is te zien hoe de eerste veranderingen in de opbouw van de controls tot stand zijn gekomen. Ook is in de RAD te zien wat de oorspronkelijke opdracht was.

Elke deel van de Image Viewer Control is een aparte control zijn met zijn eigen verantwoordelijkheid. Samen zullen deze vier controls de Image Viewer Control vormen. Elk control, los van de eerste, is gebaseerd op de vorige, waardoor bij voltooiing van een control een belangrijke functionaliteit toegevoegd wordt.

De *eerste control* zal de eigenlijke image viewer zijn met al zijn basisfuncties. Hierin zal een image kunnen worden bekeken op verschillende manieren. Hierin kan een bitmap vergroot, verkleind, geroteerd en geïnverteerd worden.

Een document kan bestaan uit één of meer images. Het kunnen “scrollen” binnen een document naar een andere bitmap zal de *tweede control* mogelijk moeten maken. De image die moet worden weergegeven zal dan aan de eerste control worden gegeven.

Als een document of een selectie van documenten zal worden opgevraagd zal dat door Drovi worden gegeven in een map. Het kunnen selecteren van een document in een map zal gebeuren door de *derde control*. Het gevraagde document zal dan worden doorgegeven aan de tweede control.

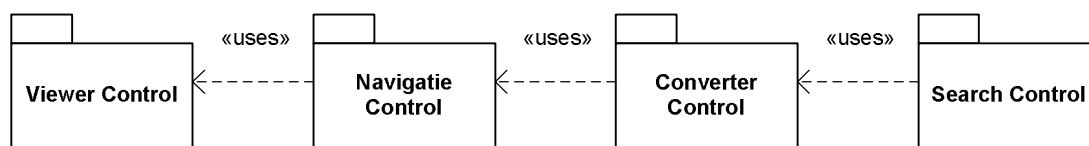
De *vierde control* zal het Drovi-zoekscherm vervangen. Hierin zal één of meer documenten kunnen worden opgezocht met de verschillende zoekcriteria. De gevraagde documenten worden dan in een map gezet die wordt gebruikt in de derde control. Zo zie je dat elke control een service biedt aan de onderliggende control.

Tweede Verandering

Nadat de eerste control bijna helemaal was afgerond, zijn er weer twee gesprekken gevoerd met de opdrachtgever. Hierin werd besproken wat precies de verantwoordelijkheden van de tweede en derde control moesten zijn. En doordat enkele zaken wat duidelijker waren geworden voor de opdrachtgever over de oude image viewer, hebben de tweede en derde control weer andere verantwoordelijkheden toe gekregen. De vierde control is het zelfde gebleven.

De eerste control, de Viewer Control, was al bijna afgerond. Daarvan is niets veranderd. De tweede control is nu de Navigatie Control. De verantwoordelijkheid van deze control is de navigatie tussen de verschillende images, attachments en documenten. Een document bestaat uit verschillende attachments en attachments bestaan weer uit verschillende images. In de RAD van de Navigatie Control staat meer uitgelegd hierover. Als aan deze control een aantal documenten wordt gegeven, kan het door alle images heen navigeren. Eigenlijk is de Navigatie Control de tweede en derde control die bij de eerste verandering zijn beschreven.

De derde control is de Converter Control. De images, attachments en documenten zitten in een bepaald structuur. Het is de verantwoordelijkheid van deze control om deze structuur te maken van alle images die aan hem worden gegeven. Een andere verantwoordelijkheid van deze control is ook converteren. Attachments kunnen bestaan uit images, maar ook uit Word-, Excel- of pdf-bestanden. Deze control moet ervoor zorgen alsof het lijkt dat deze bestanden ook images zijn, door Word, Excel of Acrobat Reader buiten de Image Viewer Control om te openen.



Figuur 7: Globaal overzicht van het complete product

In *figuur 7* is te zien hoe de verschillende controls met elkaar samenwerken. Als de gebruiker via de Search Control (Drovi-scherm) een aantal documenten opzoekt, zal de Search Control deze documenten doorgeven aan de Converter Control. Deze zal de structuur in al die documenten aanbrengen. Ook zal deze control de mogelijkheid bieden om de externe programma's te starten wanneer dat nodig is. Als al de document zijn gestructureerd, wordt alles doorgegeven aan de Navigatie Control. Door deze control is het mogelijk om door alle images en (externe-)bestanden te navigeren. De image die dan getoond moet worden, wordt doorgegeven aan de Viewer Control. En hiermee kan de eindgebruiker de weergave ervan veranderen, zoals zoomen, roteren, inverteren en pannen.

5.2 Ontwikkelfases

Met de opdrachtgever is afgesproken dat de analyse-, ontwerp- en implementatiefase voor elke control apart worden doorlopen. Met uitzondering van de eerste control zijn deze fases voor de controls erna snel doorlopen. Waarom de keuze is gemaakt om elke control afzonderlijk te ontwikkelen, heeft verschillende redenen. Eén van de redenen dat al eerder is aangegeven was om de verantwoordelijkheden van elke control zoveel mogelijk van elkaar te scheiden. Hierdoor wordt het ontwikkelen van de image viewer control overzichtelijker en makkelijker. Een andere reden had te maken met de tijd. De stage duurt maar tien weken. Om alle controls in één keer te analyseren en te ontwerpen, kon onhandig zijn, kon leiden tot tijdverlies en kon leiden tot een control dat niet volledig is afgerond. Dit omdat er in het begin van de stage nog te veel niet zeker was bij de opdrachtgever over de tweede, derde en vierde control. Dit kwam ook doordat een van de medewerkers, die meer wist over de functionaliteiten van deze controls, een aantal weken niet aanwezig was. Ook was het de vraag of we wel toe zouden komen aan de tweede control, omdat het ingewikkeldste en grootste deel toch de eerste control was.

MOSCOW-prioriteiten

Waar het vooral om ging tijdens de stageopdracht, was de Viewer Control. Zoals aangegeven hierboven was het de ingewikkeldste en grootste opgave van de stageopdracht. Deze control had dan

ook de grootste prioriteit. Hier nog even ter verduidelijking worden de controls voorzien van MOSCOW-prioriteit.

Viewer Control	must
Navigatie Control	should have
Converter Control	could have
Search Control	want to have

Het doel van het project was dus minimaal de Viewer Control volledig te ontwikkelen. Gelukkig zijn de Viewer en de Navigatie Control volledig ontwikkeld. Een groot gedeelte van de Converter Control is ook gedaan.

6 Analyse

In dit hoofdstuk volgt de analyse van elk van de gebouwde controls met in paragraaf 6.1 de viewer control, in paragraaf 6.2 de navigatie control en in paragraaf 6.3 de converter control

6.1 Viewer Control

In deze paragraaf wordt de analysefase van de viewer control kort behandeld. In de bijlage in de RAD wordt dat wat uitgebreider gedaan. Tijdens deze fase hebben we meerdere gesprekken gevoerd met de opdrachtgever en eindgebruikers. Dit was nodig om te weten te komen wat de image viewer control precies zou moeten kunnen doen en hoe het eruit zou moeten zien. Door deze gesprekken hebben we de eisen van de control aangepast. De eisen hebben we verdeeld tussen functionele en niet-functionele eisen. De functionele zijn hieronder nog een keer beschreven. De niet-functionele eisen zijn weer verdeeld in kwaliteitseisen en platformeisen. De kwaliteitseisen zijn ook hieronder te vinden. De platformeisen zijn terug te vinden in het RAD van de viewer control.

De functionele eisen hier hebben een aantal subnummers erbij gekregen. De eisen die achter de subnummers staan, zijn eisen die al in het RAD zijn opgenomen. Hier zijn ze nog specifieker beschreven, zodat enkele misverstanden in de eisen kunnen worden voorkomen.

Functionele eisen

Must have:

1. Een gegeven image moet getoond kunnen worden.
2. Op een getoonde image moet gezoomd kunnen worden.
 - 2.1 De getoonde image moet gezoomd kunnen worden van 10 to 1800% met de mouse wheel van de muis.
 - 2.2 De getoonde image moet gezoomd kunnen worden van 10 to 1800% door middel van het selecteren van een rectangle van een muis.
3. Een getoonde bitmap image moet geroteerd kunnen worden
 - 3.1 De getoonde image moet met 90 graden kunnen worden geroteerd.
 - 3.2 De getoonde image moet met -90 graden kunnen worden geroteerd.
4. De kleuren van een getoonde bitmap image moeten geïnverteerd kunnen worden.
5. De getoonde image moet gescrold kunnen worden met de pijltjestoetsen.
6. De getoonde image moet, met behulp van 4 knoppen, kunnen worden bekeken met de vier fitmodes
 - 6.1 De getoonde image moet kunnen worden weergegeven in de height-fitmode.
 - 6.2 De getoonde image moet kunnen worden weergegeven in de full-fitmode.
 - 6.3 De getoonde image moet kunnen worden weergegeven in de width-fitmode.
 - 6.4 De getoonde image moet kunnen worden weergegeven in de none-fitmode.

Should have:

7. De getoonde image moet kunnen worden verschoven met de muis als hij niet in z'n geheel in de image viewer te zien is. (pannen)

Niet-functionele eisen

- *Reliability*
Onder reliability van een systeem wordt verstaan de tijd dat zo'n systeem stil mag

liggen door fouten. Voor de *image viewer control* zou de reliability 99,9% moeten zijn. Met andere woorden 999 uit de 1000 keer zal de control niet falen.

- *Availability*
Dit is de tijd dat een systeem beschikbaar moet zijn. In zijn totaliteit moet de image viewer control 99,9% van de tijd beschikbaar zijn.
- *Performance*
Dit is de kwaliteit van een systeem in zijn geheel. De viewer moet onder elke omstandigheid flikkervrij zijn.
- *Response time*
De tijd die de *gebruiker* moet wachten op resultaat van een bewerking of functie, nadat hij/zij de functie heeft laten uitvoeren. Responstijd zou kleiner kunnen zijn dan *een seconde*, gemeten vanaf het moment dat de gebruiker de functie laat plaatsvinden, tot het moment dat het resultaat beschikbaar is.
- *Reusability*
Dit is het percentage van een systeem dat herbruikbaar zou moeten zijn. Het te bouwen control moet integreerbaar zijn in meerdere systemen, waardoor het 100% reusable moet zijn. Het feit dat het een control is, maakt het 100% reusable.

Struikelblok

De analyse- en ontwerpfase van een control, dat de weergave kan bewerken naar een image toe, wordt heel anders doorlopen dan die van gemiddelde andere opdrachten. Hier waren we al achter gekomen in de analysefase, omdat we tegen een aantal struikelblokken waren gelopen. Na het vastleggen van de eisen moesten de use cases worden opgesteld.

Een use case geeft de volgorde aan van verschillende handelingen die de eindgebruiker doet om een bepaalde taak uit te voeren (of om een bepaald resultaat te verkrijgen). En in geval van de viewer control zijn de use cases extreem kort. Als een eindgebruiker een taak wil uitvoeren, is de enige handeling die hij moet doen, is een knop drukken of de muis gebruiken. Na de handeling voert de image viewer meteen zijn taak uit. En dit geldt voor elke taak die de image viewer kan uitvoeren. Om te laten zien hoe kort de use cases kunnen zijn volgt hier voorbeeld van een use case voor het roteren van een image.

Use Case: Roteren

Actor actions

1. Gebruiker drukt op roteerknop

System responses

2. Toont de image geroteerd

Om zulke korte use cases te schrijven vonden wij niet erg zinvol. Hiervoor hadden wij een andere oplossing gevonden. Dit hebben wij opgelost door de verschillende handelingen die een eindgebruiker kan uitvoeren in één use case te zetten. Dit is als volgt gegaan.

Use Case: Verander de view op de image

Actor: Gebruiker

Beginsituatie: Er is een image getoond

1. De gebruiker kiest een bewerking. Er kan gekozen worden uit:
 - a. Pan: Met de muis de image verschuiven
 - b. Invert Colors: Alle kleuren van de image inverteren
 - c. Scroll: De image horizontaal of verticaal verschuiven
 - d. Rotate: De image roteren
 - e. Fit: De grootte van de image aanpassen aan de hand van 4 verschillende modes:
 - Height: De image wordt over de hele hoogte zichtbaar
 - Widht: De image wordt over de hele breedte hoogt zichtbaar

- Full: De image wordt in zijn geheel (hoogte en breedte) zichtbaar
 - None: De image wordt in zijn normale grote weergegeven
 - f. Zoom: Op de image in- of uitzoomen
 - 2. De image viewer control voert de bewerking uit en toont de bewerkte image
- Eindsituatie:** De image wordt getoond zoals gewenst

Verder kon nog een use case worden geformuleerd, namelijk:

Use Case: Toon nieuwe image

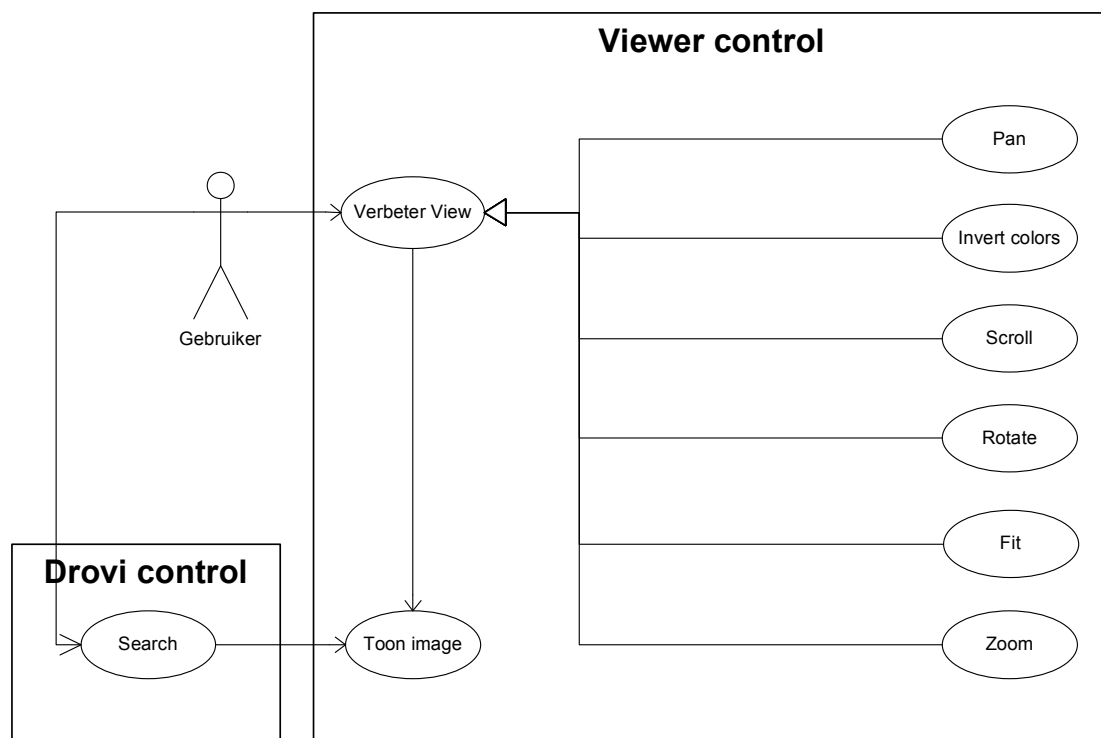
Actor: Gebruiker

Beginsituatie: Gebruiker dient een polisblad te bekijken

1. De gebruiker zoekt via Drovi het betreffende polisblad
2. Drovi zoekt de image van het blad op, laadt het en geeft het door aan de image viewer control
3. De image viewer control laat het polisblad zien.

Eindsituatie: De image van het polisblad is te zien

Het use case diagram ziet er als volgt uit.



Figuur 8: Use case diagram Viewer control

6.2 Navigatie Control

Het navigatie control is de tweede control behorende bij dit project. Zoals eerder gezegd zijn de documenten binnen DSW gestructureerd. Een 'document' kan bestaan uit 0 of meerdere image bestanden én kan een document 0 of meerdere toevoegingen (attachments) bevatten. De attachments kunnen elk mogelijke extensie hebben.

In de huidige situatie worden alle image bestanden achter elkaar geplakt, gevolgd door de attachments. Image bestanden worden ter tentoonstelling aan de image viewer gegeven, waar de attachments via een ActiveX control aan de juiste applicaties gegeven worden, mits die door de image viewer getoond kunnen worden.

Echter zal in dit control geen rekening gehouden worden met andere dan image bestanden. Het omzetten en/of doorsturen naar externe applicaties van niet-image bestanden zal gebeuren in een volgende control. Het navigatie control zal zich uitsluitend bezig houden met de correcte navigatie tussen te tonen documenten.

Eindgebruikers hebben beschikking tot een aantal knoppen voor navigatie binnen een document. Onder navigatie wordt verstaan:

- een pagina verder navigeren
- een pagina terug navigeren
- naar de laatste pagina navigeren
- naar de eerste pagina navigeren

Functionele eisen

Must have:

1. er moet genavigeerd kunnen worden tussen de daadwerkelijke pagina's (images) van een document.
2. er moet genavigeerd kunnen worden tussen de attachments van een document.
3. er moet genavigeerd kunnen worden tussen een rij van documenten.

Should have

4. navigatie knoppen moeten alleen worden getoond als de betreffende navigatie mogelijk is.

Niet-functionele eisen

Als niet-functionele eisen gelden dezelfde als de *viewer* control.

Voor de use cases geldt het zelfde als bij de vorige control, namelijk dat een aantal extreem korte use cases geformuleerd kunnen worden. Dit is opgelost door net als bij de image viewer control alle use cases samen te vatten in een algemene use case. Die is als volgt geformuleerd.

Use Case: navigeer

Actor: Gebruiker

Beginsituatie: Er is een array van documenten (images) beschikbaar, waarvan 1 image getoond is.

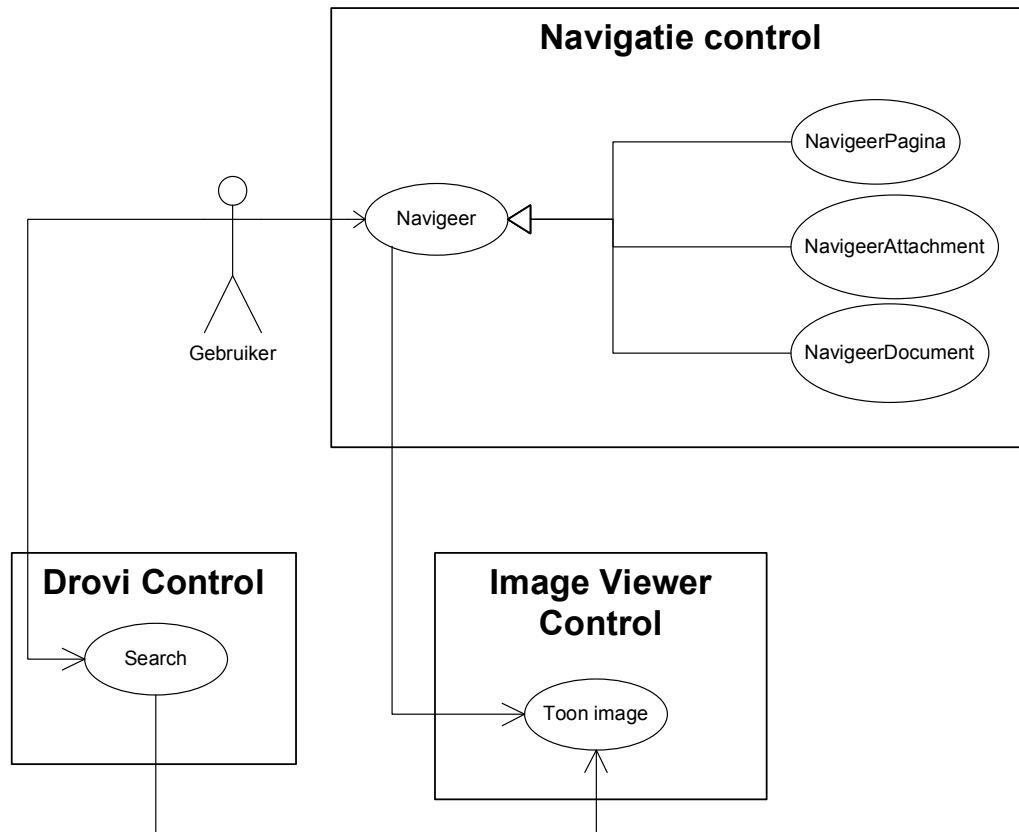
De gebruiker wil een ander image, behorende bij de array van documenten zien.

1. De gebruiker drukt op de knop met de gewenste navigatie mogelijkheid. Gekozen kan worden uit:
 - a. Eerste pagina (van het eerste document uit de array van documenten)
 - b. Laatste pagina (van het laatste document uit de array van documenten)
 - c. Volgende pagina
 - d. Vorige pagina
 - e. Volgend subdocument (attachment)
 - f. Vorig subdocument (attachment)
 - g. Volgend document
 - h. Vorig document

2. Het control zoekt het gewenste image en geeft het aan de image viewer control om het te tonen.

Eindsituatie: Het gewenste image is te zien in de image viewer.

Ten slotte staat in *figuur 9* het diagram behorende bij de use case.



Figuur 9: Use case diagram Navigatie control

6.3 Converter Control

Deze control is de derde behorende bij dit project. Het doel van deze control is het ophalen van documenten behorende bij gegeven id's uit *InfoImage* en die om te zetten in een door de navigatie control gewenste structuur.

De huidige image viewer gebruikt een *ActiveX control* om de documenten op te halen en ,aan de hand van de extensie van de pagina's in zo'n document, die opent in de image viewer of een externe applicatie bij niet-image bestanden.

Vanwege de tijdsdruk is besloten ook in de nieuwe image viewer gebruik te maken van het *ActiveX control*. De control moet documenten ophalen en omzetten in een structuur als besproken in het vorige paragraaf. Ook van de mogelijkheid om niet-image bestanden in externe applicaties te openen zal gebruik gemaakt moeten worden.

Het ActiveX control

De control werkt als volgt. Voor documenten uit de database gehaald kunnen worden, moet een gebruiker natuurlijk eerst ingelogd zijn. Dit kan gedaan worden door de methode *CALLogin* uit de klasse *CALConnector* aan te roepen. Vervolgens moet uit die zelfde klasse

de methode *CALGetColDocs* aangeroepen worden. Die methode geeft een *colOpenDoc* terug. Met behulp van de *colOpenDoc* klasse kan dan met de methode *Add*, die een *InfoImageID* als invoer krijgt, een *clsOpenDoc* opgehaald worden. De *clsOpenDoc* klasse is de klasse waarin zich een *infoImage* document bevindt. Uit die klasse kan met behulp van de methode *SavePage* een document pagina uit de database gehaald worden. De methode slaat de pagina ergens op schijf op en geeft dan de path van die locatie terug.

De control bevat een aantal plug-ins om niet-image bestanden toonbaar te maken. Een daarvan is bijvoorbeeld *GhostScript PDF Plug-in*. Elke applicatie waarvan een plug-in beschikbaar is voor de control, kan worden geopend door de control. Bij het openen van een externe applicatie wordt het betreffende niet-image bestand ook meegegeven. Dit gaat met behulp van de methode *LaunchEx* uit de klasse *clsOpenDoc*. Voor een pagina uit het document opgehaald wordt, word gecontroleerd of de pagina wel te openen is met deze control, zij het in de image viewer of extern. Dit wordt gedaan door met de methode *ImportExtension* de extensie per bijlage van het document op te vragen en te controleren.

Functionele eisen

Must have's:

1. Een gegeven document moet uit *InfoImage* gehaald kunnen worden.
2. Een opgehaald document moet omgezet kunnen worden in een door de navigatie control gewenste structuur.
3. Een niet-image bestand moet extern geopend kunnen worden.

Niet functionele eisen

Wederom zijn deze dezelfde als voor de *viewer* control.

Voor de converter control zijn twee use cases geformuleerd. Deze worden echter niet *direct* door de gebruiker aangeroepen, maar *indirect* via een van de andere controls. De use cases zijn als volgt geformuleerd.

Use Case: GetDocument

Actor: Gebruiker

Beginsituatie: De gebruiker heeft een document gevonden en wil die getoond hebben.

4. De control haalt het document op uit *InfoImage*
5. Uit het document worden alle pagina's en bijlagen gehaald en in een door de navigatie control gewenste structuur gezet.
 - a. Voor niet-image bestanden wordt een standaard image bestand geplaatst, aangevend dat het bestand extern geopend moet worden.

Eindsituatie: Het gewenste document is geladen met de eerste pagina getoond in de image viewer.

Use Case: OpenPageExtern

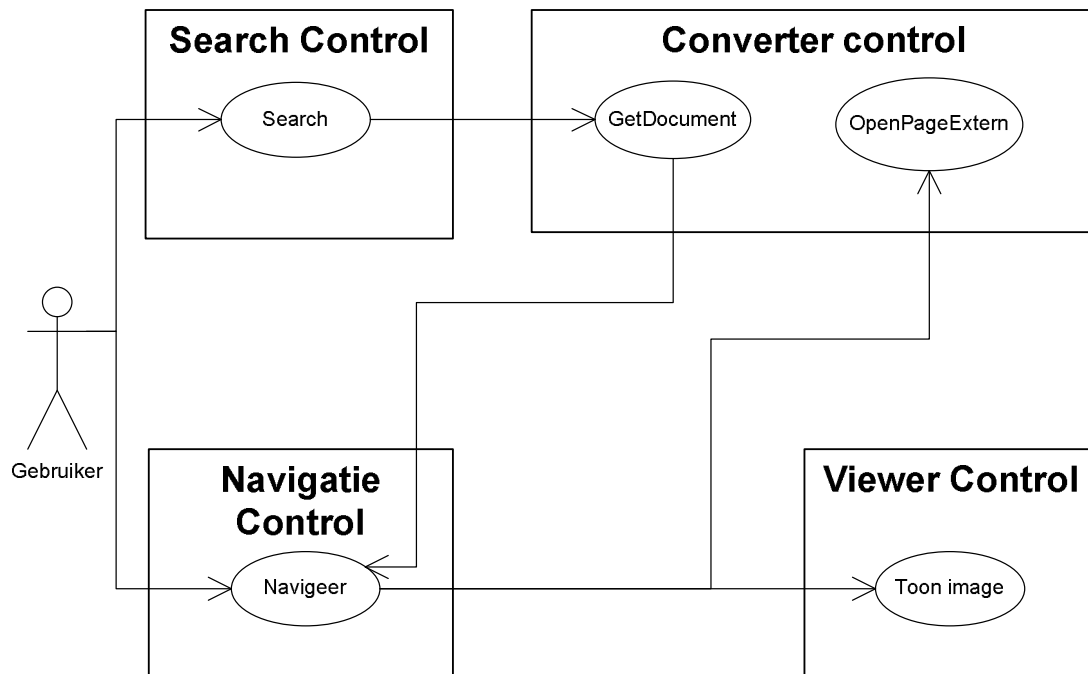
Actor: Gebruiker

Beginsituatie: Een document is geladen en de gebruiker navigeert naar een pagina dat een niet-image bestand voorstelt.

1. Het navigatie control geeft door aan het converter control welk niet-image bestand bekeken wil worden.
2. Via het ActiveX control wordt dat bestand extern geopend.

Eindsituatie: De gewenste pagina of bijlage is extern getoond.

Het use case diagram behorende bij de converter control en de bovenstaande use cases volgt in *figuur 10*



Figuur 10: Use case diagram Converter control

7 Ontwerp

In dit hoofdstuk volgt het ontwerp van dit project in het geheel. Gestart zal worden met het user interface in paragraaf 7.1, gevolgd door het systeem ontwerp in paragraaf 7.2 en het testplan in paragraaf 7.3

Struikelblok

Bij het beginnen van het ontwerpen van de eerste control (de viewer control) stonden we al met een grote vraag: Hoe ontwerp je iets dat een image kan laten zien en dat ook nog de weergave ervan kan veranderen? Zoals al eerder is aangegeven in paragraaf 6.1 is het ontwikkelen van zo'n stukje software anders dan de meeste andere opdrachten. We hadden totaal geen idee. Omdat het voor ons nog onduidelijk was, hebben we eerst onderzocht hoe het mogelijk is met het .NET-framework. En tegelijker tijd hebben we ook kennis moeten opdoen van het .NET-framework, omdat ook daarvan wij niks van af wisten. En na het onderzoek zijn we gaan ontwerpen.

In het boek Objected-Oriented Software Engineering [1] staat, dat ontwerpen het proces is van beslissen hoe de eisen zouden moeten worden geïmplementeerd met de beschikbare technologie. Maar omdat we nog niet veel wisten over de beschikbare technologie, hebben we voor het beginnen met ontwerpen eerst de beschikbare technologie bestudeerd.

Maar mede door het bestuderen van hoe een Viewer Control ontwikkeld kan worden en wat precies mogelijk is met het framework, hebben we tijdens de ontwerpfase een aantal onderdelen niet goed aangepakt.

Om precies te zijn is bij het opstellen van de volgordediagrammen iets mis gegaan. Dit had te maken met de use case, waarbij de verschillende handelingen die mogelijk zijn, zijn samengebracht. Dit om vele zeer korte use cases van elke handeling te voorkomen. Dit staat in paragraaf 6.1 beter uitgelegd. De eisen die zijn opgesteld in de RAD moeten als het ware traceerbaar zijn tussen de RAD en het ontwerpdocument. En dit was die niet precies gedaan in het ontwerpdocument van de Viewer Control.

In de RAD zijn de verschillende handelingen (requirements) van de Viewer Control in één use case geplaatst, maar dat was niet het geval in de volgordediagrammen. Voor elke handeling (requirement dus) was er een apart volgordediagram gemaakt. Hier moet eigenlijk ook maar één volgordediagram worden gemaakt met daarin alle handelingen inbegrepen.

Volgordediagrammen tonen de volgorde van de berichten tussen de objecten die een bepaalde handeling willen uitvoeren. De volgordediagrammen die zijn gemaakt zijn ook te gedetailleerd weergegeven. We hadden te veel gebruik gemaakt van de kennis die opgedaan was tijdens het onderzoeken voor het ontwerpen. We hadden dit gedaan omdat er anders heel kleine volgordediagrammen moesten worden gemaakt die eigenlijk niet veel zin hebben.

Deze misverstanden zijn vooral ontstaan door de angst een te kleine ontwerpdocument te schrijven. De analyse- en ontwerpfase van deze opdracht worden toch anders doorlopen dan menig ander opdracht.

7.1 User Interface

In deze paragraaf wordt de user interface van de gehele image viewer toegelicht.

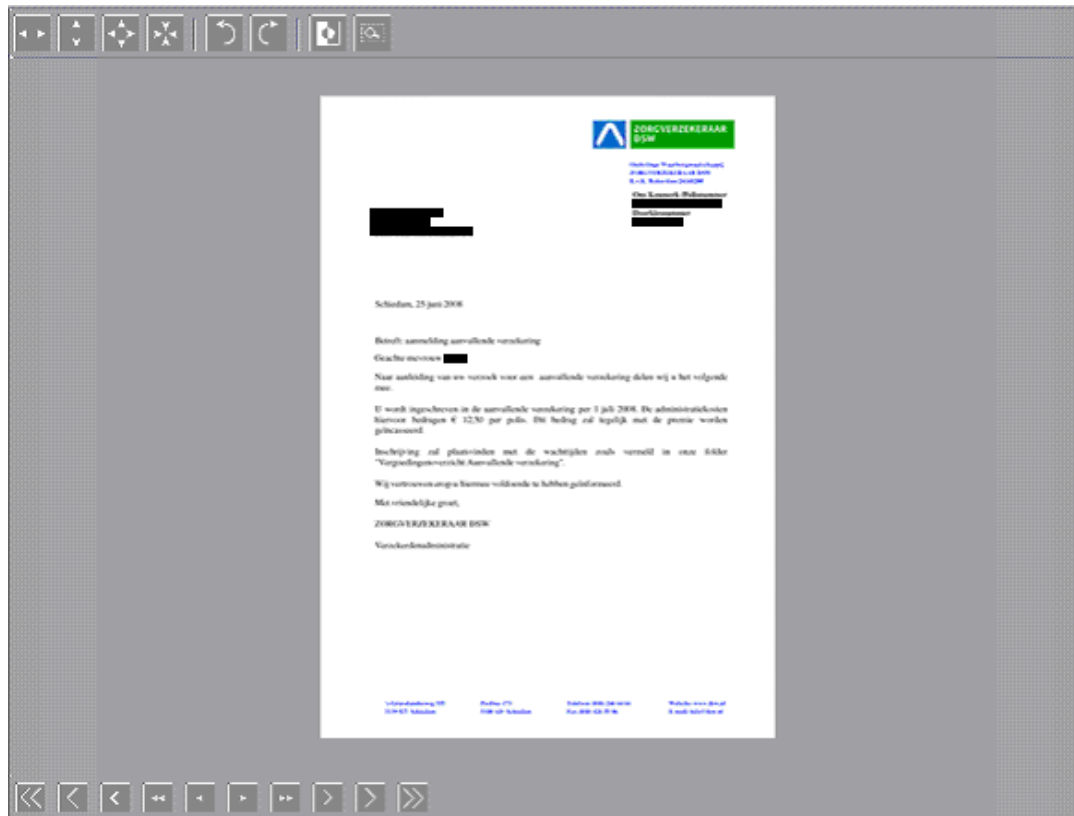
De opdracht is om een control te maken en niet een applicatie. In Visual Studio is er de mogelijkheid om te kiezen uit templates. Voor het ontwikkelen van een control is er ook een template, waarvan dus gebruik is gemaakt. Een standaard usercontrol klasse wordt dan

gegenereerd door Visual Studio. Maar in plaats van een normale windows-form voor een normale applicatie, wordt er voor een control een usercontrol-form toegevoegd.

De opbouw van de user interface is erg belangrijk bij het ontwerpen en implementeren van een viewer control. Als de view naar een image wordt veranderd, moet een deel van de user interface veranderen. Ofwel een aantal van de componenten, waaruit de user interface is opgebouwd, moeten zich aanpassen om de gevraagde view weer te geven. Het aanpassen van de componenten zal dus moeten gebeuren met behulp van een aantal klassen die moeten worden ontworpen en geschreven. Belangrijk is dus ook de keuze van de componenten die worden gebruik. Want elk component heeft weer zijn specifieke eigenschappen en mogelijkheden.

In de usercontrol-form van de eerste control bestaat uit een toolbar met knoppen en een zelfgeschreven container klasse. De containerklasse is verantwoordelijk voor het vergroten en verkleinen van de gegeven image. De user interface van de eerste control ziet er dus heel simpel uit. Het bestaat alleen uit de knoppenbalk met daaronder de containerklasse waar de image in zal worden getoond. Alle functionaliteiten zullen vanuit de knoppenbalk en de containerklasse beschikbaar zijn. De meeste functionaliteiten kunnen worden uitgevoerd door op één van de buttons in de user interface te drukken. Door op de linker- of rechtermuisknop te klikken in de containerklasse worden er ook een aantal functionaliteiten geactiveerd. In het RAD van de viewer control staat een uitgebreide uitleg van wat de knoppen en muisbewegingen precies doen.

In de usercontrol-form van de tweede control wordt dan de Viewer Control toegevoegd en de onderste toolbar. Deze knoppen zorgen voor de navigatie van de images. Een uitgebreide uitleg van deze knoppen staat in de RAD van de navigatie control. Hieronder in *figuur 11* wordt de user interface weergegeven.



Figuur 11: User Interface van de Viewer Control in de Navigatie Control

7.2 Systeem ontwerp

In deze paragraaf wordt het hele ontwerp van de image viewer control toegelicht. Zoals al eerder aangegeven is de image viewer control verdeeld in vier aparte controls. In hoofdstuk 5 wordt kort uitgelegd wat elke control precies doet. Ook wordt daar uitgelegd waarom het is verdeeld in vier controls. De controls die zijn ontwikkeld, zijn allemaal apart van elkaar ontwikkeld. Voor elke control zijn dan ook aparte RAD's en designdocuments ontwikkeld. Deze zijn allemaal te vinden als bijlage. De details die niet aanbod komen in deze paragraaf zijn wel te vinden in de bijlages.

Viewer Control

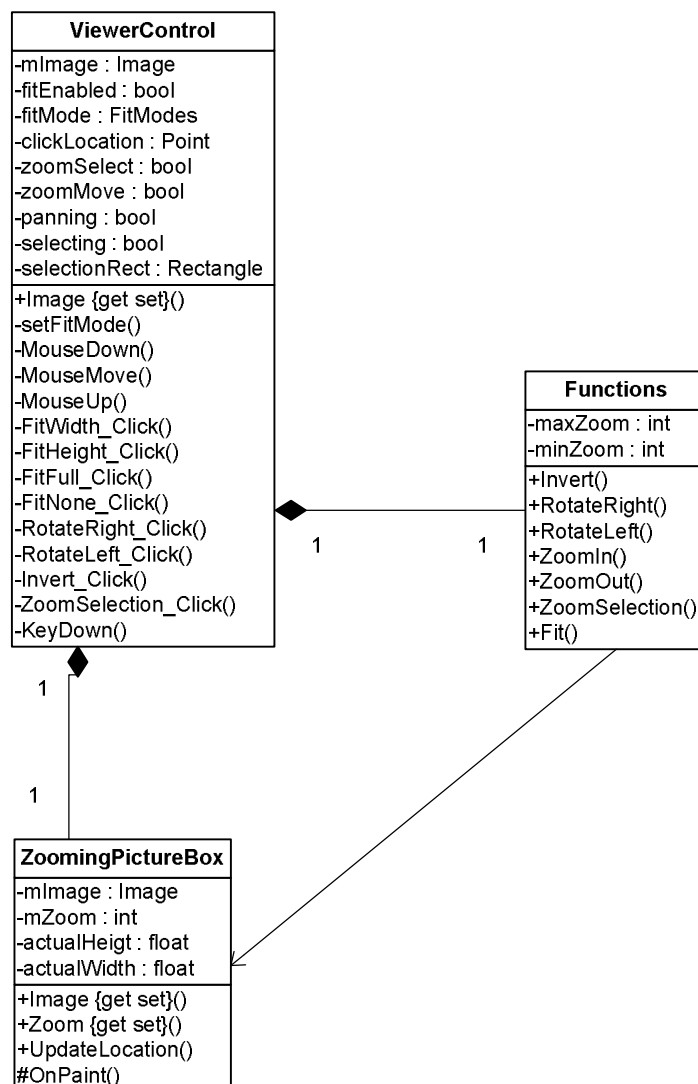
De viewer control is de eerste control en is eigenlijk de control waar het eigenlijk om draait. De images worden hierin bekeken. In paragraaf 6.1 bij de functionele eisen zijn de functionaliteiten die deze control heeft beschreven. Om deze functionaliteiten mogelijk te maken zijn er 3 klassen ontworpen: ViewerControl, Functions en ZoomingPictureBox. Wat ze precies inhouden wordt hier beschreven:

ViewerControl Dit is het hoofdscherm van de Image viewer control. Dit is de user interface met de button- en mouse-events die kunnen worden aangeroepen door de eindgebruikers.

ZoomingPictureBox Deze klasse zorgt voor de gevraagde weergave van de eindgebruiker van een bitmap.

Functions In deze klasse zitten functies die ViewerControl kan gebruiken om de bitmap weer te geven zoals de eindgebruiker dat wilt.

De klasse ViewerControl wordt opgebouwd uit een toolbar en daaronder de ZoomingPictureBox. In de klasse Functions staan de methodes om de weergave van de Image in de ZoomingPictureBox te veranderen. Bij elke keer als wordt gevraagd de weergave van de Image te veranderen zal de ZoomingPictureBox zichzelf opnieuw tekenen met de nieuwe weergave. De methode die dan wordt opgeroepen, berekent de juiste zoomfactor waarmee de Image moet worden getekend. Ook als er wordt gevraagd een andere deel van de Image weer te geven, wordt berekend vanaf welke positie van de Image dat moet worden gedaan. De nieuwe zoomfactor en positie worden dan aan de ZoomingPictureBox doorgegeven. Met behulp van deze informatie wordt in de OnPaint methode van de ZoomingPictureBox de Image opnieuw getekend. De klassendiagram van de Viewer Control is hieronder te zien.



Figuur 12: Klassendiagram van de viewer control

Navigatie Control

De Converter Control zal ervoor zorgen dat de images in de juiste structuur worden gezet. Zoals eerder aangegeven in paragraaf 6.2 zal de Navigatie Control zich uitsluitend bezighouden met de correcte navigatie tussen images en deze structuur. De image die dan moet worden getoond wordt dan doorgegeven aan de Viewer Control. En de eindgebruiker kan dan de image bekijken op welke manier dan ook.

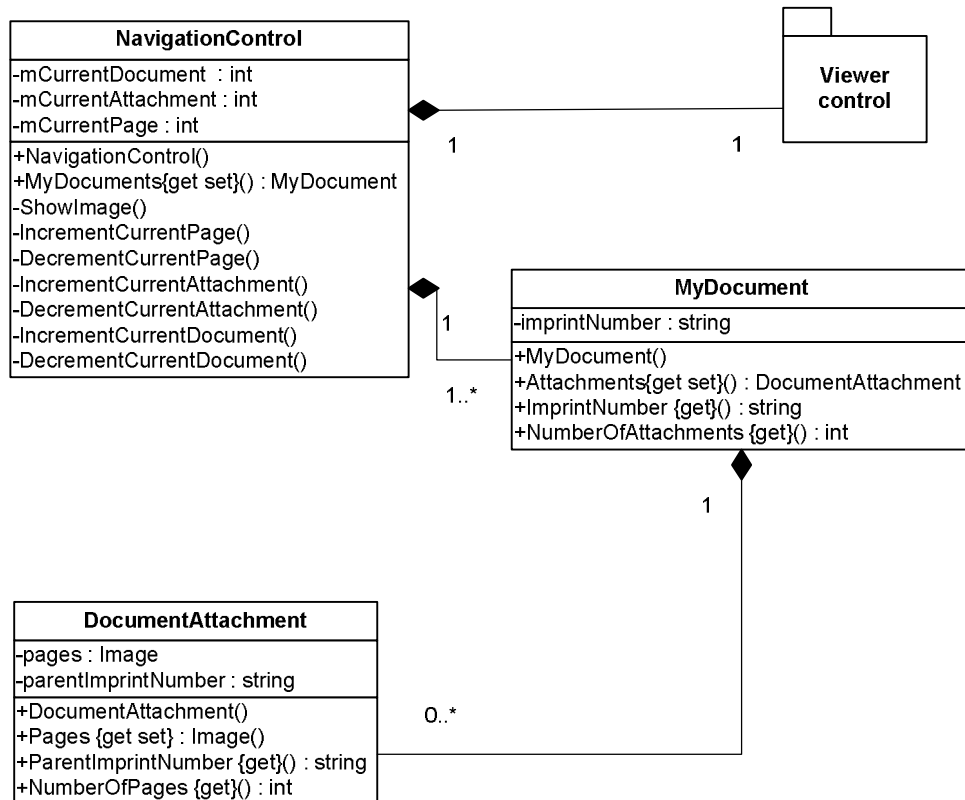
Om de navigatie mogelijk te maken zijn 3 klassen gemaakt: NavigatieControl, MyDocument en DocumentAttachment. Wat ze precies inhouden wordt hier beschreven:

NavigatieControl

Dit is het hoofdscherm van de navigatie control. Dit is de user interface met de button- en key events die kunnen worden aangeroepen door de eindgebruikers.

MyDocument Deze klasse stelt een document met één of meerdere DocumentAttachments voor.

DocumentAttachment Deze klasse stelt een bijlage met één of meerder pagina's (images) voor.



Figuur 13: Klassendiagram van de navigatie control

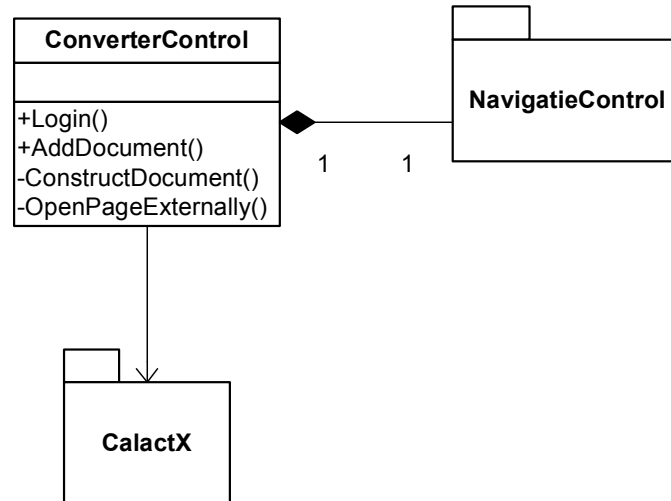
In *figuur 13* is de klassendiagram te zien. Hierin is ook de Viewer Control (blackbox) in opgenomen, om de samenwerking aan te tonen. De NavigatieControl maakt namelijk de Viewer Control aan.

Converter Control

Dit control heeft als taak het ophalen en omzetten van documenten uit *InfoImage* naar het voor de navigatie control gewenste structuur, een *MyDocument* object. Als invoer krijgt het control een *id* waarmee een document uit *InfoImage* gehaald kan worden. De converter control moet ook niet-image bestanden kunnen openen in externe applicaties.

In de huidige image viewer wordt gebruik gemaakt van een *ActiveX* control, genaamd *CalactX*, om het ophalen en omzetten van documenten te realiseren. Vanwege beperkte tijd om dit control te bouwen, is besloten ook bij de nieuwe image viewer gebruik te maken van het *ActiveX* control. *CalactX* wordt gebruikt om in te loggen op *InfoImage*, documenten op te halen en niet-image bestanden te openen in externe applicaties.

Omdat dit control, los van het construeren van *MyDocument* objecten, alleen functies van *CalactX* moet aanroepen, bestaat het maar uit een klasse, namelijk *ConverterControl*. Dit is te zien in *figuur 14* met een klassendiagram van de converter control.



Figuur 14: Klassendiagram van de converter control

7.3 Testen

In deze paragraaf komen de tests aan bod. Er wordt besproken over de aanpak van de tests. Voor de echte testklassen wordt verwezen naar het *Acceptatie & Testplan* in bijlage J. De test resultaten worden besproken in hoofdstuk 8, waar de *implementatie* van de controls wordt besproken.

Zoals in het *Acceptatie & Testplan* staat geschreven, zal getest worden middels *automatisch* en *handmatig* testen. Handmatige tests zullen *functioneel* uitgevoerd worden, waar automatische tests *structureel* uitgevoerd zullen worden. Ook zal gebruik gemaakt worden van een *coverage* tool, waarmee de kwaliteit van de tests bepaald kunnen worden.

Voor de handmatige tests wordt voor elk control een *windows applicatie* gemaakt door het control te plaatsen in een *form*. Hierdoor kan elk control als een applicatie gestart worden en de handmatige tests uitgevoerd kunnen worden.

Voor de automatische tests wordt voor elk control een *test project* gemaakt. In zo'n project bevinden zich dan de automatische test-cases.

8 Implementatie

In dit hoofdstuk zal de implementatie van de verschillende controls besproken worden. Elk control zal apart besproken worden, in de volgorde dat die zijn geïmplementeerd. Gestart wordt de implementatie van de *user interface* in paragraaf 1 met daarna de implementatie details van de viewer control in paragraaf 2 en in paragraaf 3 de beschrijving van het navigatie control. Dit hoofdstuk wordt afgesloten met de implementatie beschrijving van de test klassen.

8.1 User Interface

Zoals eerder aangegeven is de implementatiefase doorlopen met behulp van Visual Studio. De visuele objecten van een user interface implementeren als je gebruik maakt van Visual Studio is niet nodig. Wel kunnen er aantal kleine zaken aangepast worden door zelf te coderen. Maar voor de grootste gedeelte kunnen de visuele objecten gewoon worden gesleept op de user interface. In *figuur 15* is dit te zien. Visual Studio genereert dan de code van de objecten. Aanpassingen kunnen worden gedaan door de beschikbare tabellen in Visual Studio. En natuurlijk moet er worden gecodeerd wat de buttons precies moeten doen. In C# is het mogelijk om één klasse te coderen in twee verschillende files. Voor de user interface klasse wordt de gegenereerde code in één file gezet. De code die zelf moet worden geschreven voor bijvoorbeeld de buttonevents, wordt in een andere file gezet.

De user interface bestaat voor het grootste gedeelte uit de user interface van de Viewer Control. Deze is ook gewoon gesleept op de user interface van de Navigatie Control, welke op zijn beurt ook gesleept is op de user interface van de converter control. Voor de user interface van de navigatie control is een extra toolbar geplaatst voor de buttons voor het navigeren tussen de images. De zichtbaarheid van de navigeerbuttons zijn niet onder alle omstandigheden altijd het zelfde. De zichtbaarheid wordt wel bepaald door zelfgeschreven code. Want deze hangt af van de opbouw van de gegeven documenten die moeten worden getoond.

Er zijn verschillende navigeerbuttons gemaakt. In het ontwerpdocument is aangegeven welke dat allemaal zijn. Er is de mogelijkheid om te navigeren tussen images, attachments en documenten. Maar als bijvoorbeeld een document maar uit één attachments bestaat, zullen de navigeerbuttons voor de attachments verdwijnen. Een stukje implementatie hier voor is te zien in *figuur 15*. Dit is ook gedaan voor de navigeerbuttons voor de documents. Zie *figuur 16*.

Er zijn ook buttons om naar de eerste of laatste image te gaan. Maar als de eerste of laatste image al wordt getoond zullen deze buttons worden gedisabled. Zo is het ook voor alle next-buttons als de laatste image wordt getoond.

```

/// <summary>
/// make navigation attachment buttons invisible when there are not many
/// attachments in the current document
/// </summary>
private void HideNecessaryNavigationAttButtons()
{
    if (this.mDocuments != null)
    {
        int amountAttachments = mDocuments[mCurrentDocument].NumberOfAttachments;
        if (amountAttachments < maxAmountAttForAttNavigationButtons)
        {
            this.navigateAttachmentLeftButton.Visible = false;
            this.navigateAttachmentRightButton.Visible = false;
        }
        else
        {
            this.navigateAttachmentLeftButton.Visible = true;
            this.navigateAttachmentRightButton.Visible = true;
        }
    }
}

```

Figuur 15: Hide necessary Navigatie attachment buttons.

```

/// <summary>
/// make navigation document buttons invisible when there are not many
/// documents
/// </summary>
private void HideNecessaryNavigationDocButtons()
{
    if (this.mDocuments != null)
    {
        int amountDocuments = this.mDocuments.Length;
        if (amountDocuments < maxAmountDocumentForLeftRightButton)
        {
            this.navigateDocumentLeftButton.Visible = false;
            this.navigateDocumentRightButton.Visible = false;
        }
        if (amountDocuments < maxAmountDocumentForBeginEndButton)
        {
            this.navigateDocumentBeginButton.Visible = false;
            this.navigateDocumentEndButton.Visible = false;
        }
    }
}

```

Figuur 16: Hide necessary Navigatie document buttons.

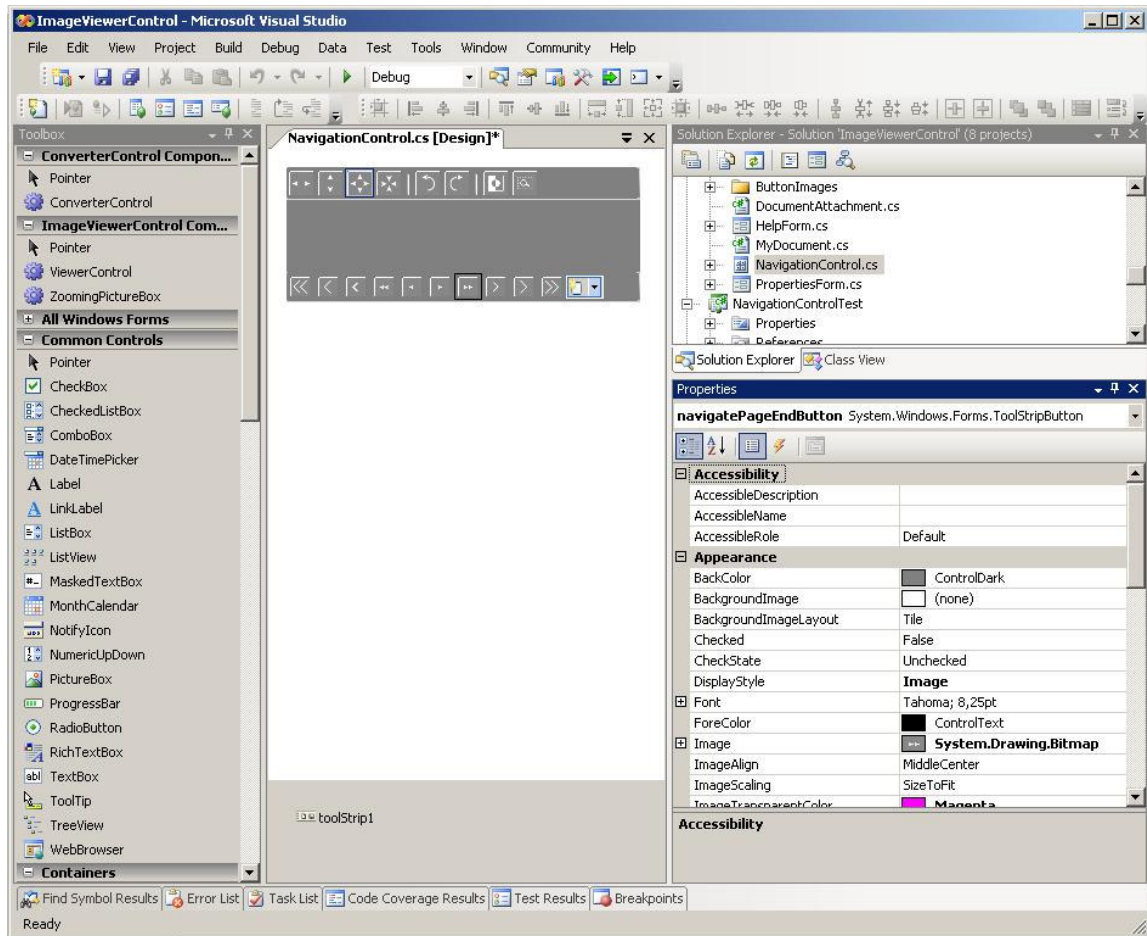
Aanpassen

De ICT-medewerkers van DSW kunnen met behulp van Visual Studio zelf een aantal kleine veranderingen toepassen op de user interface van de Navigatie Control. Net zoals dat gedaan kan worden bij de Viewer Control. Bijvoorbeeld de kleuren veranderen of de grootte van de buttons. In *figuur 17* is een printscreen van Visual Studio met het ontwerpscherm van de Navigatie Control. Aan de linkerzijde is de toolbox te zien. Hiermee kunnen visuele objecten geplaatst worden in de control. Rechtsonder is de propertietable. Hiermee kunnen de eigenschappen van de objecten, die zijn geplaatst in de control, worden veranderd.

Wat ook verandert kan worden door de ICT-medewerkers is het moment wanneer de navigeerbuttons verdwijnen. De navigeerbuttons voor de attachments verdwijnen als de document maar uit één attachment bestaat. Maar het kan worden ingesteld dat ze pas verdwijnen als het document uit twee attachments bestaan. Dit kan gedaan worden door de constante

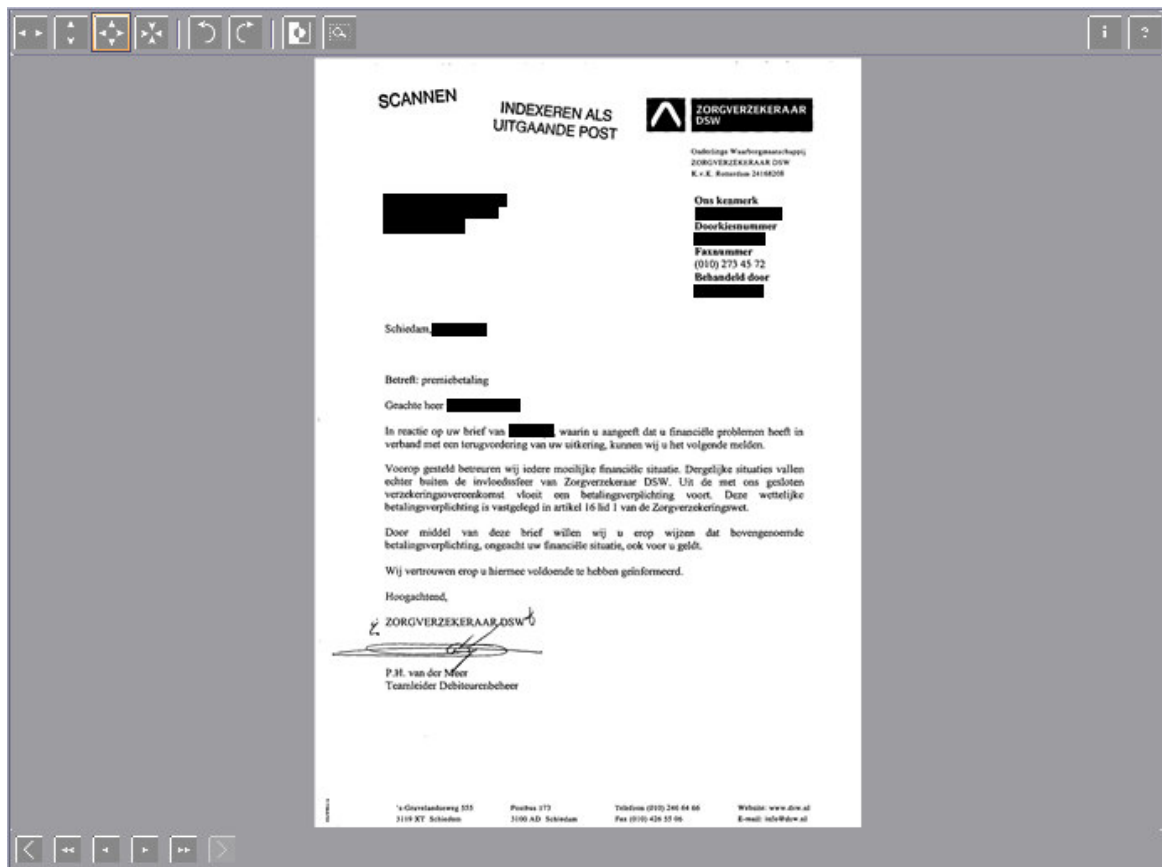
`maxAmountAttForAttNavigationButtons` die te zien in de code in *figuur 15* te veranderen van waarde.

Door de navigatiebuttons eerder of later te laten verdwijnen bij meer of minder documents, kunnen de constanten `maxAmountDocumentForLeftRightButton` en `maxAmountDocumentForBeginEndButton` worden veranderd.



Figuur 17: Visual Studio met ontwerpscherm

Na volledige implementatie ziet het uiteindelijke user interface als volgt uit.



Figuur 18: Uiteindelijke User Interface

8.2 Viewer Control

In deze paragraaf wordt de implementatie van de Viewer control besproken. De viewer control moest images kunnen tonen er die vervolgens kunnen roteren, de kleuren invertieren, vergroten, verkleinen, passen aan de hand van gedefinieerde fit modes en verschuiven.

De implementatie is gestart met de *ZoomingPictureBox*. Dit is simpel gezegd een *panel* waarop een image wordt getekend. Dit gebeurt als in *figuur 19* te zien is.

```
/// <summary>
/// update all necessary information and draw the image accordingly.
/// </summary>
/// <param name="e"></param>
protected override void OnPaint(PaintEventArgs e)
{
    if (image != null)
    {
        if (this.paintUsingInterpolation)
            e.Graphics.InterpolationMode = System.Drawing.Drawing2D.
                InterpolationMode.HighQualityBicubic;

        e.Graphics.Clear(this.BackColor);
        e.Graphics.DrawImage(image, left, top, actualWidth, actualHeight);
    }
    base.OnPaint(e);
}
```

Figuur 19: Het tekenen van een image

In *figuur 19* is te zien dat een image met *bicubic interpolation* getekend wordt. Dit zorgt voor

Een goed leesbare image als die klein getoond wordt. Uit het figuur is ook waar te nemen dat het tekenen met interpolation niet altijd gebeurt. Interpolation wordt namelijk uitgeschakeld tijdens het uitvoeren van een bewerking zoals zoomen of pannen. Na de bewerking wordt de interpolation weer ingeschakeld, waardoor een mooier plaatje te zien is.

Om bij een vergrote image het juiste gedeelte te tonen wordt de locatie waar de image wordt getekend aangepast. In *figuur 20* is een voorbeeld hoe de x-waarde van die locatie wordt berekend.

Figuur 20 laat ook zien dat de berekening van de locatie van de image afhankelijk is van de waarden `widthRatio` en `clientWidthRatio`. De eerste is de verhouding van de locatie van het gewenste middelpunt van het toonbare gedeelte van de image tot de echte breedte van de image. De tweede is de verhouding van dat zelfde punt tot de breedte van de *clientRectangle* van de picture box.

Met de *ZoomingPictureBox* in hand kon de implementatie van de *Viewer control* beginnen. Daaraan werden een *zoomingPictureBox* en een *toolstrip* toegevoegd. Ook werd de klasse *Functions* toegevoegd. Het idee achter deze klasse is om alle gewenste functies van de viewer control echt uit te voeren, waar de viewer control over het algemeen naar buttons en andere *events* luistert en de betreffende functies aanroept uit *Functions*. Zie *figuur 21* voor een voorbeeld uit de viewer control en *figuur 22* voor een voorbeeld uit functions.

```
private void UpdateLeft()
{
    // the image is larger than the container
    if (this.HorizontalScroll.Visible)
    {
        // so make sure it can't be moved out of this container; X value can't be
        // larger than 0
        if (((this.Width * this.clientWidthRatio) - (actualWidth *
                                                    this.widthRatio)) >= 0)
        {
            this.left = 0;
        }
        // can't be smaller than the containers' width - actualWidth
        else if((
            (this.Width * this.clientWidthRatio) - (actualWidth * this.widthRatio))
            <= (this.ClientRectangle.Width - this.actualWidth))
        {
            this.left = this.ClientRectangle.Width - this.actualWidth;
        }
        else
        {
            // just center the image at the desired point
            this.left = ((this.Width * this.clientWidthRatio) - actualWidth *
                        this.widthRatio);
        }
    }
    else // the image is smaller than the container, so center it at the actual
        // center point of the image
    {
        this.left = ((this.ClientRectangle.Width * defaultRatio) - actualWidth *
                    defaultRatio);
    }
}
```

Figuur 20: Berekening van de x-waarde van de locatie

```
/// <summary>
/// rotate the image 270 (-90) degrees
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void RotateLeft_Click(object sender, EventArgs e)
{
    this.functions.RotateLeft();
}
```

Figuur 21: Afhandeling van de RotateLeft_Click event

```
/// <summary>
/// Rotate the image left (270 degrees) and fitting if necessary.
/// </summary>
public void RotateLeft()
{
    if (this.picBox.Image != null)
    {
        //get the current ratio's for the center point
        float widthRatio = this.picBox.WidthRatio;
        float heightRatio = this.picBox.HeightRatio;

        this.picBox.Image.RotateFlip(RotateFlipType.Rotate270FlipNone);
        //set new ratios for the center point to stay on screen after
        //rotation
        this.picBox.WidthRatio = heightRatio;
        this.picBox.HeightRatio = 1 - widthRatio;

        if (fitEnabled)
            this.Fit(this.fitMode);

        this.picBox.UpdateAll();
    }
}
```

Figuur 22: Uitvoering van de Rotate left functie

Alle functies zijn op dezelfde manier als aangegeven voor de *Rotate Left* functie geïmplementeerd.

De zooming functionaliteit van dit control is op drie manieren geïmplementeerd. Gebruikers kunnen *inzoomen* op een selectie door de *zoom Selection* knop te drukken en dan de selectie te maken. Ook kan *in-* en *uitgezoomd* worden met de *scrollwheel* en door de *rechter* muisknop ingedrukt te houden en de muis langs een verticale lijn te bewegen.

8.3 Navigatie Control

Deze paragraaf zal een beschrijving geven over de implementatie van de Navigatie Control. Gestart zal worden met de implementatie van ondersteunde structuren, zoals *MyDocument* en *DocumentAttachment*, voor dit control. Dan wordt de implementatie van de navigatie functie besproken.

De Navigatie Control zorgt ervoor dat de images allemaal in een structuur worden gezet. Door de control kan de eindgebruiker navigeren tussen de images in de gemaakte structuur. De structuuropbouw wordt mogelijk gemaakt door twee klassen: *DocumentAttachment* en *MyDocument*.

Een document kan een aantal pagina's (images) en attachments hebben. *DocumentAttachment* representeert zo'n attachment. In deze klasse zit een array waarin de images kunnen worden opgeslagen. De benodigde properties (getters en setters) zijn hierin

geschreven. De klasse `MyDocument` representeert een document. De attachments van een document worden in `MyDocument` ook in een array geplaatst. Ook hier zijn een aantal properties geschreven. De pagina's van zo'n document worden voor het gemak geplaatst in een attachment en wel de eerste.

De navigatie control wordt gebouwd uit de *Viewer control* en een *Toolstrip* daaronder. De toolstrip bezit alle knoppen die de navigatie besturen. Zoals gezegd in de implementatie beschrijving van de *user interface* worden die knoppen als nodig verborgen of disabled. De klasse zelf bevat drie waarden die bijhouden welk image uit welke attachment uit welk document getoond is. Dit is te zien in *figuur 23*.

```
private int mCurrentDocument;  
//the document index of the document that contains the current page that is shown  
private int mCurrentAttachment;  
//the attachment index of the attachment that contains the current page that is  
//shown  
private int mCurrentPage;  
//the page index of the current page that is shown
```

Figuur 23: Members die getoonde image bijhouden

Bij het navigeren wordt aan de hand van de gewenste navigatie mogelijkheid een of meerdere members met één verhoogt of verlaagt. Een voorbeeld hiervan is te zien in *figuur 24*.

```
/// <summary>  
/// set the page, attachment and document indexes to next page in mDocuments  
/// </summary>  
private void IncrementCurrentPage()  
{  
    if (mDocuments != null)  
    {  
        //if page already is last page of the attachment it belongs to  
        if (this.InLastPageOfCurrentAttachment())  
        {  
            //increment attachment index and set page index to zero  
            bool updatePage = true;  
            this.IncrementCurrentAttachment(updatePage);  
        }  
        else  
        {  
            //increment page index  
            this.mCurrentPage = this.mCurrentPage + 1;  
            this.ShowImage();  
        }  
    }  
}
```

Figuur 24: Increment huidige pagina

Bij het verhogen of verlagen van deze waarden wordt de methode `ShowImage` aangeroepen. Die doet niets anders dan aan de hand van de current document, attachment en page indexes de juiste image geeft aan de viewer control, die de image dan toont.

De converter control bevat ook een rij van documenten. Deze rij kan gezien worden als een *map* met een aantal documenten. Aan deze map kunnen documenten worden toegevoegd. Als dit gebeurt wordt eerst gekeken als de toe te voegen document als in de map aanwezig is. Als dit het geval is, wordt dat document, de eerste pagina daarvan, getoond, anders eerst toegevoegd aan de map en dan getoond. Dit is te zien in *figuur 25*.

```
public void AddDocument(MyDocument document)
{
    if (document != null)
    {
        //create a new instance of myDocument[] if not initialized
        if (this.mDocuments == null)
        {
            this.mDocuments = new MyDocument[] { document };
        }
        else
        {
            this.AddToMDocuments(document);
        }
    }
}

/// <summary>
/// Adds a document to the array of documents of this class if the document isn't
/// already in the array
/// If the array contains the document, just show it.
/// </summary>
/// <param name="document"></param>
private void AddToMDocuments(MyDocument document)
{
    int index = this.ContainsDocument(document);
    // add only if the array doesn't already contain the document
    if (index == -1)
    {
        this.AddMDocument(document);
        //update the properties forms if the form is shown
        if (!(this.propertiesForm == null || this.propertiesForm.IsDisposed))
            this.propertiesForm.ConstructTreeView(this.mDocuments);
    }
    else
    {
        //the document already exists in the array so show it
        this.mCurrentDocument = index;
        this.mCurrentAttachment = 0;
        this.mCurrentPage = 0;

        this.ShowImage();
    }
}

/// <summary>
/// Adds a document to the end of the documents array
/// </summary>
/// <param name="document"></param>
private void AddMDocument(MyDocument document)
{
    //add the new document to the end of the array
    MyDocument[] temp = new MyDocument[this.mDocuments.Length + 1];
    for (int i = 0; i < this.mDocuments.Length; i++)
    {
        temp.SetValue(this.mDocuments[i], i);
    }
    temp.SetValue(document, this.mDocuments.Length);

    this.mDocuments = temp;
    //check if the allowed number of documents isn't exceeded
    this.CheckNumberOfDocuments();
    // show the newly added document
    this.mCurrentDocument = this.mDocuments.Length - 1;
    this.mCurrentAttachment = 0;
    this.mCurrentPage = 0;

    this.ShowImage();
}
```

Figuur 25: Add document

Als toegevoegde functionaliteit is een limiet aan de rij van documenten gevoegd. Deze waarde zorgt ervoor dat de rij met documenten niet al te groot kan worden. Bij het toevoegen of vervangen van de document array wordt gechecked of de lengte van de nieuwe rij voldoet aan de gestelde limiet. Als dat niet het geval is worden de eerste documenten uit de rij verwijderd tot aan de limiet voldaan wordt. De implementatie hiervan is te zien in *figuur 26*.

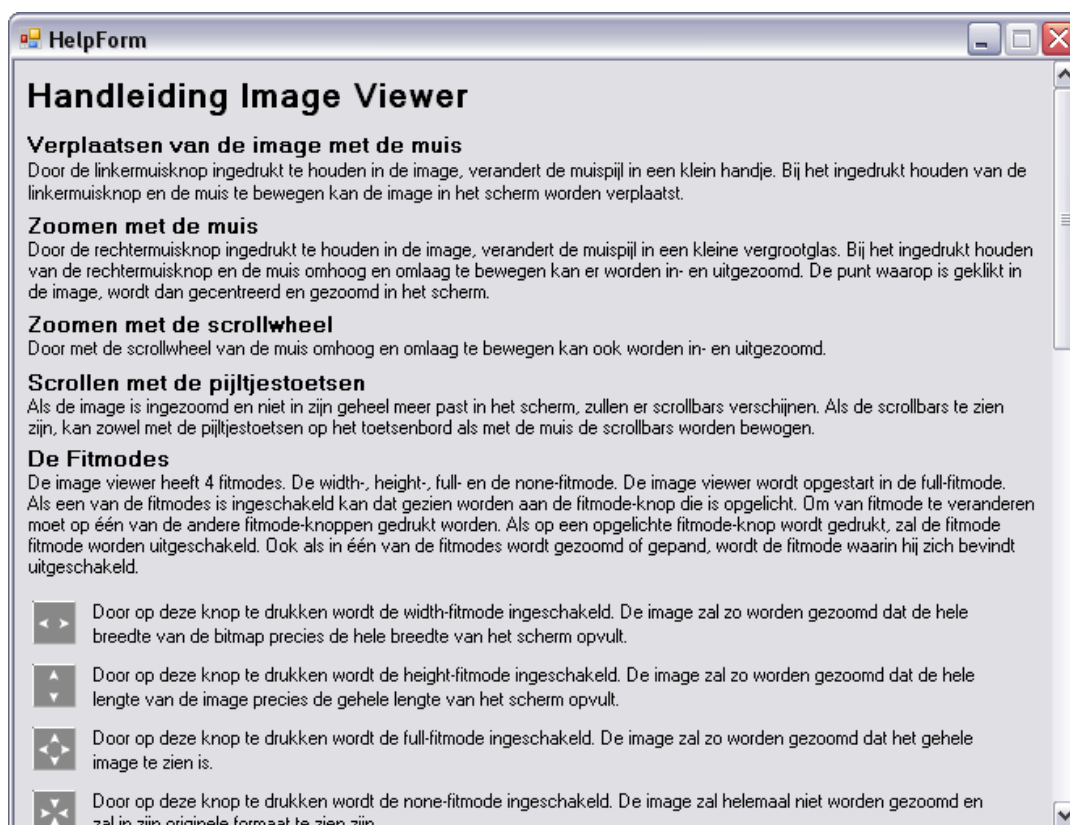
```
/// <summary>
/// Check if documents need to be removed to comply with the
/// maximum number of documents
/// </summary>
private void CheckNumberOfDocuments()
{
    int numberOfDocuments = this.mDocuments.Length;
    if (numberOfDocuments > this.maxNumberOfDocuments)
    {
        this.RemoveExtraDocuments();
    }
}

/// <summary>
/// removes any extra documents, so the number of documents equals
/// the allowed number. documents at the beginning of the array are
/// removed
/// </summary>
private void RemoveExtraDocuments()
{
    MyDocument[] myDocumentsTemp =
        new MyDocument[this.maxNumberOfDocuments];

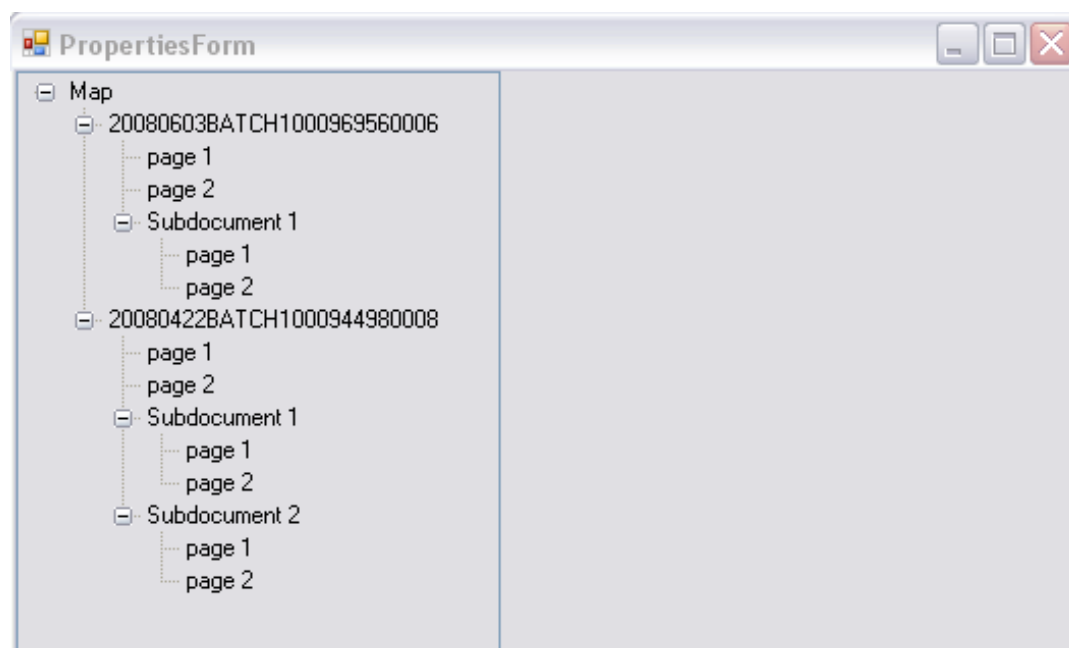
    for (int i = 0; i < this.maxNumberOfDocuments; i++)
    {
        myDocumentsTemp.SetValue(this.mDocuments.GetValue
                                (this.mDocuments.Length - i - 1),
                                myDocumentsTemp.Length - i - 1);
    }
    this.mDocuments = myDocumentsTemp;
}
```

Figuur 26: Remove extra documents

Aan dit control zijn ook een *HelpForm* en een *PropertiesForm* toegevoegd. De help form bestaat uit een handleiding voor zowel de *viewer* als *navigatie* control en de properties form laat de structuur van de huidige map zien. Printscreens van de help form en de properties form zijn respectievelijk te vinden in *figuur 27* en *figuur 28*.



Figuur 27: Help form

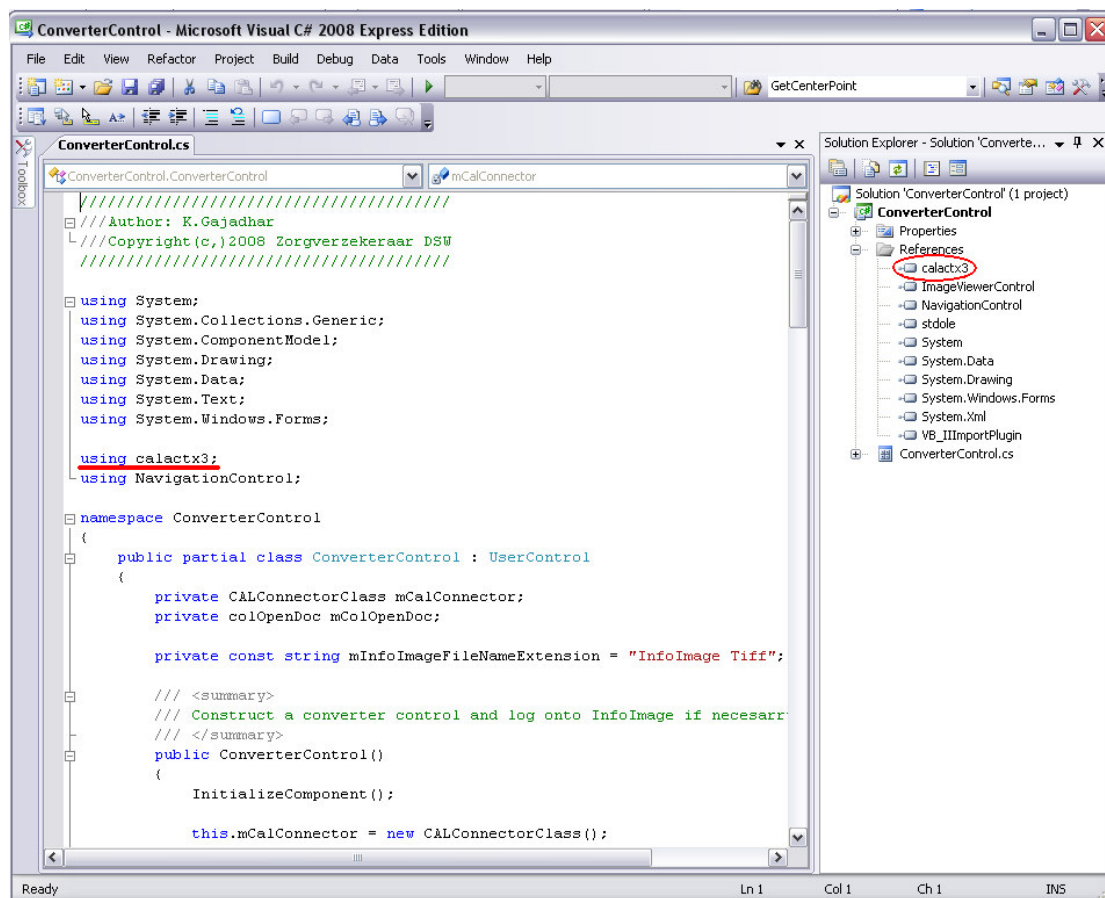


Figuur 28: Properties form

8.4 Converter Control

Zoals eerder gezegd zijn wij binnen dit project niet verder gekomen dan deze control. Deze is zelfs ook niet afgerond en getest. In deze paragraaf wordt gespecificeerd wat wel geïmplementeerd is.

Zoals eerder gezegd wordt om een tekort aan implementatie tijd gebruikte gemaakt van een *ActiveX* control. Dit control wordt net als elk ander control binnen de .Net omgeving geïmporteerd. Eerst wordt een het control als *Referentie* toegevoegd aan het project. Vervolgens kan het control in elke klasse gebruikt worden middels de *using* statement. Dit is aangegeven in *figuur 29*.



Figuur 29: Import en gebruik van CalactX

Betreffende de implementatie zijn wij niet verder gevorderd dan het ophalen en tonen van image bestanden. Die zijn als volgt geïmplementeerd.

De converter control krijgt *InfoImageId's* als invoer. Met zo'n id kan een *InfoImage* document, geplaatst in de klasse *clsOpenDoc*, opgehaald worden. Uit deze klasse worden dan de image bestanden opgeslagen in een tijdelijke map op de harde schijf. Bij het opslaan van de bestanden wordt het pad naar de map terug gegeven. De image kan dan opgehaald en geplaatst worden in een *MyDocument* object. Dit object wordt uiteindelijk als invoer gegeven aan de *navigatie* control, welke het document met navigatie mogelijkheden toont via de *viewer* control.

8.5 Tests

In deze paragraaf wordt de implementatie van de tests besproken, gevolgd door de test en coverage resultaten.

Zoals in het *testplan* gezegd, zou getest worden met zowel automatische als handmatige tests. Automatische tests werden uitgevoerd in een test project. De test klassen in zo'n project zijn gegenereerd door *Visual Studio*. Het enige wat verder nog gedaan moest worden is het specificeren van de test waarden. In *figuur 30* is een complete test methode voor het in zoomen uit de *Functions* test klasse te zien.

```
/// <summary>
///A test for ZoomIn (int)
///</summary>
[DeploymentItem("ImageViewerControl.dll")]
[TestMethod()]
public void ZoomInTest()
{
    ZoomingPictureBox picBox = new ZoomingPictureBox();

    object target = ImageViewerControlTest.ImageViewerControl_FunctionsAccessor
        .CreatePrivate(picBox);

    ImageViewerControlTest.ImageViewerControl_FunctionsAccessor accessor =
        new ImageViewerControlTest.ImageViewerControl_FunctionsAccessor(target);

    int z = 50;
    //Note: picbox has no image!
    accessor.ZoomIn(z);
    //picbox has no Image yet, so the zoom value stays default (1000 -> 100%)
    Assert.AreEqual(1000, picBox.Zoom);

    picBox.Image = new Bitmap(500, 500);

    accessor.ZoomIn(z);
    int expected = picBox.Zoom - z;
    int actual = picBox.Zoom;
    Assert.AreEqual(expected, actual);

    // can't be zoom in further than minimal zoom factor
    accessor.ZoomIn(2000000);
    expected = accessor.MinZoomFactor;
    actual = picBox.Zoom;

    Assert.AreEqual(expected, actual);
}
```

Figuur 30: ZoomInTest

Voor de handmatige tests werd voor elk van de control een *test applicatie* gemaakt. Hierbij werd het control in een windows *form* geplaatst. Daarmee kon het control als een applicatie gestart en uitvoerig visueel getest worden.

Na afronding van een control werd de test applicatie van het control aan diverse ontwikkelaars en gebruikers aangeboden om de functionaliteiten te testen. Daaruit voortvloeiende opmerkingen en aanbevelingen werden geëvalueerd en aan de hand daarvan alsnog geïmplementeerd of verbeterd.

Test resultaten

Bij het opleveren van de uiteindelijke source code slaagden alle uitgevoerde tests. Voor een beschrijving van de test cases wordt verwezen naar het *Acceptatie & Test plan* in bijlage J.

Om de kwaliteit van de automatische tests te bepalen werd de *code coverage* tijdens het uitvoeren van de tests gemeten. Een 100% coverage is voor deze tests onmogelijk, dit omdat specifiek de functies getest worden. In *figuur 31* en *32* staat de coverage voor de tests van respectievelijk de viewer en navigatie control.

Hierarchy	Not Covered (Blocks)	Not Covered (% Blocks)	Covered (Blocks)	Covered (% Blocks)
Kgajadhar@PCD1284 2008-09-30 18:35:14	319	35,37 %	583	64,63 %
ImageViewerControl.dll	319	35,37 %	583	64,63 %
{ } ImageViewerControl	306	34,42 %	583	65,58 %
+ Functions	26	13,40 %	168	86,60 %
+ ViewerControl	258	51,39 %	244	48,61 %
+ ViewerControl.MyCursor	0	0,00 %	7	100,00 %
+ ZoomingPictureBox	22	11,83 %	164	88,17 %
+ { } ImageViewerControl.Properties	13	100,00 %	0	0,00 %

Figuur 31: Coverage viewer control

Zoals *figuur 31* aangeeft, werd een gemiddelde coverage van 64% bereikt voor de viewer control. De klassen *functions* en *ZoomingPictureBox* werden voor ruim 80% gecoverd, waar de coverage voor de *ViewerControl* bleef haken op 48%. Dit omdat de afhandeling van *events* werden genegeerd in de tests.

Hierarchy	Not Covered (Blocks)	Not Covered (% Blocks)	Covered (Blocks)	Covered (% Blocks)
Kgajadhar@PCD1284 2008-09-30 19:38:29	1814	69,10 %	811	30,90 %
DocumentControl.dll	1261	73,19 %	462	26,81 %
{ } NavigationControl	1248	72,98 %	462	27,02 %
+ DocumentAttachment	5	45,45 %	6	54,55 %
+ HelpForm	992	100,00 %	0	0,00 %
+ MyDocument	13	28,26 %	33	71,74 %
+ NavigationControl	152	26,43 %	423	73,57 %
+ PropertiesForm	86	100,00 %	0	0,00 %
+ { } NavigationControl.Properties	13	100,00 %	0	0,00 %
+ ImageViewerControl.dll	553	61,31 %	349	38,69 %

Figuur 32: Coverage navigatie control

Voor de navigatie control werd een gemiddelde coverage van 27% gehaald. Dit omdat ook voor dit control alleen de toegevoegde functionaliteit uitvoerig werd getest.

9 Conclusie en Reflecties

In dit hoofdstuk zullen wij onze conclusies en aanbevelingen presenteren. In paragraaf 9.1 zullen we de conclusies trekken die betrekking hebben op het project en het proces en in paragraaf 9.2 zal een reflectie en verantwoording per project lid gegeven worden.

9.1 Conclusies

Dit project werd gestart met de bedoeling een image viewer control te ontwikkelen voor de Zorgverzekeraar DSW. De functionaliteit van dit control werd verdeeld onder vier verschillende controls namelijk de *viewer*, *navigatie*, *converter* en *search* control. De eerste control zou het bewerken, zoals zoomen, roteren, enz, van images afhandelen. De tweede zou zich bezig houden met het navigeren tussen images behorende bij een document structuur. De derde control zou documenten ophalen en omzetten naar een gewenste structuur en de vierde control zou de gebruiker in staat stellen documenten te zoeken.

Als resultaat van dit project werd tenminste een compleet werkende *viewer* control verwacht. Met de andere controls zou gestart worden aan de hand van beschikbare tijd.

Het ontwikkelen van elk van de controls werd benaderd als een individueel project. Elk control vereiste daarom een eigen analyse, ontwerp en implementatie.

Als uiteindelijke resultaat werd de *viewer*, *navigatie* en een deels geïmplementeerde *converter* control opgeleverd. Dit project kan daarom een **succes** genoemd worden.

Communicatie rond dit project verliep over het algemeen uitstekend. Het project werd gedaan te DSW, dus waren de werkgever en gebruikers altijd en gemakkelijk te benaderen. Het project groep bestond uit twee personen, dus werd de communicatie binnen de groep eenvoudig geregeld.

Voor de project leden was dit project een uitstekende ervaring. Er werd kennis gemaakt met de interne software ontwikkeling in het bedrijfsleven en niet te vergeten een nieuwe programmeer taal en omgeving. Het tonen en bewerken van images was voor beide leden ook iets nieuws, daarover werd heel veel bijgeleerd.

9.2 Reflectie en verantwoording

Reflectie Abdoelrasheed Jarmohamed

Gedurende 13 weken heb ik samen met Kishen gewerkt aan het Bachelorproject, dat onderdeel is van de Bachelorprogramma van de opleiding Technische Informatica op de TU Delft. Voor het Bachelorproject hebben we een Image Viewer Control gebouwd. Dat is gedaan in opdracht van DSW Zorgverzekeraar. Een voorwaarde van DSW was dat we tijdens het gehele project aanwezig moeten zijn op het kantoor van DSW. Dankzij de goede werksfeer was het een prettige ervaring. Ook is het een fijne afwisseling om voor een aantal weken naar kantoor te gaan in plaats van naar de faculteit.

Wat ik vooral heb geleerd is dat in het bedrijfsleven een opdracht altijd kan veranderen. Er kunnen natuurlijk harde afspraken worden gemaakt. Maar als zelfs bij de opdrachtgever bepaalde zaken onbekend zijn, is het moeilijk om zulke afspraken te maken.

Maar door de veranderingen in het project heb ik geleerd dat het heel belangrijk is om veel te communiceren wat betreft de opdracht zelf en wat de opdrachtgever precies wil. Wat ik ook

heb gezien is dat de wensen van de opdrachtgever en de wensen van de eindgebruikers kunnen verschillen. Het was dan aan ons en de opdrachtgever om een aantal wijzigingen aan te brengen in de oorspronkelijke opdracht.

Doordat de projectgroep maar uit twee leden bestond is de communicatie altijd goed verlopen. De communicatie met de opdrachtgever ging ook gemakkelijk, omdat we samen op de zelfde afdeling werkten. We konden op elk moment iemand aanspreken als er iets onduidelijk was of als we iets nodig hadden. Vooral daarom was erg handig dat we elke dag op kantoor waren.

De opdracht moest worden ontwikkeld met het .NET-framework. En omdat geen van ons hier eerder mee had gewerkt, moest we eerst kennis opdoen over dit framework. Gelukkig hebben we dit snel kunnen doen en konden we hierdoor ook snel beginnen met implementeren.

De documentatie voor dit project was een zware opgave. Dit had vooral te maken doordat de opdracht in vieren was verdeeld. En voor elke deel moest er opnieuw een RAD en een SSD geschreven worden. En als je dan maar met twee leden bent, wenste je weleens dat je er nog een projectlid was.

Ik heb aan twee van de drie controls gewerkt, die zijn ontwikkeld. Dat waren de Viewer Control en de Navigatie Control. Voor de Navigatie Control en de documentatie was ik de eindverantwoordelijke.

Reflectie Kishen Gajadhar

Vanaf 1 juli 2008 tot 30 september 2008 heb ik samen met Rashied gewerkt aan ons bachelor project. Dit project heeft in totaal 13 weken geduurd en heeft plaats gevonden op het kantoor van de zorgverzekeraar DSW te schiedam. Daar moesten wij per week een totaal van 32 uur doorbrengen.

Voor mij heeft dit project een eerste kennismaking met de software ontwikkeling in het bedrijfsleven betekend. Het project verliep niet precies zoals wij dat deden op de TU. Over het uiteindelijke resultaat, de *image viewer control*, was in eerste instantie heel wat anders verwacht. Na goede interactie tussen ons, de werkgever en andere software ontwikkelaars over verschillende mogelijkheden, werden uiteindelijk de juiste requirements achterhaald en verwerkt.

Communicatie binnen de groep verliep eenvoudig en daardoor uitstekend. Wij werkten met zijn tweeën aan het project dus communiceerden wij gemakkelijk middels mail, telefoon en vaak persoonlijk.

De implementatie moest gebeuren in C#, voor ons beide een toen nog onbekende programmeer taal. Na de eerste kennismaking met die taal werd duidelijk dat het niet veel verschilt met voor ons een bekende taal, namelijk Java. Na het doornemen wat voorbeeld code werd de werking met C# voor mij duidelijk en kon relatief gemakkelijk geïmplementeerd worden.

Mijn verantwoordelijkheden binnen dit project waren de *viewer* en *converter* control. Ook het testen viel voor een groot deel onder mijn verantwoording. Uiteindelijk hebben Rashied en ik, los van de *converter* control, samengewerkt aan elk van de controls. Zodoende konden wij middels voldoende discussie tot een goed resultaat komen.

Referenties

- [1] DSW, Personeelshandboek
- [2] Lethbridge, T.C., Laganière, R. (2005), “Object-oriented software engineering”, 2nd edition, New York, McGraw-Hill Education
- [3] Mayhew, D.J. (1992), “Principles and guidelines in software user interface design”, Prentice Hall
- [4] Pezze, M., Young, M. (2007), “Software testing and analysis”, USA, Wiley

Bezochte websites

- www.codeproject.com
- www.c-sharpcorner.com
- www.csharphelp.com
- www.msdn.microsoft.com

De bovenstaande websites zijn bezocht voor c# voorbeelden en algorithmen. De websites bevatten heel wat voorbeeld projecten over heel wat onderwerpen. Ook zijn de forums behorende bij de sites geraadpleegd. Zo konden heel wat ideeën bedacht worden.

Bijlage A: Opdrachts omschrijving

Stage opdracht

Job Scheffers
.Net Systeemontwikkeling
DSW Zorgverzekeraar

Achtergrond:

Binnen DSW wordt momenteel een custom-made applicatie gebruikt om gescande documenten te bekijken. Met deze toepassing kunnen administratieve medewerkers de via post ontvangen documenten voor zich halen om vervolgens de betreffende informatie in een back-end applicatie in te voeren en met die informatie eventuele vervolg handelingen uit te voeren. De huidige applicatie is een stand-alone applicatie, waardoor het moeilijk is voor andere applicaties om aan te sturen welk plaatje getoond moet worden. Een voorbeeld hiervan is de workflow applicatie die bepaalt welk dossier de medewerker moet behandelen en dus welke gescande brief/brieven de imageviewer moet tonen. De aansturing van de image viewer loopt nu via een ActiveX control dat onderdeel is van de image viewer. De medewerkers moeten zelf hun applicatie en de image viewer zo size, dat beiden op het beeldscherm passen. De ontwikkelaar van een applicatie heeft geen enkele invloed op wanneer en hoe de image viewer getoond wordt.

Opdracht:

Er is behoefte om de functionaliteit van de besproken image viewer in een usercontrol te plaatsen zodat deze functionaliteit in diverse applicaties gebruikt kan worden. De ontwikkelaar kan dan het tonen van het gescande document een integraal onderdeel van de UI maken. Gevraagd wordt naar een image viewer control, in .Net omgeving, waarbij de documenten, bitmap images, *efficient* vergroot, verkleind, geroteerd, enz. kunnen worden.

Op basis van dit usercontrol wordt een tweede usercontrol gevraagd, dat als invoer een array van bitmap images ontvangt die verkleind in een rij worden getoond en waarbij een geselecteerde image in het eerste usercontrol wordt weergegeven.

Vervolgens is er nog behoefte aan een usercontrol dat kennis heeft van de bij DSW gebruikte image database en de bijbehorende kenmerken database. Dit derde control voegt een zoekfunctie toe. Daarbij wordt de images database doorzocht en de resultaten weergegeven in een array, zoals beschreven bij het tweede usercontrol.

Bijlage B: Research verslag

BSc-project IN3405



Voor onderzoek

Omgeving:

Zorgverzekeraar DSW
Technische Universiteit Delft
Faculteit EWI Technische Informatica
Variant Softwaretechnologie (ST)

Projectgroep:

Abdoelrasheed Jarmohamed
Kishen Gajadhar

Commissie:

Job Scheffers (Zorgverzekeraar DSW, begeleider)
B.R. Sodoyer (TU Delft, coördinator / begeleider)

Datum: 11 juli 2008

Status: Final

Versie: 2.0

©Juli 2008 Zorgverzekeraar DSW

Alle in dit verslag afgebeelde gegevens, waaronder teksten, foto's, illustraties, grafische materiaal, (handels)namen, logo's, waren- en dienstmerken, zijn in eigendom van zorgverzekeraar DSW en worden beschermd door auteursrecht, merkenrecht en/of enig ander intellectueel eigendomsrecht.

Inhoudsopgave

1. Introductie	77
2. Custom vs. COTS software	50
2.1 Aanschaf en geschatte kosten.....	50
2.2 Implementatie en integratie	51
2.3 Onderhoud	52
2.4 Opdracht van DSW.....	52
3. C# vs. Java	54
4. Conclusie.....	57
Literatuurlijst	58

1. Introductie

Voor u ligt het rapport van het vooronderzoek van het *image viewer control* project. Dit rapport is geschreven in het kader het bovengenoemde bachelorproject welke een onderdeel is van het bachelorcurriculum van de opleiding technische informatica van de Technische Universiteit Delft. Het doel van dit rapport is om de bevindingen van het onderzoek duidelijk te maken. Het onderzoek dient als voorbereiding op het te volgen project, de onderzoeksonderwerpen mogen daarbij zelf worden gekozen.

Bij de opdrachtgever, Zorgverzekeraar DSW, worden de software systemen zelf ontwikkeld. Zo is als stage opdracht om een *custom image viewer control* gevraagd. Maar van dit stukje software zijn er een groot aantal soortgelijke en commerciële applicaties. De volgende vraag ontstond: “*Is het juist dat DSW vraagt om een custom image viewer, waar er een groot aanbod van commerciële Off-The-Shelf image viewers bestaat?*”

Om deze vraag te beantwoorden wordt eerst gekeken naar de voor- en nadelen van custom tegenover COTS software en de mogelijke voorwaarden voor een bepaalde keus. Daarna worden de bevindingen van dat onderzoek in context gebracht met de situatie te DSW.

Ook worden in dit onderzoek de twee programmeertalen, C# en Java, met elkaar vergeleken. Er wordt gekeken naar de overeenkomsten en verschillen van de twee talen. Een vergelijking met twee talen heeft meestal alleen zin als die twee erg veel op elkaar lijken. Als dat niet het geval is, is het beter om ze afzonderlijk van elkaar te onderzoeken. In dit geval lijken de twee talen erg veel op elkaar en is het zinvol om ze te vergelijken. Zeker als je één van de talen al beheerst. Tijdens de opleiding komt de programmeertaal Java uitgebreid aan bod, in tegenstelling tot C#. De opdracht die DSW ons heeft gegeven moet worden gedaan met de taal C#. Daarom zullen we als aanvulling tot het eerste onderzoek ook de twee programmeertalen met elkaar vergelijken.

Dit zal gedaan worden door verschillende aspecten van de twee talen naast elkaar te zetten en die te omschrijven. Daarbij wordt gedacht aan programmeeromgeving, manier van compileren en natuurlijk de syntaxis en werking.

De opbouw van dit rapport is als volgt. In hoofdstuk 2 wordt alle aandacht besteed aan de eerste onderzoeksvraag, waar gelijk een antwoord op gezocht zal worden. In hoofdstuk 3 volgt de vergelijking tussen de programmeertalen C# en Java. Het rapport wordt afgesloten met een conclusie in hoofdstuk 4.

2. Custom vs. COTS software

Het aanschaffen van software voor bedrijven vereist vele beslissingen. De meest belangrijke beslissing is wellicht de keus tussen generic, “Commercial Off-The-Shelf”, of custom software.

Custom software is ontwikkeld om de specifieke eisen van een gebruiker te vervullen waarbij het in de meeste gevallen onbruikbaar is voor andere gebruikers. Deze software wordt door ontwikkelaars in dienst van de gebruiker ontwikkeld. Custom software biedt, in vergelijking met andere software types, de meeste flexibiliteit waardoor bestaande bedrijfsprocessen nauwkeurig nagebootst kunnen worden. Dit type software is bedoeld voor een klein aantal gebruikers, meestal degene die de software heeft laten ontwikkelen. Het succes ervan hangt af van de vervulling van hun eisen en wensen.

Generic of COTS software is ontwikkeld om verkocht te worden via de open markt, om functies te bieden die vele gebruikers nodig hebben en om te draaien op algemene platformen en machines. De eisen voor zulk software worden meestal achterhaald door marktonderzoek. Generic software bezit over het algemeen goed onderzochte en doordachte functies die over een lange periode zijn ontwikkeld. Dit type software is ontworpen, door meestal erkende software verkopers, om de beste oplossingen in een bepaald gebied en goede ondersteuning voor die oplossingen te leveren. COTS software producten omvatten een beschrijving of een definitie van de functies, documentatie behorende bij die functies, en een beschrijving van de vereiste systeemeisen[1].

Bij de zorgverzekeraar DSW is er behoefte aan een *image viewer control* voor het bekijken van gescande documenten. Het ontwikkelen van deze control is als stage opdracht gegeven aan de auteurs. Maar voor en dergelijke image viewer bestaat er heel wat generic software. In dit rapport wordt daarom naar het verschil, de voor- en nadelen van custom en generic software gekeken, waarna de bevindingen in context met de situatie bij DSW gebracht zullen worden.

In dit rapport worden beide typen software vergeleken in drie verschillende opzichten: Aanschaf en de daarbij verbonden kosten, implementatie en integratie, en onderhoud. Daarna volgt een opsomming van de requirements voor het gevraagde stukje software en een conclusie waaruit zal blijken als de beslissing van DSW om de gewenste control zelf te laten ontwikkelen de juiste is.

2.1 Aanschaf en geschatte kosten

De aanschaf van een nieuw software systeem komt met hoge risico's aan verbonden. Om deze risico's te minimaliseren, moeten bedrijven enkele belangrijke factoren in acht nemen bij het kiezen van een software oplossing die het best bij hun past. Twee van de factoren die in acht genomen moeten worden tijdens de aanschaf zijn het vergaren van *business requirements* en een schatting van de *Total Cost of Ownership*[2]. Tussen een custom en COTS software systeem zijn de uitkomsten van beide factoren erg verschillend.

Bij het aanschaffen van een nieuw software systeem is het vergaren van de requirements een van de eerste taken. Over het algemeen kunnen deze requirements verdeeld worden in vier groepen, volgens het MoSCoW model (*Must have*, *Should have*, *Could have* en *Want to have*). Een COTS systeem zal meer moeite hebben met het vervullen van de requirements, immers die bezitten alleen functies bedoeld voor een groot aantal gebruikers en geen specifieke functies voor een bepaalde gebruiker. Een custom systeem, aan de andere kant,

wordt van de grond uit gebouwd. Dit kan zodanig gedaan worden om te voldoen aan alle requirements uit de eerste drie groepen. Requirements uit de laatste groep worden eventueel aan het eind van het project behandeld. Een custom systeem kan daarom de bestaande bedrijfsprocessen zo precies mogelijk imiteren.

Dit betekent dat COTS systemen in de meeste gevallen alleen de *must* requirements kunnen vervullen[3]. Het bedrijf zou zeer flexibel moeten omgaan met de andere requirements. Aan de andere kant wordt een custom systeem zodanig ontworpen om aan alle requirements te voldoen. De enige beperkingen vloeien voort uit technische en creatieve beperkingen van de ontwikkelaars. De keuze tussen een COTS of custom systeem komt hier neer op de bereidheid van het bedrijf om requirements te laten vervallen.

Het schatten van de kosten voor een COTS systeem gaat iets makkelijker dan die voor een custom systeem[3]. Over het algemeen hebben de COTS systemen vaste prijzen die tot stand komen als de verkopers aan de hand van vorige ervaringen een goede schatting van ontwikkelingskosten maken. De schatting bij custom systemen kunnen heel erg verschillend zijn. Dit is volledig afhankelijk van de implementatie van het systeem. Het flexibele van een custom systeem kan namelijk leiden tot een situatie waar veel te veel tijd besteed wordt aan *want to have* requirements. Deze requirements zouden na toevoeging aan het systeem weinig vooruitgang voor het gehele systeem betekenen. Als custom systemen groeien en zich steeds aanpassen aan veranderende business requirements, wordt de kans groter dat geplande budgetten een grote onderschatting blijken te zijn.

2.2 Implementatie en integratie

Als een nieuw systeem gekozen is, komen weer andere belangrijke factoren naar boven. Deze zijn onder andere *Training requirements*, *Business process integration* en de *benodigde tijd* voor implementatie en aannee. Risico- en kostenanalyse zijn factoren van elk van deze elementen[2].

Bij het implementeren van een nieuwe software systeem zijn er een aantal training requirements. Enkele zijn: trainingen voor eindgebruikers, trainingmateriaal en handleidingen. Bij custom systemen zouden deze zelf geschreven en onderhouden moeten worden. De trainingen zouden ook zelf ontwikkeld moeten worden, hoewel de lengte van de trainingen relatief kort zullen zijn. Dit omdat het systeem zodanig ontworpen zal zijn om zo dicht mogelijk bij de huidige bedrijfsprocessen te passen. Erkende COTS systemen hebben over het algemeen een aantal gepubliceerde artikelen waarnaar gerefereerd kan worden. Ook is er meestal goede training en ondersteuning beschikbaar. Dit reduceert het risico van het zelf ontwikkelen van een goed en effectief trainingsplan dat het bedrijf ten goede komt. Aan de andere kant zijn de door verkopers ontwikkelde trainingen vaak veel te algemeen[3]. Gebruikers zullen de algemene training moeten toepassen op hun business scenario, waardoor het trainingsproces vertraging kan oplopen.

Een belangrijke factor in elk groot nieuw softwareproject is de integratie met bestaande bedrijfsprocessen. Het integreren van grote COTS systemen vereist bijna altijd wat aanpassingen aan huidige processen[4]. De grote van die aanpassingen hangt af van het nieuwe software systeem. Custom software systemen hebben dat probleem niet. Die zijn van scratch uit opgebouwd om de bestaande processen te imiteren. Dit betekent dat er voor COTS systemen een groter risico bestaat dat het systeem niet succesvol in het bedrijf geïntegreerd kan worden .

De benodigde tijd voor implementatie en aanneming van een nieuw software systeem is een groot en belangrijk element van de projectkosten als geheel[4]. Custom software wordt van scratch uit ontworpen en geïmplementeerd, waardoor veel onderzoek-, ontwerp- en implementatietijd vereist is. Afhankelijk van het business scenario kunnen, maar hoeven er geen mijlpalen voor handige deliverables bestaan. Dit betekent dat *return on investment* over het algemeen pas aan het eind van het project gerealiseerd kan worden[5]. Het ontwerpen van grote en complexe scenario's kunnen maanden lang duren en de verfijning zelfs jaren. Aan de andere kant zijn COTS systemen complete systemen, klaar voor aanneming en gebruik volgens een bepaalde startconfiguratie. Dit duidelijke voordeel van COTS systemen wordt gecompenseerd door de kosten voor het bedrijf om zich aan te passen aan een compleet nieuw systeem.

2.3 Onderhoud

Als het systeem al compleet geïntegreerd en werkend is binnen het bedrijf, wordt alle aandacht gericht op onderhoud en updates[2]. Belangrijke factoren daarbij zijn het *aanpassen* van het systeem aan de hand van veranderende businessprocessen en structuur, het *toevoegen* van nieuwe producteigenschappen en het *oplossen* van softwarefouten.

COTS systemen zijn vaak gelimiteerd betreffende de flexibiliteit om veranderingen in de bedrijfsstructuur en processen af te handelen. Grote bedrijven zouden invloed kunnen uitoefenen op software verkopers en het ontwikkelproces van het COTS systeem om zodanig de gewenste aanpassingen te verkrijgen. Maar ontwikkelaars van COTS software zullen geen resources verspillen aan aanpassingen of het toevoegen van nieuwe functionaliteiten aan het systeem als het geen verbetering is voor alle gebruikers van hun software. Om de gewenste aanpassingen toch door te voeren kunnen bedrijven de aanpassingen zelf doen[5]. Dit brengt wel hogere kosten en een kleinere kans op compatibiliteit van het COTS systeem met toekomstige updates.

Bij custom systemen kunnen veranderingen snel aangebracht en ontworpen worden. Het ontwerp komt ook heel erg overeen met de business requirements. Daardoor hebben custom systemen grote flexibiliteit. Maar als nadeel kan net als bij de implementatie de nadruk vallen op *want to have* requirements, waardoor extra kosten ontstaan.

Als software fouten ontdekt worden en die opgelost moeten worden, is de prioriteit en snelheid van de reactie afhankelijk van de service overeenkomst getekend met de verkoper en de aard van de fout zelf. Een COTS systeem zal gewoonlijk goede ondersteuningsmogelijkheden bevatten, hoewel de responsetijd nog afhankelijk is van de aard van de fout en de beschikbaarheid van software ontwikkelaars en specialisten. Een custom systeem kan door het bedrijf zelf gerepareerd worden, maar die systemen hebben een groter risico omdat ondersteuningsmogelijkheden door het bedrijf gerealiseerd moeten worden. Dit betekent dat de kans op een succesvolle reparatie kleiner is voor een COTS systeem omdat fouten misschien niet binnen een acceptabele tijdspan opgelost kunnen worden, waar bij custom systemen de potentiële kosten hoger liggen aangezien ontwikkelings- en reparatiekosten door het bedrijf zelf gedekt moeten worden.

2.4 Opdracht van DSW

Zoals eerder vermeld zijn de auteurs van dit document aangenomen door de Zorgverzekeraar DSW om een custom *image viewer control* voor hen te ontwikkelen. De functionele requirements van de control werden ook door DSW gespecificeerd. Deze requirements

ingedeeld volgens het welbekende MoSCoW model (zie tabel 1). Met de gemodelleerde requirements in hand is op zoek gegaan naar een COTS of Open Source control dat in een user of custom control geplaatst kan worden.

Requirements	Prioriteit
Toon een gegeven bitmap	<i>Must</i>
Fit-modes: full, height, width & none	<i>Must</i>
Zoom: factor 10 – 1800%	<i>Must</i>
Rotatie: +90° & -90°	<i>Must</i>
Invert colors	<i>Must</i>
Scrollbars (alleen als nodig)	<i>Must</i>
Op selectie inzoomen	<i>Must</i>
Bicubic interpolation mode	<i>Must</i>
Mouse wheel voor zoom functie	<i>Should</i>
Pijltjes toetsen voor scrollen	<i>Should</i>
PgUp & PgDn toetsen voor verticaal scrollen	<i>Should</i>
Toon van een gegeven array van bitmaps een selecteerde	<i>Could</i>
Search functie	<i>Could</i>

Tabel 1: Requirements van DSW

Van de onderzochte image viewer controls voldoen geen aan alle requirements. Elk van de gewenste functies is in tenminste één product te vinden, maar geen enkel product bevat alle functies. Daarbij is op de eerste plaats gekeken naar *must* requirements, gevolgd door *should* en *could*. Enkele van deze producten zijn:

- **ImageMan.Net V1.51** van Data Techniques Inc. (data-tech.com)
- **ImagXpress™ V9.0** van Pegasus Imaging Corporation (pegasusimaging.com)
- **THBImage 5.0.3.1** van THBComponentware (thbimage.com)
- **ImageGear for .NET v15.1** van Accusoft Corp (accusoft.com)
- **ImageGlue.NET V6.0** van webSupergoo Software (websupergoo.com)

Op www.devdirect.com is een lijst te vinden van andere onderzochte image viewer controls. Deze omvatten zowel COTS als Open Source viewer controls.

Op dit moment wordt te DSW ook een custom image viewer gebruikt. De te ontwikkelen control moet beschikken over tenminste alle functies van de oude. Voor DSW is het daarom geen optie om requirements te laten vervallen om een COTS product te kiezen. Uitgaande van dit punt is de keus voor custom software, dat zodanig ontworpen kan worden om aan alle eisen te voldoen, de juiste.

3. C# vs. Java

Voor de ontwikkelaars van de *image viewer control* is C# een nog onbekende taal. Voor de opdrachtgever is het wel van groot belang dat het te bouwen control in die taal wordt gebouwd. Dit omdat de uiteindelijke applicatie waarin de *image viewer control* geplaatst zal worden, in C# gebouwd wordt. Echter hebben de ontwikkelaars geen ervaring met die taal maar ruim voldoende met een andere programmeertaal, namelijk Java. In dit hoofdstuk zal daarom een vergelijking worden gemaakt tussen die twee programmeertalen.

De manier van compilen

C# is ontwikkeld door Microsoft als een onderdeel van het .NET platform. Dit is een taal onafhankelijk platform. Naast C#, vallen hieronder bijvoorbeeld ook de talen Visual Basic en C++, die beide object-georiënteerde programmeertalen zijn. Een applicatie geschreven in één van deze talen wordt gecompileerd naar byte-code. In het .NET platform heet deze code de Microsoft Intermediate Language (MSIL). Na het compilen zal de Common Language Runtime (CLR of runtime) de applicatie uitvoeren. De runtime vertaalt dan de MSIL-code naar de taal van de machine waar de applicatie wordt uitgevoerd. Hierbij wordt zowel gebruik gemaakt van het Just-in-time techniek als de Garbage Collector.

Dankzij het platform kunnen applicaties, geschreven in verschillende programmeertalen, met elkaar interacteren en zelfs met elkaar worden geïntegreerd. Dit maakt het .NET platform een zeer krachtig ontwikkeltool voor applicaties en uitgebreide systemen.

Java is geen onderdeel van een platform, wat dat betreft staat het op een eiland. Wel is het OS onafhankelijk. Zoals C# gebruikt maakt van het CLR voor het compilen van MSIL naar machine code gebruikt Java de JVM (Java Virtual Machine) voor het compilen van Java bytecode naar machinetaal. Ook hier gebruikmakend van de Garbage Collector en Just-in-time techniek.

Doordat Java en C# object-georiënteerde talen zijn, kan het gebruik maken van een uitgebreide library. Java heeft zijn API reference, hiërarchisch verdeeld in packages van klassen. Het .NET platform heeft het .NET Framework. Deze bestaat uit twee componenten, de CLR wat al is besproken en het .NET Framework class library. Alle .NET programmeertalen kunnen gebruik maken van deze library. Ook deze is hiërarchisch verdeeld, alleen heten de packages hier namespaces. Het .NET Framework class library is ook weer verdeeld in twee delen: de Base Class Library (BCL) en de Framework Class Library (FCL). De BCL is een kleine subset van de hele class library en bestaat uit belangrijke klassen die de basis vormen van de API van de runtime. De FCL is een superset van de BCL klassen and refereert naar alle klassen onder het .NET Framework.

Programmeer omgeving

Voor het programmeren zijn er verschillende ontwikkelomgevingen (IDE) waarmee gewerkt kan worden. Voor Java is er bijvoorbeeld Eclipse of JBuilder. In het .NET platform mag er natuurlijk geen programmeeromgeving ontbreken en dat doet het ook niet. Hier wordt gebruik gemaakt van een heel uitgebreide programmeeromgeving: Visual Studio. Deze omgeving is bruikbaar voor vele programmeertalen in het .NET platform voor het ontwikkelen van bijvoorbeeld Windows-applicaties met een graphical user interface, websites en webapplicaties.

Verschillen en overeenkomsten in werking en syntaxis

Java is een geraffineerde versie van C en C++ zonder pointers, multiple inheritance en operator overloading, maar met multiple-platform support. C# kan weer worden beschouwd als een hybride van Java en C++, met een aantal nieuwe mogelijkheden en veranderingen. Het

is ook 100% object-georiënteerd, heeft geen pointers of multiple inheritance, maar wel operator overloading. Door de vele overeenkomsten van de twee talen en omdat het beide talen zijn die zijn afgeleid van C en C++, is ook de syntaxis bijna het zelfde. Hierdoor zijn er ook heel veel keywords die met elkaar overeenkomen.

In dit gedeelte zullen nog een aantal belangrijke overeenkomsten, al dan niet met kleine verschillen, kort samengevat worden beschreven.

- Ook C# heeft een hiërarchisch klassenstelsel. Hier zijn alle klassen subklassen van `System.Object`, zoals in Java alle klassen subklassen zijn van `java.lang.Object`.
- Interfaces worden ook ondersteund in C#. Ook hier kan overerving plaatsvinden van meerdere interfaces. Overerving van meerdere klassen kan bij beide niet.
- Het mechanisme van de Exceptions zijn bijna helemaal het zelfde. Beide gebruiken een “try” en een “catch” blok voor het afhandelen van een exception. Alle exception-klassen stammen af van de klasse `Exception`.
- Datatypen en referentietypen kunnen automatisch in elkaar worden omgezet met behulp van inpakklassen. (Boxing en Unboxing)
- De methoden in Java worden geschreven met kleine letter, in C# wordt dat gedaan met een hoofdletter.
- Er zijn een aantal keywords in C++ die anders zijn in Java, maar het zelfde betekenen. Zoals:
 - `extends` en `implements` in Java, `:` (dubbele punt) in C#
 - `instanceof` in Java, `is` in C#
 - `package` in Java, `namespace` in C#
- In C# zijn er naast `private`, `public` en `protected` die er ook in Java zijn, twee extra access modifiers `internal` en `internal protected`.
- In C# zijn er de zelfde primitieve typen als in Java. Alleen zijn er in C# nog een aantal extra: `ulong`, `uint`, `ushort` and `sbyte`.
- De generics-template is het zelfde in concept als in Java, alleen wordt het tijdens de runtime anders uitgevoerd.
- Om een object `Serializable` te maken in Java moet de interface `Serializable` worden geïmplementeerd. In C# wordt dat gedaan door de `[Serializable]` attribuut toe te voegen.
- C# en Java hebben een feature voor het genereren van het commentaar van de source code in een apart document. Java heeft Javadoc om dat te doen. C# doet dat automatisch door het gebruik van het XML-formaat voor de documentatie.

Dit zijn een aantal belangrijke features die zich in C# bevinden en helemaal niet in Java.

- C# maakt gebruik van delegates. Een delegate is een referentietype dat een referentie heeft naar een methode. Dit is handig als wanneer je weet dat je applicatie een actie moet uitvoeren door een methode aan te roepen, maar nog niet weet welke tijdens het compilen.
- Behalve datatypen en referentietypen heeft C# ook “structs”. Dit kan gezien worden als lichtgewicht klassen, om het gebruik van geheugen te beperken.
- In C# kan gebruik worden gemaakt van properties. Het is een feature die de get- en setmethodes in Java herbergt.
- C# introduceert een nieuwe feature genaamd “indexers”. Dit wordt gebruikt een object te gebruiken als een array. Is ook wel bekend als smart arrays.
- In C# bestaat er de mogelijkheid om de argumenten in een methode verwijzen naar de eigenlijke items die worden doorgegeven, in plaats van kopieën van de items.

-
- Het gebruik van letterlijke strings. Tabs, aanhalingstekens, backslashes en een nieuwe lijn kunnen letterlijk worden gebruikt.
 - In C# kan een klasse worden gedefinieerd in meerdere source files. Dit is handig in geval van automatisch gegenereerde code. Wordt vooral gebruikt in bijvoorbeeld Visual Studio.

Er zijn ook een aantal features die zich in Java wel bevinden en in C# niet, zoals: checked exceptions, inner classes en extensions.

Tussen C# en Java zijn er aantal overeenkomsten en verschillen die niet in dit rapport zijn aangegeven. Door de ontwikkelaars worden die als minder belangrijk geacht. Met de omschreven overeenkomsten en verschillen in gedachte denken zij de programmeertaal C# voldoende te beheersen.

4. Conclusie

Dit rapport werd het vooronderzoek behorende bij het *Image viewer control* project bij de Zorgverzekeraar DSW. De onderzoeksvraag werd als volgt geformuleerd:

“Is het juist dat DSW vraagt om een custom image viewer, waar er een groot aanbod van commerciële Off-The-Shelf image viewers bestaat?”

Om antwoord te geven op deze vraag werden de voor- en nadelen, en de eventuele voorwaarden van een keus tussen *custom* en *COTS* software onderzocht. Vervolgens werden de bevindingen van het bovenstaand onderzoek in context gebracht met de situatie te DSW. Daarna werd gezocht naar COTS en Open Source software dat voldeed aan de requirements van DSW.

Van alle onderzochte controls werd geen één gevonden dat voldeed aan *alle* requirements van de zorgverzekeraar. Elk gewenste functie was wel te vinden in tenminste één control. Uitgaande van dit punt werd geconcludeerd dat het efficiënter is om een *custom* image viewer control te bouwen. De keus van DSW om dit te laten doen is dus de juiste.

Als aanvulling tot het bovenstaand onderzoek werd gekeken naar de verschillen overeenkomsten tussen twee programmeertalen namelijk C# en Java. Dit omdat de opdrachtgever een stukje software vraagt in C#, waar de in dienst genomen ontwikkelaars voldoende ervaring in Java bezitten.

C# bevat een aantal eigenschappen, zoals boxing, enumerations en pass by reference, die een Java programmeur plezierig zullen verrassen bij het gebruik van C#. Aan de andere kant zijn er eigenschappen in Java die zeker gemist zullen worden, zoals inner classes, checked exceptions en extensions.

Voor de ontwikkelaars lijkt de overstap van Java naar C#, en omgekeerd, eentje die niet veel inspanning zal kosten, immers de talen komen in meerdere opzichten overeen dan ze verschillen.

Literatuurlijst

- [1] National Research Council, (1997), “Ada and beyond: Software policies for the department of defense”, Washington, D.C. National Academy Press
- [2] Lethbridge, T.C., Laganière, R. (2005), “Object-oriented software engineering”, 2nd edition, New York, McGraw-Hill Education
- [3] Beheshti, J., Dupuis, J. “Problems with COTS Software: A Case Study”, McGill University
- [4] Abts, C., (2002), “COTS-based Systems and Make vs. Buy Decisions: the Emerging Picture”, Texas, A&M University, Information & Operations Management Department
- [5] Equils, D.J., “Method for Enhancing the Process of Software Tool Evaluation and Selection: COTS, Heritage, and Custom Software Reviewed”, Jet Propulsion Laboratory, Pasadena, CA, U.S.A.
- [6] Obasnjó, D., (2007), “A comparison of Microsoft’s C# programming language to Sun Microsystems’ Java programming language”
- [7] Microsoft, (2002), Microsoft Official Course Book “Introduction to C# programming with Microsoft .NET”

Bezochte websites:

- [8] http://en.wikipedia.org/wiki/C_sharp_%28programming_language%29
- [9] http://en.wikipedia.org/wiki/Microsoft_.NET#Microsoft_.NET
- [10] http://en.wikipedia.org/wiki/Comparison_of_C_Sharp_and_Java
- [11] <http://www.csharpcorner.com/UploadFile/srimanigvenkataraman/ComparisonJavaandCs11292005035430AM/ComparisonJavaandCs.aspx>

Elk van de bovenstaande websites is op 9 juli 2008 voor het laatst bezocht.

Bijlage C: Plan van Aanpak

BSc-project IN3405



Plan van Aanpak

Omgeving:

Zorgverzekeraar DSW
Technische Universiteit Delft
Faculteit EWI Technische Informatica
Variant Softwaretechnologie (ST)

Projectgroep:

Abdoelrasheed Jarmohamed
Kishen Gajadhar

Commissie:

Job Scheffers (Zorgverzekeraar DSW, begeleider)
B.R. Sodoyer (TU Delft, coördinator / begeleider)

Datum: 3 Juli 2008

Status: Wachten op acceptatie (B.R. Sodoyer)

Versie: 2.0

©Juli 2008 Zorgverzekeraar DSW

Alle in dit verslag afgebeelde gegevens, waaronder teksten, foto's, illustraties, grafische materiaal, (handels)namen, logo's, waren- en dienstmerken, zijn in eigendom van zorgverzekeraar DSW en worden beschermd door auteursrecht, merkenrecht en/of enig ander intellectueel eigendomsrecht.

Inhoudsopgave

1. Introductie	61
2. Projectopdracht	62
2.2 <i>Projectomgeving</i>	62
2.3 <i>Doelstelling van het project</i>	62
2.4 <i>Beschrijving projectfases en documentatie</i>	62
2.5 <i>Eisen</i>	63
2.6 <i>Kritische succesfactoren</i>	63
3. Aanpak.....	64
3.1 <i>Het waterval model</i>	64
3.2 <i>De projectopbouw</i>	64
4. Planning	66
5. Project verloop	68
5.2 <i>Communicatie</i>	68
5.3 <i>Documentatie</i>	68
Literatuurlijst	69
Bijlage A: Commentaar in code.....	70

1. Introductie

Voor u ligt het plan van aanpak voor het *image viewer control* project. Dit plan is geschreven in het kader het bovengenoemde bachelorproject welke een onderdeel is van het bachelorcurriculum van de opleiding technische informatica van de Technische Universiteit Delft. Het doel van dit plan is om een duidelijke planning te verkrijgen waarin alle activiteiten gestructureerd beschreven zijn. Dit plan zal daarom gebruikt worden in alle fases van het project.

De opbouw van dit document is als volgt. In hoofdstuk 2 is de opdracht van opdrachtgever, zorgverzekeraar DSW, uitgelegd. Vervolgens staan in hoofdstuk 3 de te doorlopen fases van begin tot eind beschreven. In hoofdstuk 4 is een globale planning van de activiteiten opgenomen gevolgd door het verloop van communicatie en documentatie binnen het project in hoofdstuk 5.

2. Projectopdracht

In dit hoofdstuk wordt de projectopdracht in het licht gebracht. Allereerst wordt de projectomgeving besproken, gevolgd door de doelstelling en een beschrijving van het project. Uiteindelijk worden de eisen en beperkingen en de kritische succesfactoren van het project behandeld.

2.2 Projectomgeving

Het project vindt in zijn geheel plaats in het kantoor van DSW te Schiedam. Gewerkt wordt op de ICT-afdeling in de .Net groep. Binnen DSW wordt momenteel een custom-made applicatie gebruikt om gescande documenten te bekijken. Met deze toepassing kunnen administratieve medewerkers de via post ontvangen documenten voor zich halen om vervolgens de betreffende informatie in een back-end applicatie in te voeren en met die informatie eventuele vervolg handelingen uit te voeren. De huidige applicatie is een stand-alone applicatie, waardoor het moeilijk is voor andere applicaties om aan te sturen welk plaatje getoond moet worden. Een voorbeeld hiervan is de workflow applicatie die bepaalt welk dossier de medewerker moet behandelen en dus welke gescande brief/brieven de image viewer moet tonen. De aansturing van de image viewer loopt nu via een ActiveX control dat onderdeel is van de image viewer. De medewerkers moeten zelf hun applicatie en de image viewer zo sizen, dat beiden op het beeldscherm passen. De ontwikkelaar van een applicatie heeft geen enkele invloed op wanneer en hoe de image viewer getoond wordt

2.3 Doelstelling van het project

De doelstelling van het project is het ontwerpen van een viewer met minstens de zelfde functionaliteiten als de huidige, maar dan in een vorm waarbij ontwikkelaars de viewer in hun eigen scherm kunnen integreren. Deze nieuwe viewer moet ook nog gemakkelijk geïntegreerd kunnen worden in de verschillende componenten van het systeem die gebruikt worden. Bij de verschillende componenten moet worden gedacht aan bijvoorbeeld de invoercomponent, de opvraagcomponent, scancomponent of in de toekomst nog te ontwikkelen componenten.

2.4 Beschrijving projectfasen en documentatie

Voor het beginnen aan het eigenlijke project, zal er eerst twee weken onderzoek moeten plaatsvinden. Dit is verplicht gesteld door de TUD. Het onderzoek kan over verschillende onderwerpen gaan en de keuze van de onderwerpen kunnen we zelf bepalen. De onderwerpen moeten wel in nauw verband staan met het project. De onderwerpen van onze onderzoek zullen zijn “Het gebruik van .Net en C#” en “De keus van DSW om een custom image viewer te (laten) ontwikkelen in plaats van een generic versie aan te schaffen”.

Na het onderzoek beginnen we met het project. Het project wordt verdeeld in verschillende fasen die uitgebreid zullen worden gedocumenteerd. Allereerst worden de eisen onderzocht en geanalyseerd. Hierna wordt met behulp van de eisen alles ontworpen. Vervolgens wordt het ontwerp geïmplementeerd en getest.

Tijdens het gehele project zal er een wekelijks voortgangsrapport gemaakt worden als dat nodig is. Dit zal een overzicht zijn van mijlpalen, veranderingen, problemen, toekomstige zaken met betrekking tot de stand van het project.

De documentatie van de analyse zal bestaan uit een Requirement Analysis Document. Voor het ontwerp een Design Document. De documentatie voor de implementatie zal ook worden opgesteld. Na de implementatie zal er een testplan worden geschreven. En dan tot slot wordt er een eindrapport samengesteld.

Daadwerkelijke opdracht

Het ontwikkelen van een image viewer in een usercontrol, zodat deze functionaliteit in diverse applicaties gebruikt kan worden. Daarbij is een usercontrol een type customcontrol, welke een software component in .Net omgeving is. De ontwikkelaar (DSW-medewerker) kan dan het tonen van het gescande document een integraal onderdeel van de UI maken.

Gevraagd wordt naar een image viewer control, in .Net omgeving, waarbij de documenten, bitmap images, *efficiënt* vergroot, verkleind, geroteerd, enz. kunnen worden.

Op basis van dit usercontrol wordt een tweede usercontrol gevraagd, dat als invoer een array van bitmap images ontvangt die verkleind in een rij worden getoond en waarbij een geselecteerde image in het eerste usercontrol wordt weergegeven.

Vervolgens is er nog behoefte aan een usercontrol dat kennis heeft van de bij DSW gebruikte image database en de bijbehorende kenmerken database. Dit derde control voegt een zoekfunctie toe. Daarbij wordt de imagedatabase doorzocht en de resultaten weergegeven in een array, zoals beschreven bij het tweede usercontrol.

2.5 Eisen

De eisen van DSW omvatten het implementeren van een image viewer met tenminste de functionaliteiten van de huidige stand-alone image viewer alsook de mogelijkheid voor DSW ontwikkelaars om de viewer te integreren in hun eigen scherm. Een belangrijke kwaliteitseis is ook duidelijke externe- en interne documentatie van de code. Dit is van belang met het oog op toekomstige situaties waar de toepassing gebruikt zal worden. De ontwikkeling en implementatie dienen op locatie bij DSW plaats te vinden. Daarbij is een aanwezigheid van 32 werkuren per week vereist. Als programmeertaal is C# een eis.

2.6 Kritische succesfactoren

Een van de cruciale succesfactoren is 'kennis'. Kennis van de .Net programmeeromgeving en het programmeren met C#. Daarnaast kennis van de te gebruiken image control-technieken zoals zoom- en rotatie technieken. Het programmeren met C# zullen we nog moeten leren. Dit zal worden gedaan met behulp van een leerboek van C# dat ter beschikking is gesteld door de werkgever. Ook zal de benodigde kennis van het Internet worden gehaald als dat nodig is. Een risicofactor is daarmee dus 'kennis', aangezien dit zal bepalen in welke mate een goede analyse, ontwerp en uiteindelijk consistente implementatie kan plaatsvinden. Een ander succesfactor is het regelmatig overleggen om de voortgang van het project te bewaken en bij te sturen waar nodig. Goede onderlinge communicatie en tijdig problemen signaleren naar de opdrachtgever staan centraal. Ze vormen een belangrijk uitgangspunt om het project tot een succes te maken. Daarnaast dient het op te leveren product minstens de huidige functionaliteiten te bieden.

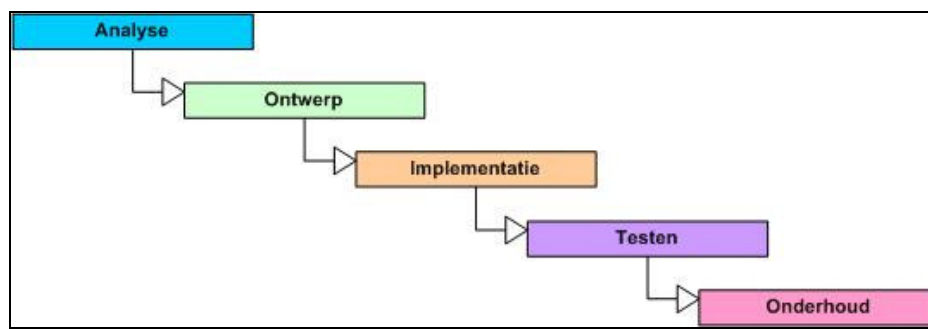
3. Aanpak

In dit hoofdstuk wordt de aanpak verduidelijkt. Er zal ingegaan worden op de verschillende fases van het project en wat de bedoeling is van elke fase.

Wat betreft de aanpak en de te gebruiken methodieken heeft de opdrachtgever ons hierin helemaal vrijgelaten. We kunnen dus de aanpak en de methodieken die we willen gebruiken in het project zelf bepalen. De verschillende fases zullen we aan de hand van het watervalmodel afwerken. En de methodiek die we zullen gebruiken is UML.

3.1 Het waterval model

Dit project zal gefaseerd worden volgens het bekende watervalmodel (zie figuur 1). Deze methodiek is aangehouden omdat deze gestructureerd is en in de praktijk veelvuldig gebruikt wordt. Na elke fase zal de feedback van de opdrachtgever voldoende zijn zodat eventuele fouten of misvattingen opgelost kunnen worden voordat de volgende fase wordt gestart.



Figuur 1: Het waterval model

3.2 De projectopbouw

Het project zal als volgt verlopen:

Analyse:

Tijdens deze fase zullen de gebruikerseisen achterhaald worden. Dit zullen wij doen door middel van gesprekken met de opdrachtgever, enkele gebruikers en de ontwikkelaar van het huidige image viewer. De eisen en wensen voortvloeiende uit deze gesprekken zullen worden voorzien van een MoSCoW-prioriteit. Vervolgens zullen use-cases en scenario's bedacht en gemodelleerd worden in UML. Dit alles zal samengevat worden in het RAD. Na evaluatie en goedkeuring zal pas worden overgegaan naar de volgende fase en anders worden de benodigde aanpassingen gemaakt.

Ontwerp:

Aan de hand van het verkregen MoSCoW-model in de analyse, zal er een eerste ontwerp gemaakt worden. Dit ontwerp bestaat uit een gedetailleerd klassendiagram, volgorde diagrammen en een GUI schets. Eventueel worden er ook nog een use-case diagram, statechart diagram en een activity diagram gemaakt met behulp van UML. Deze diagrammen worden allemaal toegevoegd in de Design Document.

Het opgestelde ontwerp zal hier ook kritisch worden bekeken door de opdrachtgever en begeleider. Hierbij wordt gekeken of het ontwerp daadwerkelijk in overeenstemming is met de verkregen requirements. Dit zal gebeuren door vanuit elk van de opgestelde use-cases de volgordediagrammen te bestuderen om te kijken of aan de vereiste requirements worden voldaan en de benodigde functies in het ontwerp ter beschikking zijn.

Door de verkregen feedback zal eventueel het eerste ontwerp worden aangepast en zal er een uiteindelijk ontwerp komen, wat dient als basis voor de implementatiefase.

Implementatie:

Tijdens de implementatie fase zal de software geschreven worden. Het ontwerp dat is gemaakt in de vorige fase zal worden gerealiseerd. Tussen het implementeren door zal ook het nodige worden getest. Dit zullen kleine testjes zijn die handmatig zullen worden gedaan. Eventueel zullen er ook kleine testapplicaties worden geschreven.

Allereerst zal een geraamte van de code worden gemaakt, waarbij voor iedere functie doormiddel van pre- en postcondities wordt aangegeven wat de functionaliteit omvat van de gegeven functie. Voor documentatie doeleinden zal hierbij ook commentaar worden toegevoegd over de werking van de functie zoals deze binnen het ontwerp beoogd is.

Testen:

Nadat alles is geïmplementeerd en de applicatie functioneert, zal het worden getest. Dit zal ook handmatig worden gedaan. Het testen wordt gedaan door onszelf, de opdrachtgever en eventueel ook nog door een aantal gebruikers. Per testcase wordt de verwachte uitkomst bij een bepaalde invoer gespecificeerd. Als de werkelijke uitvoer precies de verwachte uitkomst is, wordt de test als succesvol beschouwd.

4. Planning

In dit hoofdstuk volgt een planning voor de komende weken. Per week wordt globaal aangegeven wat de bijbehorende activiteiten en deliverables zijn.

Voor dit project zijn in totaal 12 weken uit getrokken. Twee daarvan zijn gereserveerd voor een verplicht onderzoek. Zoals eerder vermeld gebruiken wij het waterval model als methodiek voor dit project. Na elk van de vier ontwikkeling fases zal verbetering of goedkeuring volgens alvorens naar een volgende fase wordt overgegaan.

Hier onder een overzicht van de komende 12 weken met per week de activiteiten en deliverables.

Week 1 (1-07-2008 / 4-07-2008)

We starten met het maken van een plan van aanpak. Simultaan daaraan zullen we starten met het onderzoeken van de gestelde onderwerpen namelijk waarom DSW haar software zelf laat ontwikkelen in plaats van off-the-shelve versies aan te schaffen. Ook zal de werking van C# en de .Net omgeving onderzocht worden.

Deadline: 3-07-2008: 15.00 'plan van aanpak; 1^e concept'

Week 2 (7-07-2008 / 11-07-2008)

Eventuele verbeteringen opnemen in het plan van aanpak en het onderzoek vervolgen.

Deadline: 8-07-2008: 09.00 'plan van aanpak; final'

11-07-2008: 09.00 'onderzoeksrapport'

Week 3,4 (14-07-2008 / 25-07-2008)

In week 3 zal de analyse officieel van start gaan. Eerst worden de eisen achterhaald. Dit zal gebeuren door verschillende gesprekken met gebruikers en het zelf doornemen van de huidige applicatie en documentatie. De verkregen informatie zal worden geanalyseerd door middel van gebruikersanalyses, use-case diagrammen, globaal klassendiagram en lijst van eisen. Deze eisen worden gecategoriseerd in functionele en niet-functionele eisen. Op de functionele eisen passen we vervolgens het MoSCoW model toe. Dit alles wordt globaal gedocumenteerd en ter goedkeuring voorgelegd aan de opdrachtgever. In afwachting van feedback van de opdrachtgever zal, in week 4, het document verfijnd en na de feedback afgerond worden. Eventueel wordt ook gestart met de volgende fase.

Deadline: 25-07-2008 – 09.00 'Requirements Analysis Document'

Week 5, 6 (28-07-2008 / 8-08-2008)

In deze weken zal gewerkt worden aan het ontwerp. Aan de hand van de informatie verkregen in de analyse fase zal de software in detail ontworpen en beschreven worden. Hiervoor zal gebruik gemaakt worden van 'klassendiagrammen' en 'volgordediagrammen'. De use-cases zullen worden gebruikt om de gestelde functionaliteiten te controleren. Verificatie door middel van volgorde diagrammen, het doorlopen van een functie en het resultaat vergelijken, zal ook gebruikt worden.

Naast het functionele ontwerp zal ook tijd worden besteed aan het UI ontwerp. Dit zal worden gedaan door een aantal schetsen of prototypen aan de opdrachtgever en gebruikers voor te leggen. Aan de hand van de goedkeuring van een der prototypen zal een ontwerp geselecteerd worden.

Ook zal wat tijd worden besteed aan een testplan; hoe wij de applicatie, de individuele units en dergelijke gaan testen. Initieel is het de bedoeling dat voor elke klasse een test klasse geschreven en uitgevoerd wordt. Als na afloop alle tests succesvol zijn verlopen volgt een test van de applicatie in zijn geheel. Dat zal worden gedaan door middel van handmatige tests; de applicatie draaien, functies laten uitvoeren en verifiëren dat alles succesvol verloopt.

Evenals in de vorige fase zal hier eerst een concept document voorgelegd worden aan de opdrachtgever en aan de hand van eventuele feedback worden aangepast.

Deadline: 8-08-2008: 09.00 'Design Document'

11-08-2008: 09.00 'Test plan'

Week 7 t/m 10 (11-08-2008 / 5-09-2008)

Week 7 is het begin van de implementatie. Hier zal het echte programmeerwerk starten. Het ontwerp zal geïmplementeerd worden in de taal C# en volgens het testplan getest worden. Aan de hand van het testplan wordt gekeken of elke functie de juiste functionaliteit biedt en of het product aan de gestelde requirements voldoet. Na afronding van een klasse of functionaliteit wordt die gelijk getest.

In week 8 wordt een evaluatie middag georganiseerd. Dan zal samen met de opdrachtgever en begeleider de voortgang besproken worden, welke doelen gehaald zijn en waar nodig aanpassingen of prioriteiten doorvoeren.

Week 11,12 (8-09-2008 / 19-09-2008)

In week 11 zal de geïmplementeerde code voor de laatste keer uitvoerig getest en de implementatie afgerond worden. De nadruk ligt echter meer op het testen van de applicatie. Alle opgeleverde klassen zullen nog op volledige functionaliteit en samenhang getest worden. Door middel van handmatige invoer zal hierbij ook worden getest. Gebruikers zullen worden benaderd om een aantal test gevallen door middel van handmatige invoer te testen. In week 12 zal ook met het eindrapport gestart worden.

Week 13 (22-09-2008 / 26-09-2008 & 30-09-2008)

De laatste week van onze stage is bedoeld om eventuele vertragingen goed te maken, maar vooral om het eindrapport af te ronden. Aan het eind van elk van de vorige fases zal een document worden ingeleverd dat deel uit gaat maken het eindrapport. In deze week zal de samenvoeging afgerond moeten zijn. Aan het eind van deze week wordt het eindrapport en de definitieve werkende applicatie ingeleverd.

Deadline: 30-09-2008: 24.00 'Eindrapport' & 'Image viewer control'

5. Project verloop

In dit hoofdstuk wordt gespecificeerd hoe de communicatie binnen de groep zal verlopen en aan de hand van welke standaarden de documentatie geproduceerd zal worden.

5.2 Communicatie

De groepsleden zitten dagelijks op het kantoor van DSW. Mondelinge communicatie kan daarom gemakkelijk. Verder kan ook gecommuniceerd worden door middel van mail en telefoon. Voor bestanden overdracht zal gebruik worden gemaakt van de mail services.

Verder zal ten behoeve van een goed project verloop een wekelijkse vergadering worden ingepland. Daar zal de huidige stand van zaken besproken worden en toekomstige handelingen volgens de planning verdeeld worden. Van elk van deze vergaderingen zal een samenvattend document volgen.

Met de TU begeleider zal als nodig een evaluatie gesprek georganiseerd worden. Via mail worden alle documenten uitgewisseld. Daaronder vallen de in te leveren documenten door de groep en de reflectie daarop door de begeleider.

5.3 Documentatie

Aan het eind van elk van de fases wordt een document ingeleverd bij de begeleider. Zoals aangegeven zijn dit het plan van aanpak, het RAD, het Design document inclusief testplan en het eindrapport. Het eindrapport wordt gevormd door de eerdere documenten samen te voegen en het toevoegen van inleidende en concluderende hoofdstukken. De structuur van het eindrapport komt er dan zo uit te zien:

Voorwoord

Inhoudsopgave

Hoofdstuk 1:

Inleiding

Hoofdstuk 2:

Bedrijfsprofiel DSW

Hoofdstuk 3:

Probleemstelling en analyse

Hoofdstuk 4:

Ontwerp

Hoofdstuk 5:

Implementatie & resultaten

Hoofdstuk 6:

Proces

Hoofdstuk 7:

Conclusie, aanbevelingen en tekortkomingen

Hoofdstuk 8:

Reflectie

Literatuurlijst

Bijlage A:

Verklarende woordenlijst

Bijlage B:

Onderzoeksrapport

Bijlage C:

Initiële plan van aanpak

Alle documenten worden in MS Word geproduceerd en met de volgende specificaties:

- Lettertype: Times New Roman
- Lettergrootte: 11
- Datum, Status (in ontwikkeling, wachten op acceptatie, goedgekeurd), Versienr

De final versies van de documenten zullen in PDF worden ingeleverd

Documentatie van de code zal gebeuren volgens de standaarden van DSW zelf (zie bijlage A)

Literatuurlijst

Lethbridge, T.C., Laganière, R. (2005), “Object-oriented software engineering”, 2nd edition, New York, McGraw-Hill Education

Bijlage A: Commentaar in code

(Bron: <http://nt54/Wiki/Commentaar%20in%20code.ashx>)

Inleiding

Het commentariëren van code is een belangrijk deel van de oplevering van code. Hoewel het een taak is die niet altijd plezier is, zorgt het er wel voor dat je code lees- en begrijpbaar blijft, voor zowel jezelf als voor ieder ander die ermee in aanmerking komt.

Een goed stuk code is niet alleen goed qua concept, algoritme, of ontwerp, maar ook qua leesbaarheid. Soms is een algoritme te complex om direct te kunnen begrijpen zonder verdere uitleg, of er wordt een functie aangeroepen met een misleidende naam wat voor meer verwarring zorgt. Ook kan het soms zijn dat je terug moet kijken naar een stuk code dat je maanden of wel jaren terug hebt geschreven. In zulke gevallen zou het handig geweest zijn als je commentaar bij je code geschreven had. Commentaar in code zorgt ervoor dat (moeilijke) stukken code snel te snappen en begrijpbaar worden. Een ander voordeel is dat door het schrijven van commentaar je verplicht wordt erover na te denken en zodoende eventuele bugs eruit kunt halen.

Soorten Commentaar

Code Commenting

Dit verwijst naar het gebruiken van functie- en variabelnamen die zichzelf uitleggen. Een voorbeeld hiervan:

```
function addUserToDatabase(userName, userAge)
```

Inline Commenting

Dit is de meest basisvorm van commentaar. Commentaar wordt hierbij geschreven aan het einde van een regel code, of in een functie. Voorbeeld:

```
function calculateHitPoints(cUser) {  
    var nStrength = document.getElementById("enemyStrength").value;  
    // grab current enemy strength  
  
    // subtract user size : small = 1, medium = 2, large = 3  
    var nDamage = (nStrength * 3) ♦ cUser.nSize;  
    return cUser.nCurrentHitPoints ♦ nDamage;  
}
```

Function Commenting

Dit soort commentaar wordt gevonden boven een functie. Het geeft details over de functie, waaronder de over de parameters en return waarden. In C# moet hierbij gebruik gemaakt worden van XML commenting. Voorbeeld (non-XML):

```
/*  
 * Summary:          Calculate hitpoints after attack using formula
```

```
*          new = current ♦ ((enemyStrength*3) ♦ size)
* Parameters:  cUser ♦ object containing hero's stats
* Return:     Boolean indicating life or death
*/
function calculateHitPoints(cUser) {
    ...
}
```

Class / Page Commenting

Dit soort commentaar gaat over een hele pagina, of class. Meestal bevat dit soort commentaar een algemeen overzicht met informatie als laatste edit datum, bijbehorende bestanden, auteur, contact informatie, etc. Voorbeeld:

```
/*
-----
Title       :
Author      :
Description :
Created     :
Modified    :
-----
• /
```

XML Commenting

C# ondersteunt XML code documentatie. De C# compiler is vervolgens in staat om dit XML commentaar te gebruiken om er XML documentatie van te genereren. Om XML commentaar te schrijven moet men gebruik maken van een speciale syntax, '///' (drie forward-slashes). Na deze slashes moet gebruik gemaakt worden van een van de voorgedefiniëerde tags (of je eigen custom-tag).

Gebruik bij het schrijven van code en commentaar altijd XML commentaar!

Predefined Tag	Used for
<c>	A way to indicate that text within a description should be marked as code
<code>	A way to indicate multiple lines as code
<example>	Lets you specify an example of how to use a method or other library member
<exception>	Lets you document an exception class
<include>	Lets you refer to comments in another file, using XPath syntax, that describe the types and members in your source code.
<list>	Used to insert a list into the documentation file
<para>	Used to insert a paragraph into the documentation file
<param>	Describes a parameter
<paramref>	Gives you a way to indicate that a word is a parameter
<permission>	Lets you document access permissions

<remarks>	Where you can specify overview information about the type
<returns>	Describe the return value of a method
<see>	Lets you specify a link
<seealso>	Lets you specify the text that you might want to appear in a See Also section
<summary>	Used for a general description
<value>	Lets you describe a property

IntelliSense

Een bijkomend voordeel bij het gebruik van XML comments is dat ze toegevoegd worden aan IntelliSense en daardoor het schrijven van code vergemakkelijken. Als bij een methode de parameters beschreven worden met XML commentaar, en elders deze methode aangeroepen wordt, dan wordt de opgegeven commentaar getoond ter informatie.

Voorbeeld:

```
using System.Collections.Generic;
```

```
class Program
{
    /// <summary>
    /// DoSomething takes a <see cref="List{T}" />
    /// </summary>
    /// <remarks>
    /// This function does ...
    /// </remarks>
    void DoSomething(List al) { }
}
```

Algemene Richtlijnen

- Kies een taal: de taal van het commentaar is in principe afhankelijk van de doelgroep. Dit zal normaal gesproken een programmeur zijn die het Engels voldoende beheerst om commentaar in deze taal te kunnen lezen. Omdat keywords in C# en ook veel objecten en classes afstammen van het Engels, is deze taal het meest geschikt voor commentaar.
- Gebruik de in C# ingebouwde ondersteuning voor commentaar. Als je boven een methode drie slashes typt dan genereert Visual Studio een XML template voor commentaar. Dit heeft allerlei voordelen: Het commentaar is meteen beschikbaar in intellisense, het dwingt een bepaald formaat af voor het commentaar, er kunnen HTML-rapporten worden gegenereerd op basis van het commentaar (onder andere via 'Tools-Build Comment Web Pages...'). Als

de code zich in een andere assembly bevindt dan kan de code worden opgeslagen als XML waardoor het commentaar beschikbaar is in alle code die gebruikt maakt van die assembly.

- Denk aan de hoeveelheid: code kan door veel commentaar ondoorzichtig worden. Er is dan te weinig programmacode aanwezig op een scherm zodat het overzicht lastig is te bewaren. Veel commentaar is overbodig als je goede namen kiest voor variabelen en methoden. Wanneer een variabele 'maximum' heet is het niet nodig in commentaar erbij te vermelden dat dit het maximum is. Een variabele die x heet heeft een dergelijk commentaar wel nodig.
- Beschrijf het 'Wat', niet het 'hoe': het doel van commentaar is het onderhoud van de code gemakkelijker te maken. De beste manier om dit te doen is dingen toe te voegen die niet direct uit de code volgen. Dit betekent dat het van belang is de reden aan te geven waarom iets gedaan wordt, en niet zozeer de manier waarop. Deze manier van commentariëren is ook beter bestand tegen codewijzigingen. Vaak wordt de manier waarop iets gedaan wordt aangepast, maar minder vaak de reden waarom.

Voorbeeld

Fout:

```
private ArrayList GetCustomersForName(string name)
{
    // This collection contains the found customers
    ArrayList list;
    list = new ArrayList(); //Create a new ArrayList
    // Loop all customers in the collection
    foreach (Customer customer in list)
    {
        // Compare the name with the passed parameter and type
        if (customer.Name == name && customer.Type ==
CustomerType.Business)
        {
            // Add this customer
            list.Add(customer);
        }
    }
    // Return the collection
    return list;
}
```

Goed:

```
/// <summary>
/// Find all the customers with the passed name and put them
/// all in a new collection. If the customer is a Business-type
/// the Name-property contains the CompanyName. These customers can
/// not be found by their name.
/// </summary>
/// <param name="name">The name of the customer.</param>
/// <returns>A collection of customers</returns>
private ArrayList GetCustomersForName(string name)
{
    ArrayList list;
```

```
list = new ArrayList();
foreach (Customer customer in list)
{
    if (customer.Name == name && customer.Type ==
CustomerType.Business)
    {
        list.Add(customer);
    }
}
return list;
}
```

In het eerste voorbeeld wordt teveel beschreven wat er gedaan wordt, terwijl dit ook uit de code volgt. Relevante informatie (zoals waarom een businesstype niet meedoet) wordt niet vermeldt. Het tweede voorbeeld legt niet uit wat de code doet, maar beschrijft waarom de code zo gemaakt is. Het is dan ook niet nodig commentaar te geven bij standaard statements.

Bijlage D: RAD viewer control

BSc-project IN3405



**ZORGVERZEKERAAR
DSW**

Requirements Analysis Document Viewer control

Omgeving:

Zorgverzekeraar DSW
Technische Universiteit Delft
Faculteit EWI Technische Informatica
Variant Softwaretechnologie (ST)

Projectgroep:

Abdoelrasheed Jarmohamed
Kishen Gajadhar

Commissie:

Job Scheffers (Zorgverzekeraar DSW, begeleider)
B.R. Sodoyer (TU Delft, coördinator / begeleider)

Datum: 15 juli 2008

Status: wachten op acceptatie

Versie: 1.0

©Juli 2008 Zorgverzekeraar DSW

Alle in dit verslag afgebeelde gegevens, waaronder teksten, foto's, illustraties, grafische materiaal, (handels)namen, logo's, waren- en dienstmerken, zijn in eigendom van zorgverzekeraar DSW en worden beschermd door auteursrecht, merkenrecht en/of enig ander intellectueel eigendomsrecht.

Inhoudsopgave

1. Introductie	77
2. Bedrijfsprofiel.....	78
2.1 Overzicht zorgverzekeraar DSW	78
2.2 Rol van ICT Binnen DSW	79
2.3 Software Ontwikkeling Binnen DSW	79
3. Probleemdomein.....	80
3.1 Domeinanalyse.....	80
3.2 Toekomstige Ontwikkelingen	80
3.3 Daadwerkelijke opdracht.....	81
4. Algemene analyse	82
4.1 Aanpak.....	82
4.2 De Image Viewer.....	82
5. Gebruikersanalyse.....	83
6. Use case analyse.....	84
7. Requirements.....	86
7.1 Functionele requirements	86
7.2 Non-functionele requirements.....	87
Literatuurlijst	88
Bijlage A: Vraaggesprekken	89
Bijlage B: Gebruikersprofiel (ingevuld)	93
Bijlage C: Vragenlijst gebruikers	94

1. Introductie

Dit analyse rapport wordt geschreven in het kader van het bachelorproject van de opleiding technische informatica van de TU Delft. Deze opdracht vindt plaats bij zorgverzekeraar DSW te Schiedam.

Tijdens deze analyse is het ten eerste de bedoeling informatie te verzamelen over het gekozen probleemdomain. Dit gaat aan de hand van *interviews* en *observatie*. Het doel van het verzamelen van informatie is duidelijk: zonder informatie kan er geen overwogen keus gemaakt worden over het verloop van het project. Door middel van deze analyse kan het probleemdomain duidelijk gedefinieerd en afgebakend worden tot een eenduidig probleemdomain waarna een softwaresysteem zal kunnen worden ontworpen om dit probleem op te lossen.

Nadat de informatie is verworven wordt de informatie geanalyseerd. Hierbij kan gebruik gemaakt worden van gebruikersanalyses, use-case diagrammen en een lijst van de requirements. Deze analyse is nodig om uit te zoeken welke informatie nodig is voor het realiseren van het project en om uit te zoeken wat wel en niet haalbaar of wenselijk is voor dit project. Hierdoor kan het project gestructureerd worden uitgevoerd en kan er een goed werkend systeem ontworpen worden.

Het doel van de analyse is dus het afbakenen en definiëren van het probleemdomain dat gebruikt wordt voor het stroomlijnen van het ontwerptraject dat daarna verder zal gaan.

De opbouw van dit rapport is als volgt. In hoofdstuk 2 wordt een bedrijfsprofiel opgesteld van de zorgverzekeraar DSW. Vervolgens wordt in hoofdstuk 3 het probleemdomain besproken en in hoofdstuk 4 de gebruikers. In hoofdstuk 5 komt de use case analyse aan bod gevolgd door de lijst van requirements in hoofdstuk 6.

2. Bedrijfsprofiel

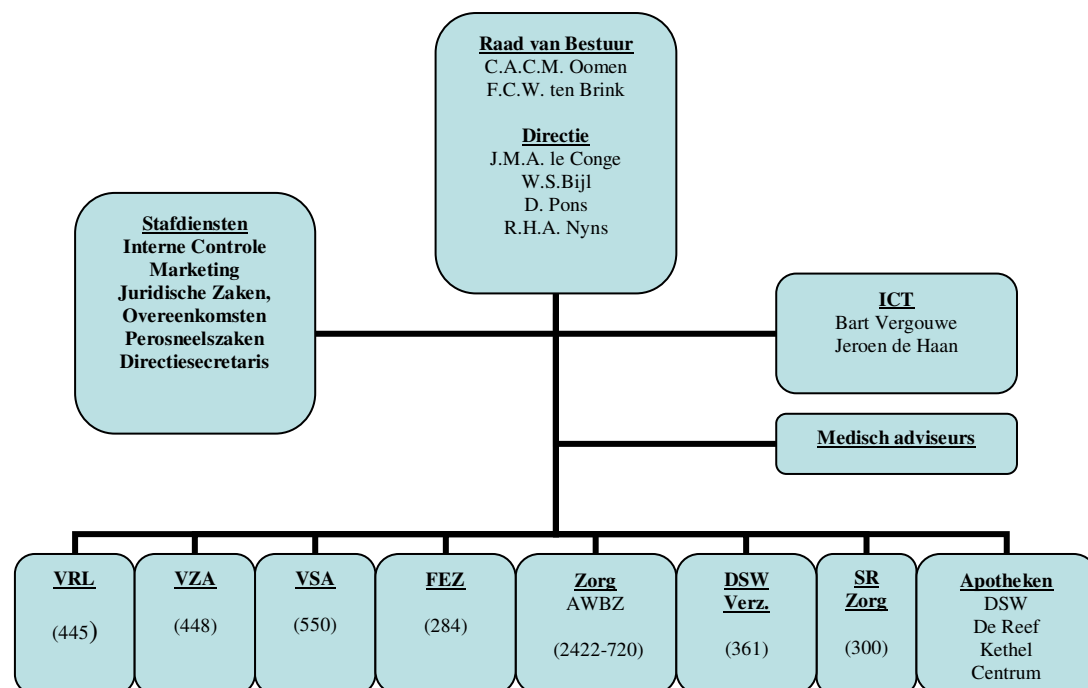
Het uit te voeren project zal voor een groot deel plaats vinden in het kantoorgebouw van de Zorgverzekeraar DSW te Schiedam. In dit hoofdstuk wordt het profiel van DSW beschreven. In paragraaf 2.1 volgt een kort overzicht van de bedrijfsorganisatie. Vervolgens in paragraaf 2.2 een indicatie van het belang van ICT binnen DSW. Ten slotte in paragraaf 2.3 een overzicht van het software ontwikkelingsproces.

2.1 Overzicht zorgverzekeraar DSW

Sinds januari 2006 zijn de mensen in Nederland wegens de Ziekenfondswet verplicht een ziekenfondsverzekering af te sluiten. Zorgverzekeraar DSW voert als gevolg van deze wet de verplichte ziekenfondsverzekering uit en is marktleider in de regio Delfland Schieland en Westland. Deze zorgverzekeraar heeft ruim 330.000 verzekerden.

Omdat DSW vindt dat iedereen recht heeft op betaalbare zorg van hoge kwaliteit, is zij betrokken bij een groot aantal zorginhoudelijke projecten. Het voornaamste doel van DSW is het bevorderen van een optimale gezondheidszorg voor haar verzekerden.

Binnen DSW zijn er verschillende afdelingen waaronder 'VZA: Verzekerden Administratie' welke direct gebruik zal gaan maken van het te bouwen control. Een overzicht van de organisatie wordt gegeven in *figuur 1* [1].



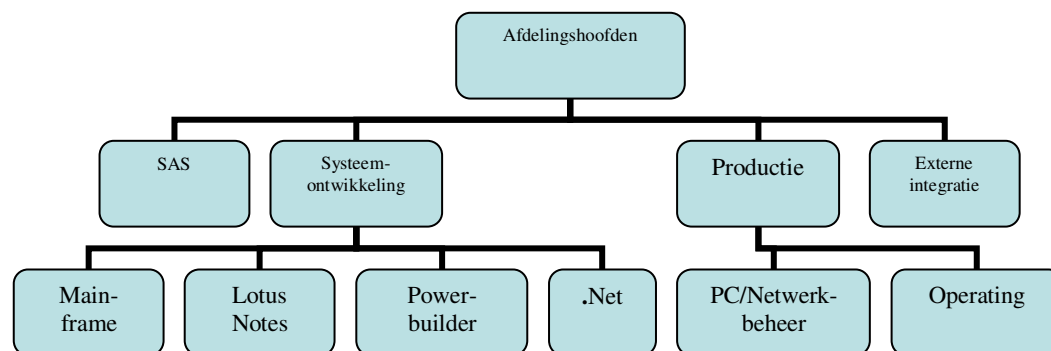
VSA – Verstrekkingen administratie
FEZ – Financieel Economische Zaken
DSW Verzekeringen
AWBZ – Volksverzekering

VZA – Verzekerden administratie
SR Zorg – SR Zorgverzekeraar
Apotheken – DSW, De Reef, Kethel, Centrum

Figuur 1: Bedrijfsorganisatie DSW

2.2 Rol van ICT Binnen DSW

Om zorg van hoge kwaliteit te kunnen bieden maakt DSW gebruik van allerlei mogelijke middelen op het gebied van automatisering. Hierbij worden de nieuwste technieken gebruikt. Administratieve verwerkingen worden bijna volledig met de computer gedaan binnen DSW en de afdeling ICT zorgt ervoor dat al deze systemen beschikbaar zijn. Bij het ontwikkelen van systemen wordt er rekening gehouden met de wensen en kennis van gebruikers. Ontwikkeling vindt hierbij 'in huis' plaats door de eigen ontwikkelaars voor de verschillende platformen. Systeem ontwikkeling vindt plaats voor/in Mainframe, Lotus Notes, Powerbuilder en .Net. Dit kan ook gezien worden in *figuur 2* waarin de hiërarchie binnen de afdeling ICT wordt weergegeven[1].



Figuur 2: Organisatie van de ICT afdeling

2.3 Software Ontwikkeling Binnen DSW

Bij het maken van software, wordt binnen DSW *Rational Unified Process-RUP* als methodiek gebruikt. RUP is grofweg een iteratief software ontwikkelingsproces. Door prototypering en terugkoppeling worden requirements voortdurend aangepast. Iteratief ontwikkelen binnen het project d.m.v deelproducten zorgt ervoor dat eventueel veranderende wensen van een klant niet het hele product beïnvloeden maar per deelproduct aangepast kan worden.

RUP is gebaseerd op de volgende principes:

- *Ontwikkel software iteratief (requirements, prototype, feedback & terugkoppeling)*
- *Maak gebruik van component gebaseerde architectuur*
- *Test het systeem*
- *Maak gebruik van versiebeheer tijdens de software ontwikkeling*

DSW schrijft niet strikt voor dat RUP gebruikt moet worden. Er kan ook gekozen worden voor een andere methodiek.

3. Probleemdomein

In dit hoofdstuk volgt een beschrijving van het probleemdomein. In paragraaf 3.1 wordt de domeinanalyse beschreven en wordt uitgelegd hoe er nu wordt gewerkt. In paragraaf 3.2 wordt beschreven waarnaar in de nabije toekomst wordt gestreefd. De opdracht zelf wordt beschreven in paragraaf 3.3.

3.1 Domeinanalyse

DSW heeft een eigen ICT-afdeling waar ze alle systemen ontwikkelen en onderhouden voor de verschillende afdelingen van DSW. De software in dit project zal worden ontwikkeld voor de Verzekerden-Administratie afdeling (VZA) en de Verstrekking-Administratie afdeling (VSA). Deze afdelingen versturen veel post en krijgen ook veel post binnen. Dit zijn meestal brieven van en naar klanten of ziekenhuizen. Vaak zijn dit ook rekeningen van klanten of ziekenhuizen. Al deze documenten moeten door verschillende medewerkers worden bekeken en worden allemaal digitaal opgeslagen, zodat iedere medewerker op de afdelingen alle documenten kan opvragen en inzien. De documenten worden dus gescand en in een database opgeslagen. Voor het bekijken van de documenten is een aantal jaren geleden een losstaande image viewer ontwikkeld met de gewenste mogelijkheden. In de image viewer zit een link naar de “opvraagstelsysteem” voor de documenten. Dit opvraagstelsysteem heet “Drovi”.

Als er op de link wordt gedrukt komt er een zoekscherm (versimpelde versie van Internet Explorer) tevoorschijn waarmee met de benodigde zoekcriteria kan gezocht worden naar een bepaald document of een selectie van documenten. Als de criteria zijn ingevoerd in de verschillende velden en op de zoekknop wordt gedrukt, stuurt het zoekscherm het verzoek door naar een IIS (Microsoft Internet Information Services of Server). De IIS zorgt ervoor dat er contact wordt gemaakt met de *Drovi-database*. Dit is een database met alle indices en id-nummers van alle documenten die zijn opgeslagen in een andere database genaamd: *InfoImage*. De indices en id-nummers worden dan gestuurd naar het zoekscherm. Deze zoekscherm laat dan een lijst van de namen van de gevraagde documenten zien. Als op een naam wordt geklikt, zal de index van het document door worden gegeven naar de image viewer. De image viewer maakt contact met InfoImage. Zoekt dan met behulp van de index naar het document en laat het vervolgens zien.

Als een medewerker bijvoorbeeld informatie uit een document wil invoeren in het “invoersysteem” van DSW, opent hij naast het invoersysteem ook nog de image viewer. Daarna zal hij Drovi moeten openen, vervolgens zoekt hij de documenten en als laatste wordt het gewilde document getoond in de image viewer. De informatie in de documenten die de medewerker nodig heeft zal hij dan kunnen invoeren in het invoersysteem.

3.2 Toekomstige Ontwikkelingen

De medewerkers van DSW werken met vele verschillende applicaties. Er zijn applicaties om bijvoorbeeld nieuwe klanten in te voeren, applicaties om het dossier van een klant te beheren en applicaties om de gescande documenten in te voeren in de database. Bij veel van de applicaties hebben de medewerkers informatie nodig uit de documenten. Naast de schermen van het systeem waarmee ze informatie kunnen bewerken, moeten ze ook vaak de image viewer openen om de documenten te bekijken. De medewerkers moeten altijd hun beeldscherm zo indelen dat alles overzichtelijk blijft en de documenten leesbaar zijn in de image viewer. Dit is vaak onhandig, tijdrovend en dus inefficiënt.

DSW wil in de nabije toekomst af van de losstaande image viewer. Het wil nu een image viewer die ingebouwd is in de bestaande schermen van elk systeem waarbij het nodig is om informatie uit de documenten te halen. Hierdoor zullen de medewerkers geen tijd meer verliezen met het telkens opnieuw indelen van hun beeldscherm.

De ICT-afdeling zal dus een image viewer control (component of plugin) moeten maken, die herbruikbaar is en ingebouwd kan worden daar waar het handig en noodzakelijk is.

3.3 Daadwerkelijke opdracht

Het doel van het project is om een usercontrol te ontwikkelen met minimaal de zelfde functionaliteiten als de huidige image viewer. De systemen die ontwikkeld worden voor DSW, worden bijna allemaal ontwikkeld met de programmeertaal C# van het .NET-omgeving. De image viewer control zal dan ook in C# geschreven moeten worden, om het integreren van de control mogelijk en makkelijk te maken.

De image viewer control zal in vierën worden gedeeld. Een van de redenen van het opsplitsen is om het ontwikkelen zo overzichtelijk mogelijk te houden en de functionaliteiten van elkaar te scheiden. Er zullen vier verschillende controls worden gemaakt die elk hun eigen functionaliteiten hebben en toevoegen aan het geheel. Elk control, los van de eerste, is gebaseerd op de vorige, waardoor bij voltooiing van een control een belangrijke functionaliteit toegevoegd wordt.

De *eerste control* zal de eigenlijke image viewer zijn met al zijn basisfuncties. Hierin zal een bitmap kunnen worden bekeken op verschillende manieren. Hierin kan een bitmap vergroot, verkleind, geroteerd en geïnverteerd worden.

Een document kan bestaan uit één of meer bitmaps. Het kunnen “scrollen” binnen een document naar een andere bitmap zal de *tweede control* mogelijk moeten maken.

Als een document of een selectie van documenten zal worden opgevraagd zal dat door Drovi worden gegeven in een map. Het kunnen selecteren van een document in een map zal gebeuren door de *derde control*.

De *vierde control* zal het Drovi-zoekscherm vervangen. Hierin zal één document of meer documenten kunnen worden opgezocht met de verschillende zoekcriteria.

Dat de controls allemaal worden ontwikkeld zal afhangen van de voortgang van het project. De prioriteit zal liggen bij de eerste control. Als deze helemaal voldoet zullen de andere controls één worden één, met de vorige als basis, worden ontwikkeld. Ook zullen deze apart worden geanalyseerd en ontworpen. Dit om ons volledig te concentreren op de eerste control. In dit rapport wordt vooral aandacht gegeven aan de eerste control.

De eerste control zal pas voldoen aan de eisen als een bitmap zonder enige flikkering van het beeldscherm kan verkleind, vergroot, geroteerd en geïnverteerd worden. Ook moet het vergroten en verkleinen zo worden gedaan dat de bitmap erna scherp en goed leesbaar blijft, zoals de huidige image viewer dat ook doet.

4. Algemene analyse

In dit hoofdstuk volgt een algemene analyse van de behoeften voor de eerste control.

4.1 Aanpak

Tijdens dit project zullen we een image viewer ontwikkelen. Dat het een control moet zijn en niet een applicatie maakt in principe niet heel veel uit voor een programmeur. Het belangrijkste van het project is toch dat een stukje programmatuur wordt ontwikkeld waarmee een bitmap kan worden getoond en deze met verschillende manieren kan worden bekeken.

We weten allemaal dat er vele van dit soort applicaties en usercontrols bestaan. We hebben tijdens ons vooronderzoek zelfs onderzoek gedaan naar een aantal van dit soort controls. Het concept is dus niet nieuw. Het tonen, zoomen, roteren of inverteren van een bitmap zijn geen functionaliteiten die nog moeten worden onderzocht en uitgevonden. Tijdens ons project zal het niet nodig zijn om dit te doen. De algoritmes bestaan al en zijn al in vele programmeertalen geschreven. Tijdens ons project zullen we ons meer concentreren op het onderzoeken van de algoritmes en de programmatuur die al bestaan.

De functionele requirements van ons project komen erg overeen met de requirements die de huidige losstaande image viewer heeft die nu wordt gebruikt in DSW. De algoritmes in deze image viewer zullen wij dus ook onderzoeken. Het enige nadeel is dat deze is geschreven in de programmeertaal Visual Basic 6.0. Wij zullen dus ook bestuderen hoe Visual Basic werkt, zodat wij de algoritmes kunnen begrijpen.

Om het begrijpen van de code makkelijker te maken is er een mogelijkheid om de code van de image viewer te converteren naar C#. Op het Internet zijn vele converters beschikbaar.

4.2 De Image Viewer

De huidige image viewer bevat alle basis functies als de te bouwen control, het eerste control. De belangrijkste en ingewikkeldste daarvan is het *zoomen* en de *bicubic interpolation*, welke zeer efficiënt zijn geïmplementeerd.

5. Gebruikersanalyse

In dit hoofdstuk wordt de gebruikers geanalyseerd. Bij de analyse zijn de gebruikers van het uiteindelijk geïntegreerde *image viewer* en hun karakteristieken geïdentificeerd. Omdat deze zaken invloed hebben op het ontwerp en implementatie van het uiteindelijke product, is dit van belang. Voor deze analyse wordt een *gebruikersprofiel* opgesteld.

Proces

Om een zo goed mogelijke gebruikersanalyse uit te voeren, dienen een aantal stappen uitgevoerd te worden. Als eerste stap dienen de gebruikers geïdentificeerd te worden. Dit wordt gedaan middels gesprekken met de opdrachtgever (*zie bijlage A*). Door het stellen van vragen aan die individuen vindt het opstellen van het gebruikersprofiel plaats. Een samengevat resultaat van dit gesprek is in tabelvorm te vinden in *bijlage B*. Aan de hand van deze tabellen kan een gebruikersprofiel opgesteld worden.

Gebruikers

Na een gesprek met de opdrachtgever werd duidelijk wie de eindgebruikers van het uiteindelijke *image viewer* zullen zijn. Over het algemeen wordt de huidige *image viewer* gebruikt door medewerkers van de afdelingen VZA (Verzekerden administratie) en VSA (Verstrekingen administratie). Deze gebruiken de *image viewer* voor het zelfde doel, namelijk het inzien van gescande documenten. Daarom kunnen zij onder één categorie van gebruikers geplaatst worden.

Gebruikersprofiel

Het gebruikersprofiel is een belangrijke methodologie voor de gebruikersanalyse. Volgens Mayhew geeft het gebruikersprofiel een overzicht van relevante kenmerken van de gebruikers. Daarnaast kan het een indicatie geven van de voorkeuren van de gebruiker. Ook kan het van nut zijn voor evaluatie. Het gebruikersprofiel geeft de persoonlijke kenmerken, fysieke kenmerken, ervaring en soort gebruik van de gebruiker aan [3]. Na analyse van de antwoorden gegeven op de gestelde vragen (*zie Bijlage B*) vindt het opstellen van het gebruikersprofiel plaats. Zij zullen als fysieke gebruikers direct gebruik maken van de *image viewer*.

In Tabel ‘*Psychologische karakteristieken*’ uit bijlage A, staat weergegeven dat de gebruikers een redelijk vermogen hebben om zelfstandig nieuwe dingen te leren. Ze zien graag verbetering in huidige systemen willen, maar bij al te veel verandering hebben zij weinig motivatie.

In Tabel ‘*Kennis & ervaring*’ staat weergegeven dat de meeste gebruikers een gemiddelde opleiding hebben genoten. Ook is er goede ervaring met het gebruik van applicaties en worden redelijk wat applicaties gebruikt bij de uitvoering van taken. De computervaardigheid van de gebruikers is hoog en de moedertaal hoofdzakelijk Nederlands.

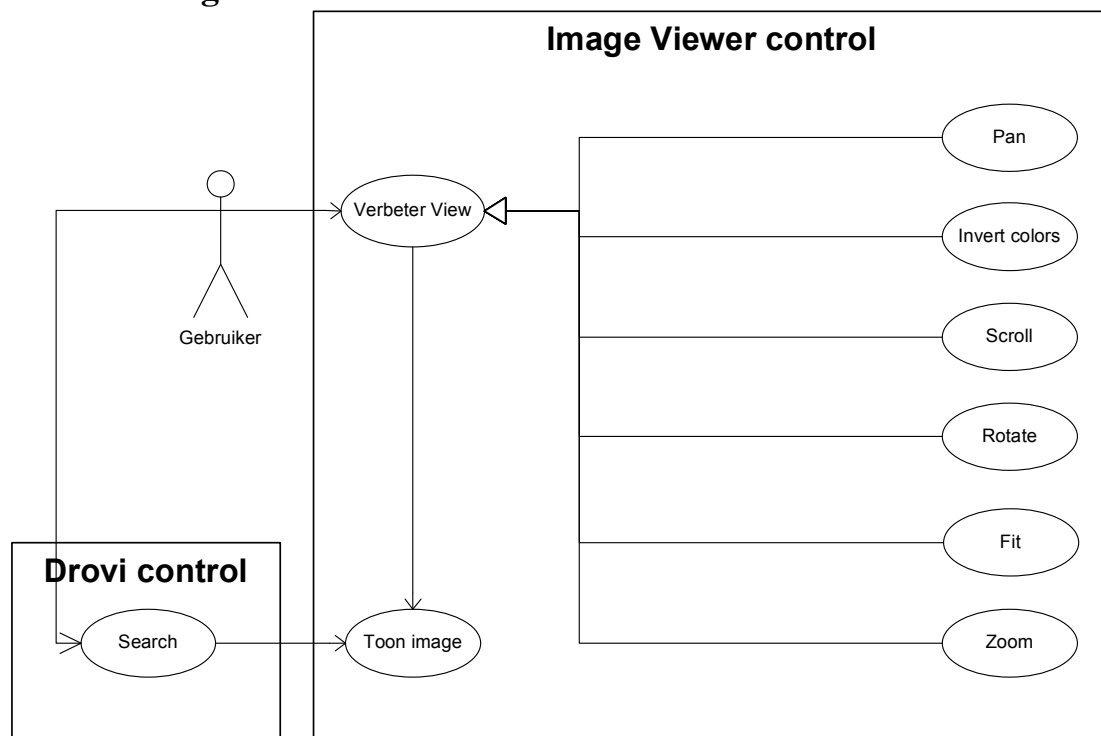
In Tabel ‘*Werk & taak karakteristieken*’ staat weergegeven dat de gebruikers de *image viewer* heel vaak gebruiken. De basistraining, bij de *image viewer* meer een introductie, was goed. Ook geven de gebruikers aan naast de *image viewer* ook een aantal andere tools te gebruiken.

In Tabel ‘*Fysische karakteristieken*’ staat weergegeven dat onder de gebruikers de meeste niet kleurenblind, rechtshandig en van het vrouwelijke geslacht zijn.

6. Use case analyse

In dit hoofdstuk wordt een overzicht gegeven van de use cases van de *image viewer control*. Een use case diagram wordt ook gebruikt. Use case diagrammen worden gebruikt om functionele eisen te achterhalen. Elke use case laat een interactie tussen de gebruiker en het systeem zien. Eerst wordt een use case diagram weergegeven, vervolgt door een beschrijving van de use cases. Deze zijn geformuleerd na *observatie* van de huidige image viewer én het *interviewen* van de eindgebruikers. Zie *bijlage C* voor de interviewvragen.

Use case diagram



Figuur 3: Use case diagram Image Viewer control

Use case beschrijvingen

Use Case: Toon nieuwe image

Actor: Gebruiker

Beginsituatie: Gebruiker dient een polisblad te bekijken

6. De gebruiker zoekt via Drovi het betreffende polisblad
7. Drovi zoekt de image van het blad op, laadt het en geeft het door aan de image viewer control
8. De image viewer control laat het polisblad zien.

Eindsituatie: De image van het polisblad is te zien

Use Case: Verander de view op de image

Actor: Gebruiker

Beginsituatie: Er is een image getoond

3. De gebruiker kiest een bewerking. Er kan gekozen worden uit:
 - a. Pan: Met de muis de image verschuiven
 - b. Invert Colors: Alle kleuren van de image inverteren
 - c. Scroll: De image horizontaal of verticaal verschuiven
 - d. Rotate: De image roteren
 - e. Fit: De grootte van de image aanpassen aan de hand van 4 verschillende modes:
 - Height: De image wordt over de hele hoogte zichtbaar
 - Widht: De image wordt over de hele breedte hoogt zichtbaar
 - Full: De image wordt in zijn geheel (hoogte en breedte) zichtbaar
 - None: De image wordt in zijn normale grote weergegeven
 - f. Zoom: Op de image in- of uitzoomen
4. De image viewer control voert de bewerking uit en toont de bewerkte image

Eindsituatie: De image wordt zoals gewenst getoond

7. Requirements

In dit hoofdstuk wordt een overzicht van de functionele- en non-functionele requirements gegeven. De functionele eisen worden in paragraaf 7.1 ingedeeld volgens het MoSCoW model en in paragraaf 7.2 volgen de non-functionele requirements

MoSCoW model

De functionele requirements zullen via het MoSCoW model worden weergegeven. Het MoSCoW model is een veel gebruikte methode voor het verdelen van de requirements ofwel eisen naar prioriteit. De verschillende prioriteiten hierbij zijn achtereenvolgens:

- Must have (Mo):** zijn absoluut *noodzakelijk* om te implementeren
- Should have (S):** *zouden* in principe geïmplementeerd moeten worden
- Could have (Co):** worden alleen geïmplementeerd indien er tijd over is
- Want to have (W):** mooi om te hebben voor opdrachtgever

7.1 Functionele requirements

In deze paragraaf worden de functionele eisen voor de image viewer control opgesomd. Dit is gedaan volgens het MoSCoW model.

Must have

4. Een gegeven bitmap image moet getoond kunnen worden
5. Op een getoonde bitmap image moet ingezoomd kunnen worden
6. Een getoonde bitmap image moet geroteerd kunnen worden
7. De kleuren van een getoonde bitmap image moeten geïnverteerd kunnen worden.
8. Met de pijltjestoetsen moet gescrold kunnen worden met de bitmap.
9. De getoonde bitmap moet, met behulp van 4 knoppen, kunnen worden bekeken met de vier fitmodes: height, width, full & none.
 - height - de bitmap image moet in zijn gehele lengte getoond kunnen worden naar mate van de gehele lengte van de viewer.
 - width - de bitmap image moet in zijn gehele breedte getoond kunnen worden naar mate van de gehele breedte van de viewer.
 - full - de bitmap image moet in zijn geheel getoond kunnen worden naar mate de maximale grootte van de viewer.
 - none - de bitmap image wordt getoond naar wensen van de gebruiker.

Should have

10. Slepen met de linkermuisknop in het document om het te verplaatsen (Pannen).

Could have

11. Functie balk die ingesteld kan worden op “automatisch verbergen”
12. Functie balk die verplaatst kan worden

7.2 Non-functionele requirements

In deze paragraaf volgen de niet functionele requirements. Deze zijn onderverdeeld in *quality*, *platform* en *process* requirements.

Quality requirements

- *Reliability*
Onder reliability van een systeem wordt verstaan de tijd dat zo'n systeem stil mag liggen door fouten. Voor de *image viewer control* zou de reliability 99,9% moeten zijn. Met andere woorden 999 uit de 1000 keer zal de control niet falen.
- *Availability*
Dit is de tijd dat een systeem beschikbaar moet zijn. In zijn totaliteit moet de image viewer control 99,9% van de tijd beschikbaar zijn.
- *Performance*
Dit is de kwaliteit van een systeem in zijn geheel. De viewer moet onder elke omstandigheid flikkervrij zijn.
- *Response time*
De tijd die de *gebruiker* moet wachten op resultaat van een bewerking of functie, nadat hij/zij de functie heeft laten uitvoeren. Responstijd zou kleiner kunnen zijn dan *een seconde*, gemeten vanaf het moment dat de gebruiker de functie laat plaatsvinden, tot het moment dat het resultaat beschikbaar is.
- *Reusability*
Dit is het percentage van een systeem dat herbruikbaar zou moeten zijn. Het te bouwen control moet integreerbaar zijn in meerdere systemen, waardoor het 100% reusable moet zijn. Het feit dat het een control is, maakt het 100% reusable.

Platform requirements

- *Computing platform*
De minimale eisen waarom het systeem gedraaid kan worden. Voor de image viewer control geldt als minimale systeem eisen:
Processor: Intel Pentium 4, 2.0 GHz
Werkgeheugen: 512 MB
Videokaart: geïntegreerd
Beeldscherm: 18" (1280x1024)
Operating system: Windows XP
- *Technology to be used*
De systemen waarin het te bouwen control geïntegreerd zal worden, zullen worden geschreven in C#. Voor de control is het van belang dat het coderen ook in C# gebeurt. Voor het zoomen dient gebruik gemaakt te worden van *bicubic interpolation* of een soortgelijke methode voor het verbeteren van een vergrote image.

Process requirements

- *Development process (methodology)*
Het ontwerp en de analyse worden gedocumenteerd in UML. Het testen wordt in units gedaan.

Literatuurlijst

- [1] DSW, Personeelshandboek
- [2] Lethbridge, T.C., Laganière, R. (2005), “Object-oriented software engineering”, 2nd edition, New York, McGraw-Hill Education
- [3] Mayhew, D.J. (1992), “Principles and guidelines in software user interface design”, Prentice Hall

Bijlage A: Vraaggesprekken

Vraaggesprekken

Eerste gesprek

Ondervraagde: Job Scheffers

Datum: 01-07-2008

Tijd: 10:00 – 10:45

Locatie: ICT-afdeling DSW

Tweede gesprek

*Hierbij was ook Robert Buitendijk aanwezig,
een programmeur binnen DSW.*

Ondervraagde: Job Scheffers

Datum: 16-07-2008

Tijd: 14:45 – 16:00

Locatie: ICT-afdeling DSW

Algemene & Non-functionele vragen:

Wat voor documenten of bitmaps worden er precies met de image viewer bekeken?

Een zorgverzekeraar verstuurt en krijgt veel post binnen. Dit zijn meestal brieven van en naar klanten en ziekenhuizen. Vaak zijn dit ook rekeningen van klanten of ziekenhuizen. Al deze documenten moeten door verschillende medewerkers worden bekeken en is het dus handig deze allemaal digitaal op te slaan, zodat iedereen binnen DSW alle documenten kan opvragen. De documenten worden dan gescand en in een database opgeslagen als bitmap-bestand. Deze documenten moeten dus worden kunnen bekeken met de image viewer.

Waarom maakt DSW geen gebruik van bestaande commerciële software, zoals bijvoorbeeld Acrobat Reader?

De medewerkers van DSW maken al gebruik van een bestaande image viewer. Deze is echter een losstaande en geschreven in Visual Basic. Wat DSW wil is een image viewer control. Dat is een soort plugin die een programmeur kan toevoegen in een graphical user interface van een applicatie. DSW wil een image viewer control die we kunnen toevoegen in applicaties waar we dat nodig hebben. De applicaties, die kunnen worden voorzien van een image viewer control, worden geschreven in C#. Het is dus beter om de image viewer control ook in C# te schrijven. En omdat deze niet bestaat met de requirements van DSW, is het noodzakelijk om deze te laten ontwikkelen.

Waarom wil DSW zonodig een ingebouwde viewer in hun systeem?

Voor een medewerker van DSW die werkt met een applicatie en ook nog een aparte losstaande image viewer, is het onhandig werken met 2 verschillende windows. Elke keer maar weer moet de medewerker de windows vergroten of

verkleinen om zijn scherm overzichtelijk te houden. Met een ingebouwde image viewer in de applicatie zou dat niet meer nodig zijn.

Welke afdelingen gaan gebruik maken van de control?

Op de VSA-afdeling (verstrekking administratie) en de VZA-afdeling (verzekerden administratie). Ook op de ICT-afdeling waar de control zal worden ingebouwd in de applicaties bij welke die nodig zijn.

In welke applicaties wordt de image viewer control ingebouwd?

Kan in verschillende applicaties worden ingebouwd als dat nodig is. Scanapplicatie, zoekapplicatie, invoerapplicatie en nog te ontwikkelen applicaties in de toekomst.

Wat voor methodiek moeten we gebruiken voor het ontwikkelen van de software?

Mogen jullie zelf bepalen. (Wij zijn bekend met het gebruik van UML, dus zullen we werken volgens deze methodiek)

Welke documenten verwacht u als opdrachtgever? Zijn er bepaalde vormen van documentatie gewenst (buiten die door de TU verplicht zijn gesteld)?

Ten eerste verwachten wij dat de programmeercode goed gedocumenteerd wordt. Dat niet alleen wordt uitgelegd wat er wordt gecodeerd, maar ook waarom. Ten tweede willen wij een specificatie van de methodes die worden gebruikt. Dit voor de programmeur van DSW die de control gaan inbouwen in de verschillende systemen. Als laatste nog een simpele gebruikershandleiding voor de eindgebruikers.

Wie zullen er meebeslissen over het uiterlijk van de user-interface?

Begeleider Job Scheffers en de eindgebruikers.

Kunnen eindgebruikers worden benaderd om de control te testen?

Ja, dat is mogelijk.

Funtionele vragen:

Welke functionaliteiten moet de image viewer control hebben?

Het moet een bitmap kunnen tonen, zoomen met de factor 10 tot 1800% mbv bicubic interpolation, een rectangle kunnen zoomen, 90° en -90° roteren, kleuren kunnen inverteren, hebben van vier verschillende fitmodes: full, height, width en none. Deze mogelijkheden moeten allemaal flikkervrij zijn

onder alle omstandigheden. Ook het gebruik van scrollbars als ze nodig zijn, PgUp en PgDn voor het verticaal scrollen en de mouse-wheel voor het in- en uitzoomen.

Bij het zoomen, moeten we gebruik maken van bicubic interpolation. Als er andere methodes voldoen aan de eisen, kan daar dan gebruik van worden gemaakt?

Het maakt in principe niets uit welke methode er wordt gebruikt tijdens het zoomen. Als de tekst in de documenten goed leesbaar blijft na het in- en uitzoomen dan is DSW tevreden.

Met de image viewer die nu wordt gebruikt, kunnen meerdere documenten worden bekeken. Na een interview met de eindgebruikers is gebleken dat het voor hen voldoende is één document tegelijk te kunnen openen. Vindt u dat ook goed?

Voor DSW is de eerste control van een veel grotere belang dan de controls die daarna kunnen worden ontwikkeld, omdat die al voor genoeg moeilijkheden zorgt. Het is beter om alles wat te maken heeft met volgende controls op de lange baan te schuiven en eerst een goed werkend eerste control te ontwikkelen.

In de control daarna kan er een mogelijkheid gebouwd worden die meerdere documenten opent met het gebruik van tabbladen.

Voor de tweede en derde stap, moeten wij alleen de user interface maken? Dus de programmeurs zullen de verbinding maken met de database en de eventhandlers implementeren?

Ook hier geldt wat er in de vorige vraag is gezegd. Als het zo ver is dat de volgende controls worden ontwikkeld, zullen er weer een aantal gesprekken volgen om het een en ander te verduidelijken.

Maar voor behalve de laatste control, zullen jullie niks hoeven te doen met de database. Alle input voor de eerste controls zullen worden geleverd door de programmeurs van DSW.

Als de tijd meezit zullen er een aantal controls worden ontwikkeld die in elkaar worden geplaatst. Zullen de controls in bepaalde gevallen ook los van elkaar worden geplaatst in applicaties?

Nee, dat zal niet gebeuren. Het gebruik van de verschillende controls is om de functionaliteiten te scheiden in verschillende lagen. Hierdoor zal er een beter overzicht zijn tijdens het ontwikkelen.

Na een interview met de eindgebruikers is gebleken dat het gebruik van thumbnails onhandig is. Moeten we die nog steeds erin bouwen?

Als de eindgebruikers zeggen dat ze het niet zullen gebruiken, dan is het niet nodig voor jullie om het te ontwikkelen.

Veranderingen en gemaakte afspraken:

Naar aanleiding van het gesprek met de eindgebruikers hebben we tijdens het tweede gesprek met Job Scheffers en Robert Buitendijk een aantal veranderingen toegepast die betrekking hebben met de verschillende controls. Dit omdat de oorspronkelijke tweede control is komen te vervallen en omdat de oorspronkelijke opdrachtbeschrijving geen rekening hield met het mappensysteem van Drovi.

Hier nu een beschrijving van wat is afgesproken wat de verschillende controls moeten kunnen. In plaats van drie zijn er nu vier verschillende controls.

De eerste control is niet veranderd van de oude opdrachtbeschrijving. Hierin zal gewoon een bitmap kunnen worden bekeken op verschillende manieren. Hierin kan een bitmap vergroot, verkleind, geroteerd en geïnventeerd worden.

Een document kan bestaan uit één of meer bitmaps. Het kunnen “scrollen” binnen een document naar een andere bitmap zal de tweede control mogelijk moeten maken.

Als een document of een selectie van documenten zal worden opgevraagd zal dat door Drovi worden gegeven in een map. Het kunnen selecteren van een document in een map zal gebeuren door de derde control.

De vierde control zal het Drovi-zoekscherm vervangen. Hierin zal één document of meer documenten kunnen worden opgezocht met de verschillende zoekcriteria.

Er is afgesproken dat de hoogste prioriteit ligt bij de eerste control. En als er dan tijd overblijft, werken we aan de andere controls. Voor wat precies de nieuwe controls moeten doen zullen dan eventueel nog meer gesprekken volgen.

Bijlage B: Gebruikersprofiel (ingevuld)

Psychologische karakteristieken

Cognitieve stijl	Redelijk
Attitude	Goed
Motivatie	Laag (bij verandering)

Kennis & ervaring

Leesniveau	Hoog
Typvaardigheid	Hoog
Educatie	HAVO/MBO
Systeem ervaring	Goed
Taak ervaring	Goed
Applicatie ervaring	Heel goed
Moedertaal	Nederlands
Gebruik van andere systemen	Veel
Computervaardigheid	Goed

Werk & taak karakteristieken

Frequentie van het gebruik van het systeem	Heel veel
Basis training (algemeen)	Goed
Systeemgebruik	Gemiddeld
Andere tools	Goed

Fysische karakteristieken

Kleurblindheid	Niet aanwezig
Links/Rechtshandig	Rechtshandig
Geslacht	V

Bijlage C: Vragenlijst gebruikers

Op dit moment is er een Image viewer control in ontwikkeling. U bent een van de eindgebruikers van een geïntegreerde viewer. Om een, voor u efficiënt mogelijk image viewer te bouwen, gaarne de volgende vragen beantwoorden.

1) Bekijk de huidige image viewer. Welke functies zou u terug willen zien in de nieuwe?

- a) Allemaal
- b) Enkele, nl:
 - Zoom (selectie)
 - Zoom (button)
 - Fit-modes (height, width, full & none)
 - Rotatie
 - Invert colors
 - Printen
 - Anders, nl.

**2) Welke nieuwe functies lijken u wel handig als toevoeging?
Daarbij kunt u denken aan:**

- Zoomen met behulp van de Mouse Wheel
- Scrollen m.b.v pijltjes- en PgUp & PgDn toetsen
- Anders nl.

3) Betreffende de GUI, heeft u een bepaalde voorkeur voor de plaatsing van de buttons?

- a) Ja, boven - onder - links - rechts - zwevende balk
- b) Nee
- c) Ik prefereer een soortgelijke lay-out als de huidige image viewer

4) Heeft u een voorkeur voor button grootte?

- a) Ja, ongeveer
- b) Nee
- c) Ik prefereer als button grootte dezelfde als in de huidige viewer

**5) Als uitbreiding zal de viewer een array van bitmaps ontvangen. Van die array zou een geselecteerde getoond moeten worden.
Hoe zou u zo'n array getoond willen hebben?**

- a) een lijst van thumbnails met daarboven de geselecteerde (de windows filmstrip view)
- b) Geen thumbnails maar een image selecteren via een menu (namen in een drop down box)

- c) De image selecteren uit een lijst met namen links of rechts van de main image box
- d) Anders, nl.

7)Heeft u nog suggesties voor de nieuwe image viewer?

Hartelijk dank voor uw medewerking.

Bijlage E: RAD navigatie control

BSc-project IN3405



Requirements Analysis Document Navigatie control

Omgeving:

Zorgverzekeraar DSW
Technische Universiteit Delft
Faculteit EWI Technische Informatica
Variant Softwaretechnologie (ST)

Projectgroep:

Abdoelrasheed Jarmohamed
Kishen Gajadhar

Commissie:

Job Scheffers (Zorgverzekeraar DSW, begeleider)
B.R. Sodoyer (TU Delft, coördinator / begeleider)

Datum: 15 september 2008

Status: wachten op acceptatie

Versie: 1.0

©September 2008 Zorgverzekeraar DSW

Alle in dit verslag afgebeelde gegevens, waaronder teksten, foto's, illustraties, grafische materiaal, (handels)namen, logo's, waren- en dienstmerken, zijn in eigendom van zorgverzekeraar DSW en worden beschermd door auteursrecht, merkenrecht en/of enig ander intellectueel eigendomsrecht.

Inhoudsopgave

1. Introductie	98
2. Daadwerkelijke opdracht	99
3. Algemene analyse	100
4. Use case analyse	102
5. Requirements	103
5.1 Functionele requirements	103
5.2 Non-functionele requirements	104
Bijlage A: Vraaggesprek	105

1. Introductie

Dit analyse rapport wordt als analyse voor de navigatie control geschreven. Het is hierbij belangrijk dat het analyse rapport voor de eerste control, de *viewer control*, ook gelezen is.

Tijdens deze analyse is het ten eerste de bedoeling informatie te verzamelen over het navigatie control. Dit gaat aan de hand van *interviews* en *observatie*. Het doel van het verzamelen van informatie is duidelijk: zonder informatie kan er geen overwogen keus gemaakt worden over het verloop van het project. Door middel van deze analyse kan het probleemdomein duidelijk gedefinieerd en afgebakend worden tot een eenduidig probleemdomein waarna een softwaresysteem zal kunnen worden ontworpen om dit probleem op te lossen. Omdat dit domein dezelfde is als dat voor de *viewer control*, wordt voor een probleem domein verwezen naar het RAD van de viewer control.

Nadat de informatie is verworven wordt de informatie geanalyseerd. Hierbij kan gebruik gemaakt worden van gebruikersanalyses, welke ook te vinden is in het RAD van de viewer control, use-case diagrammen en een lijst van de requirements. Deze analyse is nodig om uit te zoeken welke informatie nodig is voor het realiseren van de control en om uit te zoeken wat wel en niet haalbaar of wenselijk is voor deze control. Hierdoor kan het project gestructureerd worden uitgevoerd en kan er een goed werkend systeem ontworpen worden.

De opbouw van dit rapport is als volgt. In hoofdstuk 2 wordt de daadwerkelijke opdracht herzien gevolgd door hoofdstuk 3, waar een algemene analyse gemaakt van de toe te voegen functionaliteit in de oude image viewer. Vervolgens wordt in hoofdstuk 4 de use case analyse besproken gevolgd door de lijst van requirements in hoofdstuk 5.

2. Daadwerkelijke opdracht

Het doel van het project is om een usercontrol te ontwikkelen met minimaal de zelfde functionaliteiten als de huidige image viewer. De systemen die ontwikkeld worden voor DSW, worden bijna allemaal ontwikkeld met de programmeertaal C# van het .NET-omgeving. De image viewer control zal dan ook in C# geschreven moeten worden, om het integreren van de control mogelijk en makkelijk te maken.

De image viewer control zal in vieren worden gedeeld. Een van de redenen van het opsplitsen is om het ontwikkelen zo overzichtelijk mogelijk te houden en de functionaliteiten van elkaar te scheiden. Er zullen vier verschillende controls worden gemaakt die elk hun eigen functionaliteiten hebben en toevoegen aan het geheel. Elk control, los van de eerste, is gebaseerd op de vorige, waardoor bij voltooiing van een control een belangrijke functionaliteit toegevoegd wordt.

De *eerste control* zal de eigenlijke image viewer zijn met al zijn basisfuncties. Hierin zal een bitmap kunnen worden bekeken op verschillende manieren. Hierin kan een bitmap vergroot, verkleind, geroteerd en geïnverteerd worden.

Een document kan bestaan uit 0 of meer bitmaps gevolgd door 0 of meer bijlagen. Het kunnen navigeren binnen een document naar een andere bitmap, of bijlage zal de *tweede control* mogelijk moeten maken.

Als een document of een selectie van documenten zal worden opgevraagd zal dat door Drovei worden gegeven in een map. Het kunnen selecteren van een document in een map zal ook mogelijk gemaakt worden door de *tweede control*.

Documenten kunnen andere dan image bestanden bevatten. Deze kunnen niet worden getoond in de image viewer, dus zal de *derde control* zorgen het viewen van niet image bestanden.

De *vierde control* zal het Drovei-zoekscherm vervangen. Hierin zal één document of meer documenten kunnen worden opgezocht met de verschillende zoekcriteria.

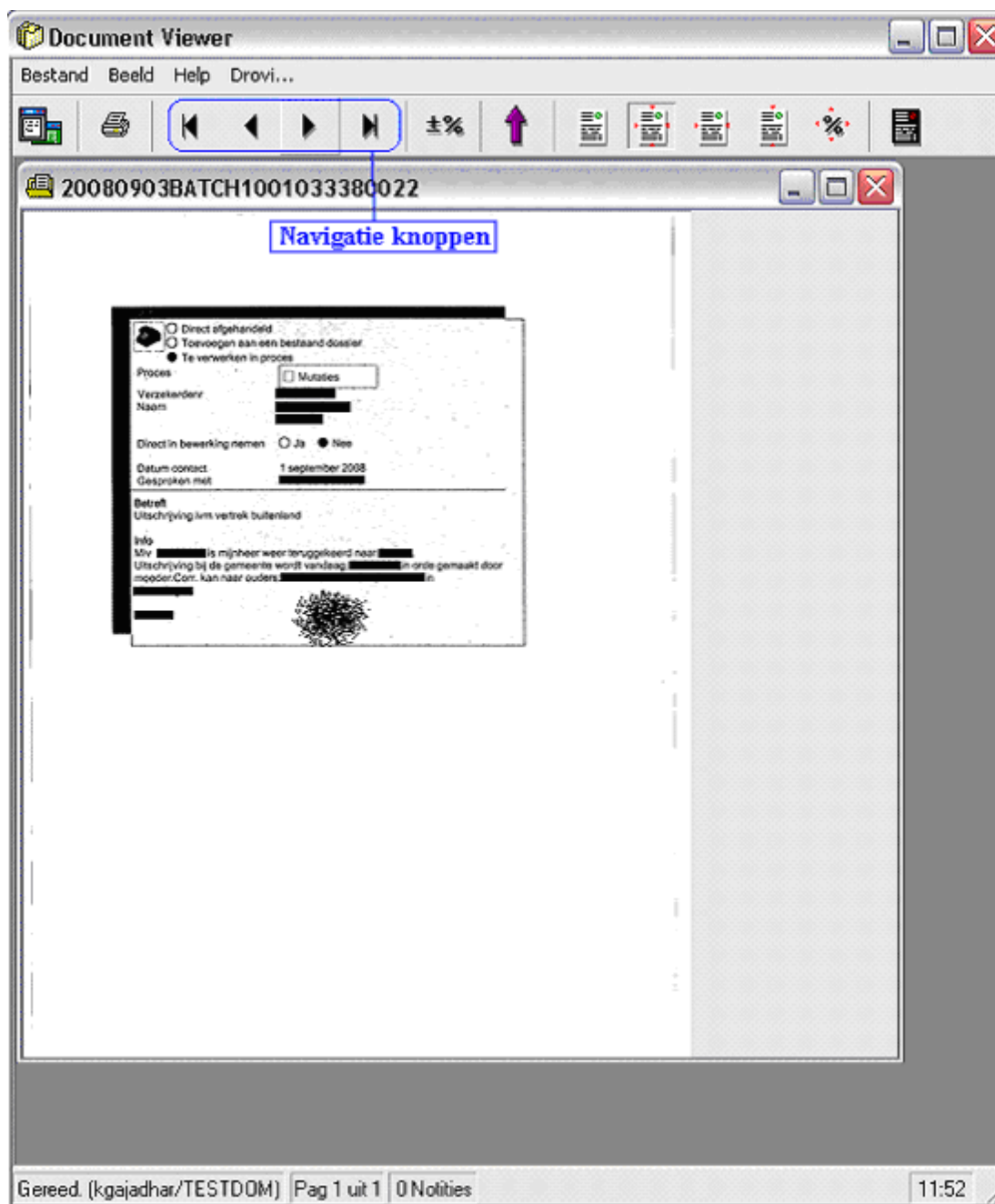
De tweede control zal pas voldoen aan de eisen als door en tussen een hele rij van documenten genavigeerd kan worden.

3. Algemene analyse

In dit hoofdstuk volgt de algemene analyse voor de navigatie control. Eerst zal gekeken worden hoe de toe te voegen functionaliteit in het huidige systeem zijn verwerkt. Vervolgens worden die functies gedetailleerd besproken.

De huidige situatie

Binnen het huidige image viewing systeem kan een invoer van meer dan één image verwacht worden. De gebruiker heeft dan de mogelijkheid tussen die images te navigeren, zie *figuur 1*



Figuur 1: Navigatie knoppen in de huidige image viewer

Zoals in *figuur 1* te zien is, zijn er vier navigatie knoppen. Van links naar rechts zijn die: *Begin document*, *Vorige pagina*, *Volgende pagina* en *Einde document*. Als binnen zo'n venster meer dan een document getoond dient te worden, komen er twee knoppen bij, namelijk: *Vorig document* en *Volgend document*. Gebruikers kunnen met de huidige image viewer navigeren zoals de naam van de genoemde knoppen suggereren.

De toekomstige situatie

Om uit te leggen wat gewild is met de navigatie control is het handig eerst het structuur van de document typen te belichten.

Binnen DSW worden de images gestructureerd in documenten. Een *document* bevat een *imprint nummer*, dat als id. fungeert, en een rij van 0 of meerdere images, die de *pagina's* van zo'n document moeten voorstellen. Ook bevat een document een rij van 0 of meerdere *bijlagen* of attachments. Een bijlage kan elk type bestand zijn, dus ook MSOffice, PDF en Image bestanden.

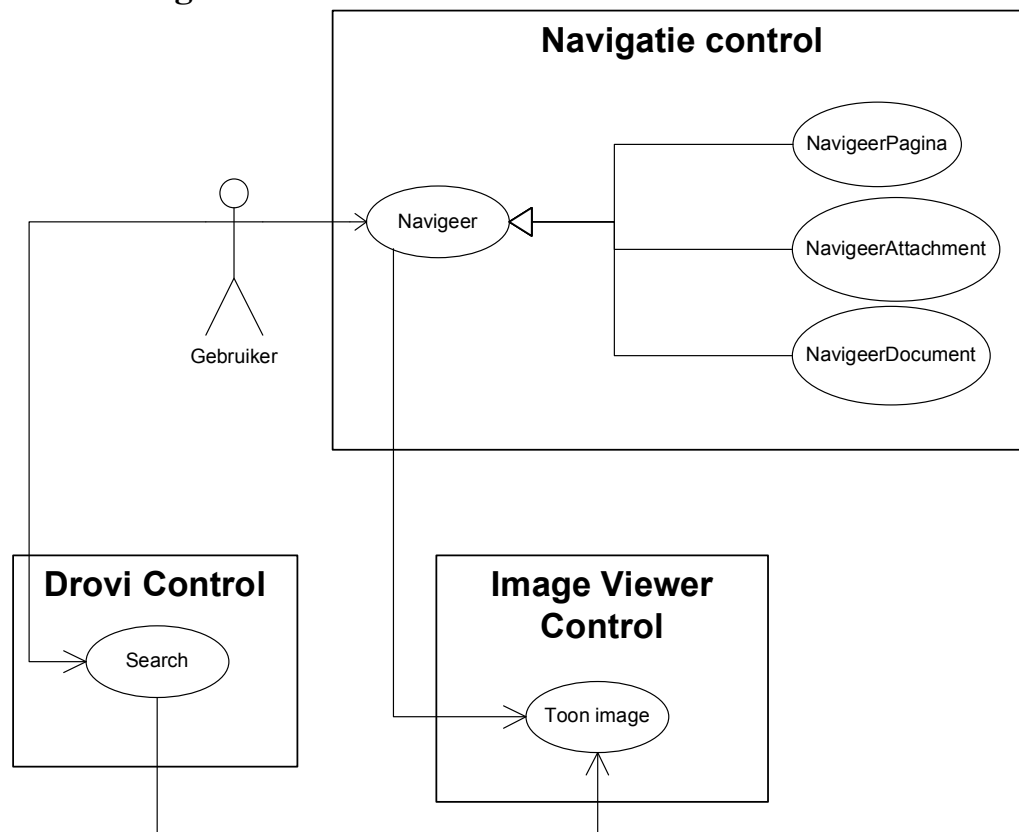
De navigatie control zal alleen image bestanden afhandelen, dus wordt aangenomen dat iedere bijlage bestaat uit een rij van images, die net als bij een document, de *pagina's* van zo'n bijlage moeten voorstellen.

De navigatie control kan een rij van documenten als invoer verwachten en moet het dus mogelijk maken tussen en door die documenten te navigeren.

4. Use case analyse

In dit hoofdstuk wordt een overzicht gegeven van de use cases van de *navigatie control*. Een use case diagram wordt ook gebruikt. Use case diagrammen worden gebruikt om functionele eisen te achterhalen. Elke use case laat een interactie tussen de gebruiker en het systeem zien. Eerst wordt een use case diagram weergegeven, vervolgt door een beschrijving van de use cases. Deze zijn geformuleerd na *observatie* van de huidige image viewer én het *interviewen* van systeem ontwikkelaars en de opdrachtgever. Het benaderen van eindgebruikers werd in dit geval achterwege gelaten. Zie *bijlage A* voor de interviewvragen.

Use case diagram



Figuur 3: Use case diagram Image Viewer control

Use case beschrijvingen

Hier volgt ons enige use case voor dit control, immers dit is ook de enige functionaliteit die wordt toegevoegd met dit control.

Use Case: navegeer

Actor: Gebruiker

Beginsituatie: Er is een array van documenten (images) beschikbaar, waarvan 1 image getoond is. De gebruiker wil een ander image, behorende bij de array van documenten zien.

1. De gebruiker drukt op de knop met de gewenste navigatie mogelijkheid. Gekozen kan worden uit:
 - a. Eerste pagina (van het eerste document uit de array van documenten)
 - b. Laatste pagina (van het laatste document uit de array van documenten)
 - c. Volgende pagina
 - d. Vorige pagina
 - e. Volgend subdocument (attachment)
 - f. Vorig subdocument (attachment)
 - g. Volgend document
 - h. Vorig document
2. Het control zoekt het gewenste image en geeft het aan de image viewer control om het te tonen.

Eindsituatie: Het gewenste image is te zien in de image viewer.

5. Requirements

In dit hoofdstuk wordt een overzicht van de functionele- en non-functionele requirements gegeven. De functionele eisen worden in paragraaf 7.1 ingedeeld volgens het MoSCoW model. De non-functionele eisen zijn de zelfde als voor de *viewer control*, maar voor het gemak worden die in paragraaf 7.2 herhaald.

MoSCoW model

De functionele requirements zullen via het MoSCoW model worden weergegeven. Het MoSCoW model is een veel gebruikte methode voor het verdelen van de requirements ofwel eisen naar prioriteit. De verschillende prioriteiten hierbij zijn achtereenvolgens:

- Must have (Mo):** zijn absoluut *noodzakelijk* om te implementeren
Should have (S): *zouden* in principe geïmplementeerd moeten worden
Could have (Co): worden alleen geïmplementeerd indien er tijd over is
Want to have (W): mooi om te hebben voor opdrachtgever

5.1 Functionele requirements

In deze paragraaf worden de functionele eisen voor de navigatie control opgesomd. Dit is gedaan volgens het MoSCoW model.

Must have

1. er moet genavigeerd kunnen worden tussen de daadwerkelijke pagina's (images) van een document.
2. er moet genavigeerd kunnen worden tussen de attachments van een document.
3. er moet genavigeerd kunnen worden tussen een rij van documenten.

Should have

4. navigatie knoppen moeten alleen worden getoond als de betreffende navigatie mogelijk is.

5.2 Non-functionele requirements

In deze paragraaf volgen de niet functionele requirements. Deze zijn onderverdeeld in *quality*, *platform* en *process* requirements. Zoals eerder gezegd zijn deze dezelfde als voor de *viewer control*.

Quality requirements

- *Reliability*
Onder reliability van een systeem wordt verstaan de tijd dat zo'n systeem stil mag liggen door fouten. Voor de *image viewer control* zou de reliability 99,9% moeten zijn. Met andere woorden 999 uit de 1000 keer zal de control niet falen.
- *Availability*
Dit is de tijd dat een systeem beschikbaar moet zijn. In zijn totaliteit moet de image viewer control 99,9% van de tijd beschikbaar zijn.
- *Performance*
Dit is de kwaliteit van een systeem in zijn geheel. De viewer moet onder elke omstandigheid flikkervrij zijn.
- *Response* *time*
De tijd die de *gebruiker* moet wachten op resultaat van een bewerking of functie, nadat hij/zij de functie heeft laten uitvoeren. Responstijd zou kleiner kunnen zijn dan *een seconde*, gemeten vanaf het moment dat de gebruiker de functie laat plaatsvinden, tot het moment dat het resultaat beschikbaar is.
- *Reusability*
Dit is het percentage van een systeem dat herbruikbaar zou moeten zijn. Het te bouwen control moet integreerbaar zijn in meerdere systemen, waardoor het 100% reusable moet zijn. Het feit dat het een control is, maakt het 100% reusable.

Platform requirements

- *Computing* *platform*
De minimale eisen waarom het systeem gedraaid kan worden. Voor de image viewer control geldt als minimale systeem eisen:
Processor: Intel Pentium 4, 2.0 GHz
Werkgeheugen: 512 MB
Videokaart: geïntegreerd
Beeldscherm: 18" (1280x1024)
Operating system: Windows XP
- *Technology to be used*
De systemen waarin het te bouwen control geïntegreerd zal worden, zullen worden geschreven in C#. Voor de control is het van belang dat het coderen ook in C# gebeurt. Voor het zoomen dient gebruik gemaakt te worden van *bicubic interpolation* of een soortgelijke methode voor het verbeteren van een vergrote image.

Process requirements

- *Development process (methodology)*
Het ontwerp en de analyse worden gedocumenteerd in UML. Het testen wordt in units gedaan.

Bijlage A: Vraaggesprek

Vraaggesprek

Ondervraagden: Frank den Blauwen (programmeur van de huidige image viewer)

Job Scheffers (opdrachtgever),

Aanwezigen: Robert Buitendijk (systeem ontwikkelaar)

Datum: 03-09-2008

Tijd: 10:00 – 10:45

Locatie: ICT-afdeling DSW

Algemene & Non-functionele vragen:

Wat is de structuur van de documenten opgeslagen in InfoImage ? (Frank)

De structuur is als volgt: Een document bestaat uit 0 of meerdere images, die de elk een pagina van het document voorstellen. Ook kan een document 0 of meerdere bijlagen bevatten. In database termen kunnen de bijlagen gespecificeerd worden als BLOB, wat dus wil zeggen dat ieder bestandstype een bijlage zou kunnen zijn van een document.

Functionele vragen:

Wat moet de navigatie control kunnen? (Job)

Deze control moet het mogelijk maken door pagina's en bijlagen van een document te navigeren. Ook moet het de navigatie tussen documenten mogelijk maken indien een rij van documenten als invoer wordt gekregen.

Wat wordt precies verstaan onder navigatie? (Frank)

Dat is het tonen van de vorige of volgende pagina, bijlage of document uit de rij van (uiteindelijke) images. In de huidige image viewer is het ook mogelijk naar de eerste of laatste pagina uit een document te navigeren. Dat mag ook toegevoegd worden.

We moeten dus op 4 manieren kunnen navigeren op 3 verschillende niveaus?

Ja.

Wat zijn de eisen voor het UI? (Job)

Het zelfde als bij de vorige control. We willen een UI dat zoveel mogelijk ruimte overlaat voor het te viewen image. Betreffende de navigatie knoppen is het daarom belangrijk die alleen te tonen als die navigatie mogelijk is.

Bijlage F: RAD converter control

BSc-project IN3405



Requirements Analysis Document

Converter control

Projectgroep:

Abdoelrasheed Jarmohamed

Kishen Gajadhar

Omgeving:

Zorgverzekeraar DSW

Technische Universiteit Delft

Faculteit EWI Technische Informatica

Variant Softwaretechnologie

Commissie:

Job Scheffers (Zorgverzekeraar DSW, begeleider)

B.R. Sodoyer (TU Delft, coördinator / begeleider)

Datum: 25 september 2008

Status: wachten op acceptatie

Versie: 1.0

©September 2008 Zorgverzekeraar DSW

Alle in dit verslag afgebeelde gegevens, waaronder teksten, foto's, illustraties, grafische materiaal, (handels)namen, logo's, waren- en dienstmerken, zijn in eigendom van zorgverzekeraar DSW en worden beschermd door auteursrecht, merkenrecht en/of enig ander intellectueel eigendomsrecht.

Inhoudsopgave

1. Introductie	108
2. Daadwerkelijke opdracht	109
3. Algemene analyse	110
4. Use case analyse	111
5. Requirements	113
5.1 Functionele requirements	113
5.2 Non-functionele requirements	113
Bijlage A: Vraaggesprek	115

1. Introductie

Dit analyse rapport wordt als analyse voor de converter control geschreven. Dit is de derde control in het kader van het bachelorproject van de opleiding technische informatica van de TU Delft. Deze opdracht vindt plaats bij zorgverzekeraar DSW te Schiedam. Bij het lezen van dit rapport is het belangrijk dat analyse van de vorige controls ook gelzen zijn.

Tijdens deze analyse is het ten eerste de bedoeling informatie te verzamelen over het navigatie control. Dit gaat aan de hand van *interviews* en *observatie*. Het doel van het verzamelen van informatie is duidelijk: zonder informatie kan er geen overwogen keus gemaakt worden over het verloop van het project. Door middel van deze analyse kan het probleemdomein duidelijk gedefinieerd en afgebakend worden tot een eenduidig probleemdomein waarna een softwaresysteem zal kunnen worden ontworpen om dit probleem op te lossen. Voor een compleet overzicht van het probleem domein wordt verwezen naar het RAD van de *viewer control*, de eerste control.

Nadat de informatie is verworven wordt de informatie geanalyseerd. Hierbij kan gebruik gemaakt worden van gebruikersanalyses, welke ook te vinden is in het RAD van de viewer control, use-case diagrammen en een lijst van de requirements. Deze analyse is nodig om uit te zoeken welke informatie nodig is voor het realiseren van de control en om uit te zoeken wat wel en niet haalbaar of wenselijk is voor deze control. Hierdoor kan het project gestructureerd worden uitgevoerd en kan er een goed werkend systeem ontworpen worden.

De opbouw van dit rapport is als volgt. In hoofdstuk 2 wordt de daadwerkelijke opdracht herzien gevolgd door hoofdstuk 3, waar een algemene analyse wordt gemaakt van de toe te voegen functionaliteit uit de oude image viewer. Vervolgens wordt in hoofdstuk 4 de use case analyse besproken gevolgd door de lijst van requirements in hoofdstuk 5.

2. Daadwerkelijke opdracht

Het doel van het project is om een usercontrol te ontwikkelen met minimaal de zelfde functionaliteiten als de huidige image viewer. De systemen die ontwikkeld worden voor DSW, worden bijna allemaal ontwikkeld met de programmeertaal C# van het .NET-omgeving. De image viewer control zal dan ook in C# geschreven moeten worden, om het integreren van de control mogelijk en makkelijk te maken.

De image viewer control zal in vierën worden gedeeld. Een van de redenen van het opsplitsen is om het ontwikkelen zo overzichtelijk mogelijk te houden en de functionaliteiten van elkaar te scheiden. Er zullen vier verschillende controls worden gemaakt die elk hun eigen functionaliteiten hebben en toevoegen aan het geheel. Elk control, los van de eerste, is gebaseerd op de vorige, waardoor bij voltooiing van een control een belangrijke functionaliteit toegevoegd wordt.

De *eerste control*, de *viewer control*, zal de eigenlijke image viewer zijn met al zijn basisfuncties. Hierin zal een bitmap kunnen worden bekeken op verschillende manieren. Hierin kan een bitmap vergroot, verkleind, geroteerd en geïnverteerd worden.

Een document kan bestaan uit 0 of meer bitmaps gevolgd door 0 of meer bijlagen. Het kunnen navigeren binnen een document naar een andere bitmap, of bijlage zal de *tweede control*, de *navigatie control*, mogelijk moeten maken.

Als een document of een selectie van documenten zal worden opgevraagd zal dat door Drovi worden gegeven in een map. Het kunnen selecteren van een document in een map zal ook mogelijk gemaakt worden door de *tweede control*.

Documenten kunnen andere dan image bestanden bevatten. Deze kunnen niet worden getoond in de image viewer, dus zal de *derde control*, de *converter control*, zorgen het viewen van niet image bestanden.

De *vierde control*, de *search control*, zal het Drovi-zoekscherm vervangen. Hierin zal één of meerdere documenten kunnen worden opgezocht met de verschillende zoekcriteria.

De derde control zal geaccepteerd worden als een gegeven document volledig getoond kan worden, zij het middels de viewer control of een externe applicatie

3. Algemene analyse

In dit hoofdstuk volgt de algemene analyse voor de converter control. Eerst zal gekeken worden hoe de toe te voegen functionaliteit in het huidige systeem zijn verwerkt. Vervolgens worden die functies gedetailleerd besproken.

De huidige situatie

Binnen het huidige image viewing systeem kan een document niet-image bestanden bevatten. Om deze te kunnen viewen is een *ActiveX* control ontwikkeld, dat gegeven een document, de pagina's en bijlagen ophaald, de extensies bekijkt, en aan de hand daarvan de betreffende pagina of bijlage opent in de image viewer of een externe applicatie.

De toekomstige situatie

In de nieuwe image viewer moet de *converter control* een soortgelijke functie uitvoeren als beschreven in de huidige situatie. Een document gehaald uit InfoImage moet compleet getoond kunnen worden, zij het in de image viewer of in een externe applicatie.

Niet-image bestanden kunnen worden geconverteerd naar image bestanden of worden geopend in een externe applicatie.

Gezien de tijdsdruk, zal geen tijd worden besteed aan het converteren van niet-image bestanden naar image bestanden. Er zal gebruik worden gemaakt van de *ActiveX* control om documenten uit InfoImage te halen en die te openen in betreffende applicaties.

Het ActiveX control

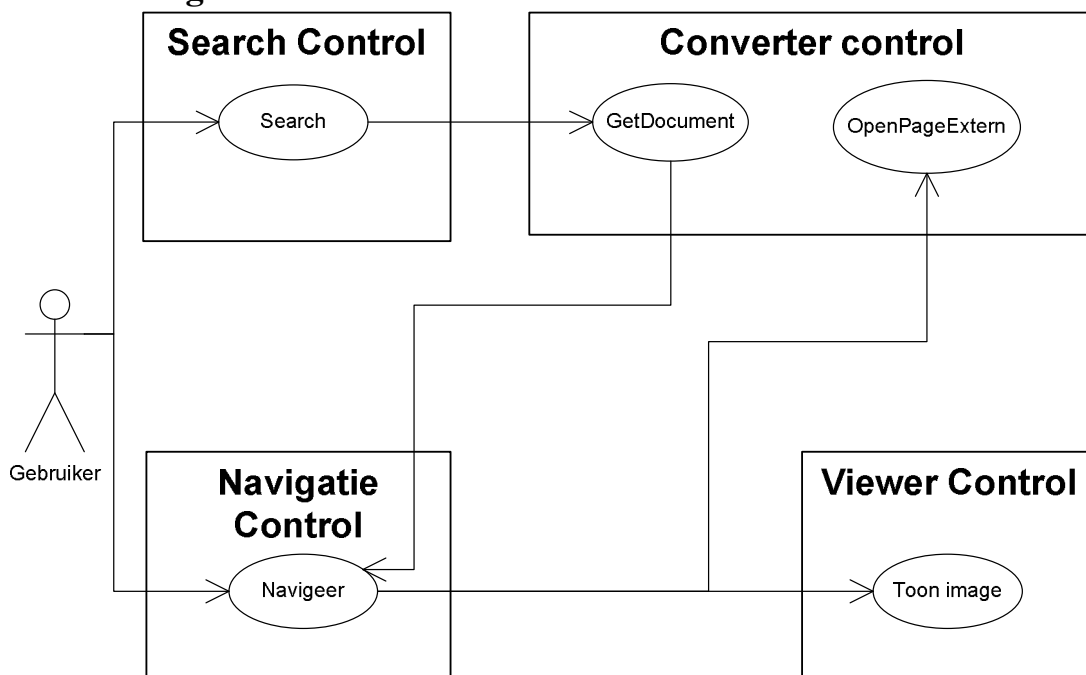
De control werkt als volgt. Voor documenten uit de database gehaald kunnen worden, moet een gebruiker natuurlijk eerst ingelogd zijn. Dit kan gedaan worden door de methode *CALLogin* uit de klasse *CALConnector* aan te roepen. Vervolgens moet uit die zelfde klasse de methode *CALGetColDocs* aangeroepen worden. Die methode geeft een *colOpenDoc* terug. Met behulp van de *colOpenDoc* klasse kan dan met de methode *Add*, die een InfoImageID als invoer krijgt, een *clsOpenDoc* opgehaald worden. De *clsOpenDoc* klasse is de klasse waarin zich een infoImage document bevindt. Uit die klasse kan met behulp van de methode *SavePage* een document pagina uit de database gehaald worden. De methode slaat de pagina ergens op schijf op en geeft dan de path van die locatie terug.

De control bevat een aantal plug-ins om niet-image bestanden toonbaar te maken. Een daarvan is bijvoorbeeld *GhostScript PDF Plug-in*. Elke applicatie waarvan een plug-in beschikbaar is voor de control, kan worden geopend door de control. Bij het openen van een externe applicatie wordt het betreffende niet-image bestand ook meegegeven. Dit gaat met behulp van de methode *LaunchEx* uit de klasse *clsOpenDoc*. Voor een pagina uit het document opgehaald wordt, word gecontroleerd of de pagina wel te openen is met deze control, zij het in de image viewer of extern. Dit wordt gedaan door met de methode *ImportExtension* de extensie per bijlage van het document op te vragen en te controleren.

4. Use case analyse

In dit hoofdstuk wordt een overzicht gegeven van de use cases van de *converter control*. Een use case diagram wordt ook gebruikt. Use case diagrammen worden gebruikt om functionele eisen te achterhalen. Elke use case laat een interactie tussen de gebruiker en het systeem zien. Eerst wordt een use case diagram weergegeven, vervolgt door een beschrijving van de use cases. Deze zijn geformuleerd na *observatie* van de huidige image viewer én het *interviewen* van systeem ontwikkelaars en de opdrachtgever. Het benaderen van eindgebruikers werd in dit geval achterwege gelaten. Zie *bijlage A* voor de interviewvragen.

Use case diagram



Figuur 1: Use case diagram Image Viewer control

Use case beschrijvingen

Hier volgt ons enige use case voor deze control, immers dit is ook de enige functionaliteit die wordt toegevoegd met deze control.

Use Case: GetDocument

Actor: Gebruiker

Beginsituatie: De gebruiker heeft een document gevonden en wil die getoond hebben.

9. Het control haalt het document op uit InfoImage
10. Uit het document worden alle pagina's en bijlagen gehaald en in een door de navigatie control gewenste structuur gezet.
 - a. Voor niet-image bestanden wordt een standaard image bestand geplaatst, aangevend dat het bestand extern geopend moet worden.

Eindsituatie: Het gewenste document is geladen met de eerste pagina getoond in de image viewer.

Use Case: OpenPageExtern

Actor: Gebruiker

Beginsituatie: Een document is geladen en de gebruiker navigeert naar een pagina dat een niet-image bestand voorstelt.

3. Het navigatie control geeft door aan het converter control welk niet-image bestand bekeken wil worden.
4. Via het ActiveX control wordt dat bestand extern geopend.

Eindsituatie: De gewenste pagina of bijlage is extern getoond.

5. Requirements

In dit hoofdstuk wordt een overzicht van de functionele- en non-functionele requirements gegeven. De functionele eisen worden in paragraaf 7.1 ingedeeld volgens het MoSCoW model. De non-functionele eisen zijn de zelfde als voor de *viewer control*, maar voor het gemak worden die in paragraaf 7.2 herhaald.

MoSCoW model

De functionele requirements zullen via het MoSCoW model worden weergegeven. Het MoSCoW model is een veel gebruikte methode voor het verdelen van de requirements ofwel eisen naar prioriteit. De verschillende prioriteiten hierbij zijn achtereenvolgens:

- Must have (Mo):** zijn absoluut *noodzakelijk* om te implementeren
- Should have (S):** *zouden* in principe geïmplementeerd moeten worden
- Could have (Co):** worden alleen geïmplementeerd indien er tijd over is
- Want to have (W):** mooi om te hebben voor opdrachtgever

5.1 Functionele requirements

In deze paragraaf worden de functionele eisen voor de navigatie control opgesomd. Dit is gedaan volgens het MoSCoW model.

Must have

13. Een gegeven document moet uit InfoImage gehaald kunnen worden.
14. Een opgehaalde document moet omgezet kunnen worden in een door de navigatie control gewenste structuur.
15. Een niet-image bestand moet extern geopend kunnen worden.

5.2 Non-functionele requirements

In deze paragraaf volgen de niet functionele requirements. Deze zijn onderverdeeld in *quality*, *platform* en *process* requirements. Zoals eerder gezegd zijn deze dezelfde als voor de *viewer control*.

Quality requirements

- *Reliability*
Onder reliability van een systeem wordt verstaan de tijd dat zo'n systeem stil mag liggen door fouten. Voor de *image viewer control* zou de reliability 99,9% moeten zijn. Met andere woorden 999 uit de 1000 keer zal de control niet falen.
- *Availability*
Dit is de tijd dat een systeem beschikbaar moet zijn. In zijn totaliteit moet de image viewer control 99,9% van de tijd beschikbaar zijn.
- *Performance*
Dit is de kwaliteit van een systeem in zijn geheel. De viewer moet onder elke omstandigheid flikkervrij zijn.
- *Response* *time*
De tijd die de *gebruiker* moet wachten op resultaat van een bewerking of functie, nadat hij/zij de functie heeft laten uitvoeren. Responstijd zou kleiner kunnen zijn dan *een seconde*,

gemeten vanaf het moment dat de gebruiker de functie laat plaatsvinden, tot het moment dat het resultaat beschikbaar is.

- *Reusability*

Dit is het percentage van een systeem dat herbruikbaar zou moeten zijn. Het te bouwen control moet integreerbaar zijn in meerdere systemen, waardoor het 100% reusable moet zijn. Het feit dat het een control is, maakt het 100% reusable.

Platform requirements

- *Computing*

platform

De minimale eisen waarom het systeem gedraaid kan worden. Voor de image viewer control geldt als minimale systeem eisen:

Processor: Intel Pentium 4, 2.0 GHz

Werkgeheugen: 512 MB

Videokaart: geïntegreerd

Beeldscherm: 18" (1280x1024)

Operating system: Windows XP

- *Technology to be used*

De systemen waarin het te bouwen control geïntegreerd zal worden, zullen worden geschreven in C#. Voor de control is het van belang dat het coderen ook in C# gebeurt. Voor het zoomen dient gebruik gemaakt te worden van *bicubic interpolation* of een soortgelijke methode voor het verbeteren van een vergrote image.

Process requirements

- *Development process (methodology)*

Het ontwerp en de analyse worden gedocumenteerd in UML. Het testen wordt in units gedaan.

Bijlage A: Vraaggesprek

Vraaggesprek

Ondervraagden: Frank den Blauwen (programmeur van de huidige image viewer)

Aanwezigen: Job Scheffers (opdrachtgever),

Robert Buitendijk (systeem ontwikkelaar)

Datum: 03-09-2008

Tijd: 10:00 – 10:45

Locatie: ICT-afdeling DSW

Funtionele vragen:

Hoe wordt binnen de huidige image viewer gezorgd dat niet image bestanden geviewed kunnen worden?

Daar heb ik een ActiveX control voor gebouwd. De extensie van zo'n bestand kan achterhaald worden. Aan de hand daarvan kan het ActiveX control de juiste applicatie aanroepen om het bestand daarin te tonen. Echter geldt dit niet voor PDF bestanden. Die worden omgezet naar image bestanden en in de image viewer getoond

Zou gebruik gemaakt kunnen worden van het ActiveX control?

Dat kan zeker, het is een COM object en kan dus gewoon gerefereerd zijn binnen visual studio.

Hoe werkt het control precies?

Het control haalt de documenten uit InfoImage, gegeven de *InfoImageId* van het betreffende document. Als een pagina getoond dient te worden, wordt gekeken naar de extensie daarvan, en geopend in de betreffende applicatie. Dit omvat dus ook de image viewer.

Bijlage G: SDD viewer control

BSc-project IN3405



Design Document Viewer control

Projectgroep:

Abdoelrasheed Jarmohamed
Kishen Gajadhar

Commissie:

Job Scheffers (Zorgverzekeraar DSW, begeleider)
B.R. Sodoyer (TU Delft, coördinator / begeleider)

Omgeving:

Zorgverzekeraar DSW
Technische Universiteit Delft
Faculteit EWI Technische Informatica
Variant Softwaretechnologie

Datum: 21 augustus 2008

Status: wachten op acceptatie

Versie: 1.0

©Augustus 2008 Zorgverzekeraar DSW

Alle in dit verslag afgebeelde gegevens, waaronder teksten, foto's, illustraties, grafische materiaal, (handels)namen, logo's, waren- en dienstmerken, zijn in eigendom van zorgverzekeraar DSW en worden beschermd door auteursrecht, merkenrecht en/of enig ander intellectueel eigendomsrecht.

Voorwoord

Dit is het design document voor de image viewer control die wordt ontwikkeld voor de Zorgverzekeraar DSW. Na het requirement analysis document is dit het tweede document voor het bachelorproject IN3405 van de opleiding technische informatica van de Technische Universiteit Delft. Hierin wordt elke stap van de ontwerpfase van de image viewer control uitgewerkt en beschreven.

Elke beslissing die is gemaakt wat betreft het design is hierin beschreven of geïllustreerd met diagrammen. Hier staan dus ook de klassendiagrammen, volgordediagrammen en een illustratie van het prototype van de user interface. Het implementeren van image viewer control zal aan de hand van dit document worden uitgevoerd.

Tijdens de ontwerpfase van ons project waren de begeleiders van de TU en DSW (J. Scheffers en B. Sodoyer) op vakantie. Deze fase hebben we dus zelfstandig doorlopen. Eventuele problemen konden we voorleggen aan één van medewerkers van DSW die als vervangende begeleider diende. Grotere problemen die we zouden kunnen tegenkomen konden we voorleggen aan de begeleiders bij terugkomst van vakantie. Gelukkig zijn we niet veel problemen tegengekomen en heeft het project geen vertraging opgelopen door de vakantie.

Op de voorkant van dit document ziet u een ander logo van DSW dan het logo die u hebt gezien op de analyse document. Ook de naam is anders. Sinds 1 augustus 2008 heeft DSW een ander logo en heet Zorgverzekeraar DSW nu DSW Zorgverzekeraar.

De kern van dit rapport is hoofdstuk 3, 4 en 6. In hoofdstuk 3 'Overzicht Ontwerp' wordt globaal uitgelegd hoe het ontwerp van de image viewer control in elkaar zit. In hoofdstuk 4 'Keuze in het Ontwerp' wordt nog een beetje dieper ingegaan in het ontwerp en wordt uitgelegd waarom bepaalde keuzes zijn gemaakt. De klassendiagram en de volgordediagrammen staan in hoofdstuk 6. Naast deze diagrammen worden een aantal details nader uitgelegd.

21 augustus 2008, Schiedam

Inhoudsopgave

Voorwoord	117
Inhoudsopgave	118
1 Inleiding.....	119
2 Doel en Prioriteit.....	120
3 Overzicht ontwerp	121
4 Keuzes in het ontwerp.....	122
5 User Interface.....	124
6 Details ontwerp	127
6.1 Klassendiagram	127
6.2 Volgordediagrammen	128
7 Planning.....	133

1 Inleiding

De analysefase hebben we afgerond. Tijdens deze fase hebben we de benodigde informatie verzameld die nodig is om de image viewer control te ontwikkelen. In de ontwerpfase zal aan de hand van deze informatie het ontwerp worden gemaakt. En aan de hand van het ontwerp zal dan geïmplementeerd worden. De ontwerpfase zal dus een belangrijke fase zijn, hierin zullen alle beslissingen worden genomen over de te ontwikkelen software. Verkeerde beslissingen en een verkeerd ontwerp zullen leiden naar een verkeerde of slechte implementatie.

In de ontwerpfase zal er dus worden voortgebouwd op de analysefase. De keuzes en eisen van de werkgever zullen worden vertaald naar een ontwerp. In het geval van een image viewer control zal er geen groot en ingewikkeld ontwerp nodig zijn. Dat wat het ontwikkelen van zo een control moeilijk maakt, zijn toch meer de user interface en de algoritmes die moeten worden gebruikt voor de verschillende requirements.

Hoe het ontwerp in elkaar zit zal worden beschreven, maar kan ook worden gezien door de verschillende diagrammen die zijn gemaakt. Deze diagrammen zijn onder andere een klassendiagram en volgorde diagrammen. Een ander onderdeel van de ontwerpfase is het maken van een prototype van de user interface. De user interface zal een belangrijke rol spelen in dit project. Omdat het hier gaat om het veranderen van de view naar de bitmaps voor de eindgebruikers.

In hoofdstuk 2 wordt kort beschreven welk deel van de opdracht wordt behandeld in dit design document. De requirements voor dit deel worden herhaald. Ook wordt aangegeven waar de prioriteiten liggen van de opdrachtgever. In hoofdstuk 3 'Overzicht Ontwerp' wordt het ontwerp globaal beschreven. Hoe we tot het ontwerp zijn gekomen en waarom we bepaalde keuze hebben gemaakt staat beschreven in hoofdstuk 4 'Keuzes in het Ontwerp'. De user interface wordt uitgebreid behandeld in hoofdstuk 5. Alle mogelijkheden van de image viewer control worden hier beschreven. In hoofdstuk 6 staan de klassen- en volgorde diagrammen. En als laatste in hoofdstuk 7 staat de planning weergegeven in een tabel.

2 Doel en Prioriteit

In dit hoofdstuk wordt kort beschreven welk deel van de opdracht wordt behandeld in dit design document. De requirements voor dit deel worden herhaald. Ook wordt aangegeven waar de prioriteiten liggen van de opdrachtgever.

In het analyse document is aangegeven dat de opdracht is opgedeeld in vier verschillende controls. Dit design document beschrijft alleen het ontwerp van de eerste control, de eigenlijke image viewer. In deze control zal een bitmap op verschillende manieren worden bekeken. De requirements in deze control zijn: het tonen van een gegeven bitmap, het in- en uitzoomen met de muisknop, het in- en uitzoomen van een geselecteerde rectangle, het roteren, het inverteren, het scrollen met de pijltjes toetsen en het bitmap bekijken in de fullmode, heightmode, widthmode en nonemode. Dit allemaal moet kunnen met het behoud van de leesbaarheid van de bitmaps. De requirements die van minder belang zijn in deze control zijn het pannen van een bitmap en het kunnen verbergen en verplaatsen van de toolbar.

Van de hele opdracht heeft de eerste control de hoogste prioriteit. Hierin bevinden zich de mogelijkheden om de view van de bitmaps te veranderen, dat waar het project grotendeels om draait. Wat erg belangrijk is voor de werkgever is dat het veranderen van de view flikkervrij gebeurt. Het ontwerp en de algoritmes moeten zo worden gekozen dat ze efficiënt gebruik maken van beschikbare resources en hierdoor voorkomen wordt dat het beeldscherm zal gaan flikkeren.

Ook de leesbaarheid van de bitmaps zijn belangrijk. Wanneer een bitmap kleiner wordt tijdens het inzoomen, moet er worden gezorgd dat het leesbaar blijft. Bij het inzoomen van bitmaps met andere applicaties, veranderen de letters van vorm doordat er een aantal pixels van de letters verdwijnen. In deze image viewer control moet dit worden voorkomen met behulp van interpolatie. Van interpolatie zal ook gebruik worden gemaakt voor het uitzoomen. Als een bitmap helemaal is uitgezoomd, kunnen de letters heel blokachtig en onduidelijk zijn. Dit moet ook worden voorkomen.

Als laatste is de herbruikbaarheid van de control erg belangrijk. Het moet gemakkelijk geïntegreerd kunnen worden in de verschillende applicaties van DSW. Tijdens het ontwerp hoeft daar niet al te veel rekening mee worden gehouden. In Visual Studio is er bij het aanmaken van elk nieuw project de keuze van verschillende templates. Voor het maken van een losstaande applicatie hoeft alleen de applicatie-template worden gekozen. Een template voor een integreerbare control is ook aanwezig. Het ontwerpen van de GUI met behulp van Visual Studio zal dus gelijk gebeuren op een ontwerpscherm van een control. De benodigde specifieke code voor een control zal dan worden gegenereerd door Visual Studio.

3 Overzicht ontwerp

In dit hoofdstuk wordt globaal beschreven hoe het ontwerp van de image viewer control in elkaar zit. Het geeft slechts een overzicht van belangrijke punten uit het gedetailleerde ontwerp en gemaakte keuzes. Voor een gedetailleerde behandeling wordt hierbij daarom verwezen naar hoofdstuk 4 en 6.

De opbouw van de user interface is erg belangrijk bij het ontwerpen en implementeren van een image viewer control. Als de view van een bitmap wordt veranderd, moet een deel van de user interface veranderen. Ofwel een aantal van de componenten, waaruit de user interface is opgebouwd, moeten zich aanpassen om de gevraagde view weer te geven. Het aanpassen van de componenten zal dus moeten gebeuren met behulp van een aantal klassen die moeten worden ontworpen en geschreven. Belangrijk is dus ook de keuze van de componenten die worden gebruikt. Want elk component heeft weer zijn specifieke eigenschappen en mogelijkheden.

De opdracht is om een control te maken en niet een applicatie. Het is de bedoeling dat deze control in meerdere applicaties binnen DSW kan worden ingebouwd/toegevoegd. In Visual Studio is er de mogelijkheid om te kiezen uit verschillende templates. Voor het ontwikkelen van een control is er ook een template, namelijk de Windows Control Library template. Een standaard usercontrol klasse wordt dan gegenereerd door Visual Studio. Ook wordt er een form toegevoegd in het designer-scherm van Visual Studio. Maar in plaats van een normale windows-form voor een normale applicatie, wordt er voor een control een usercontrol-form toegevoegd. Dit is dan de container voor alle componenten van de user interface.

In de usercontrol-form komt bovenaan een toolbar met knoppen. Met deze knoppen kunnen de meeste functies van de Image viewer control worden uitgevoerd. Onder de toolbar wordt een ZoomablePictureBox geplaatst. De ZoomablePictureBox is een zelfgemaakte subklasse van Panel waarin een image kan worden weergegeven. De bitmap wordt gezet in een Image klasse en dan weergegeven in de ZoomablePictureBox. De ZoomablePictureBox is dan verantwoordelijk voor het vergroten en verkleinen van een bitmap.

Er is nog een derde klasse, genaamd Functions. Hierin worden alle functies van de Image viewer control met behulp van ZoomablePictureBox geïmplementeerd. Dit zijn roteren, zoomen, inverteren, scrollen, pannen en de 4 fitmodes.

In de usercontrol klasse worden de mouse- en buttonevents geschreven. De klassen ZoomablePictureBox en Functions worden eerst geïnstantieerd en de methodes kunnen dan gebruikt worden in de events.

4 Keuzes in het ontwerp

Dit hoofdstuk behandelt een belangrijke keuze die tijdens het ontwerpproces gemaakt is. Het zoomen kan op verschillende manieren gebeuren. Hier wordt de gemaakte keuze nader uitgelegd.

Het ontwerp van de image viewer control, die moet worden ontwikkeld, zal niet groot en ingewikkeld zijn. Dat heeft natuurlijk een aantal voordelen. Als er wijzigingen nodig zijn in het ontwerp is dat gemakkelijk aan te passen bij een klein en simpel ontwerp. In dit geval zit de moeilijkheid niet in het ontwerp, maar in het vinden van de juiste algoritmes en het implementeren van de functies.

Het zoomen van een bitmap is de belangrijkste functies van de image viewer control. Er zijn namelijk ook andere functies die gebruik moeten of kunnen maken van het zoomen. Als een bitmap niet in zijn originele formaat wordt weergegeven, moet er altijd gebruik worden gemaakt van het zoomen. Zoals in de fitmodes (behalve de none-fit) en het zoomen van een rectangle.

Met het .NET Framework zijn er een aantal manieren om een bitmap te zoomen. Een belangrijke keuze die moet worden gemaakt voor het ontwerp is op welke manier het zoomen zal gebeuren. Deze keuze zal zelfs invloed hebben op een klein ontwerp. Bij het maken van een keuze moeten we rekening houden dat het ontwerp voldoet aan de eisen, zoals het zoomen met interpolation en het zoomen zonder enige flickering.

Een aantal methodes die nodig zijn voor de andere functies van de image viewer control, zijn ook beschikbaar in het .NET Framework. Hiervan kan ook handig gebruik van worden gemaakt.

Hier een korte uitleg over twee verschillende manieren van zoomen en eventuele andere functionaliteiten.

Zoomen met behulp van een PictureBox

In het .Net framework kan er gebruik worden gemaakt van verschillende containerklassen. De klasse PictureBox is zo een klasse. Dat is een control die een bitmap kan weergeven in de user interface.

PictureBox heeft een member (attribuut in Java) genaamd SizeMode. Door een mode te kiezen kan worden gekozen hoe de bitmap wordt weergegeven in de PictureBox. Eén van de modes is Zoom. In deze mode geeft de PictureBox de bitmap in zijn geheel weer zonder de verhoudingen te veranderen. Dus als de PictureBox groot of klein is, zal de bitmap ook groot of klein zijn. Voor het zoomen kan hier dus gebruik van gemaakt worden. Het inzoomen gebeurt dan door gewoon de PictureBox te verkleinen en het uitzoomen door de PictureBox te vergroten.

Met andere members en methodes van deze klasse kunnen andere weergaves van de bitmap worden gerealiseerd. Er zijn bijvoorbeeld methodes waarmee een bitmap kan worden geroteerd. Het scrollen is mogelijk door de PictureBox te plaatsen in een Panel. De Panel zorgt dan voor de scrollbar en als de PictureBox groter wordt dan de Panel zelf.

Zoomen met behulp van een zelfgeschreven containerklasse

Een ander mogelijkheid om te zoomen is om zelf een containerklasse te ontwerpen en creëren. Dit kan dan een derived klasse (subklasse) zijn van Panel, PictureBox, Control, ScrollableControl of een andere klasse waarvan de functies handig kunnen zijn om over te erven.

De methodes voor het tonen en zoomen van een bitmap, moeten zelf worden geschreven voor de nieuwe containerklasse.

Het roteren is mogelijk door de bitmap in een Image klasse te zetten. De Image klasse heeft methodes waarmee dat gedaan kan worden. Het scrollen kan gebeuren door gebruik te maken van de base klasse (upperklasse).

Keuze

Het nadeel van het zoomen met behulp van een PictureBox is dat de PictureBox het zoomen controleert. De PictureBox tekent de bitmap op de Panel. Zelf heb je daar weinig controle over. Inzoomen met interpolation ondersteunt de PictureBox niet. En omdat je het zoomen niet kan beïnvloeden van binnenuit, kan daar weinig tegen worden gedaan.

Ook een bitmap tekenen op een Panel met behulp van een dubbele buffer, om het flikkeren tegen te gaan, is niet mogelijk. Dit ook omdat de PictureBox controle heeft over het tekenen van de bitmap.

En of de PictureBox groot genoeg kan worden voor het inzoomen tot 1800% is ook maar de vraag.

Omdat tijdens zoomen met een PictureBox geen gebruik kan worden gemaakt van interpolation en een dubbele buffer, wordt er een containerklasse ontworpen waarbij dat wel allemaal kan.

5 User Interface

De user interface van de eerste control zal er heel simpel uitzien. Het bestaat alleen uit een knoppenbalk met daaronder de ZoomablePictureBox waar de bitmap in zal worden getoond.

De user interface van de image control viewer zal uit één hoofd-, een informatie- en een helpscherm bestaan. Alle functionaliteiten zullen vanuit het hoofdscherm beschikbaar zijn. De meeste kunnen worden uitgevoerd door op één van de buttons in de user interface te drukken. Door op de linker- of rechtermuisknop te klikken in de bitmap worden er ook een aantal functionaliteiten geactiveerd.

In het informatiescherm staat bepaalde informatie over de bitmap die wordt weergegeven. Het komt te voorschijn als de informatieknop wordt ingedrukt.

In het helpscherm staat een handleiding van de Image viewer control. Het helpscherm komt te voorschijn als de helpknop wordt ingedrukt.

Hieronder in figuur 1 wordt het hoofdscherm weergegeven. Daarna wordt elke event van deze control kort uitgelegd.



Figuur 1: Hoofdscherm

Mogelijke uit te voeren acties met de knoppen



Door op deze knop te drukken wordt de width-fitmode ingeschakeld. De bitmap zal zo worden gezoomd dat de hele breedte van de bitmap precies de hele breedte van de control

opvult. *Functional requirement 6*



Door op deze knop te drukken wordt de height-fitmode ingeschakeld. De bitmap zal zo worden gezoomd dat de hele lengte van de bitmap precies de hele lengte van de control opvult. *Functional requirement 6*



Door op deze knop te drukken wordt de full-fitmode ingeschakeld. De bitmap zal zo worden gezoomd dat het gehele bitmap te zien. *Functional requirement 6*



Door op deze knop te drukken wordt de none-fitmode ingeschakeld. De bitmap zal helemaal niet worden gezoomd en zal in zijn originele formaat te zien zijn. *Functional requirement 6*



Door op deze knop te drukken wordt de bitmap met 90 graden naar links gedraaid. *Functional requirement 3*



Door op deze knop te drukken wordt de bitmap met 90 graden naar rechts gedraaid. *Functional requirement 3*



Door op deze knop te drukken worden de kleuren van de bitmap geïnverteerd. *Functional requirement 4*



Door op deze knop te drukken wordt de zoom-selection mode ingeschakeld. *Functional requirement 2*



Door op deze knop te drukken wordt er bepaalde informatie gegeven over de bitmap.



Door op deze knop te drukken wordt de handleiding geopend.

Mogelijke uit te voeren acties met de muis

Door de rechtermuisknop ingedrukt te houden in de bitmap en de muis omhoog te bewegen, zal de bitmap worden ingezoomd. *Functional requirement 1*

Door de rechtermuisknop ingedrukt te houden in de bitmap en de muis omlaag te bewegen, zal de bitmap worden uitgezoomd. *Functional requirement 1*

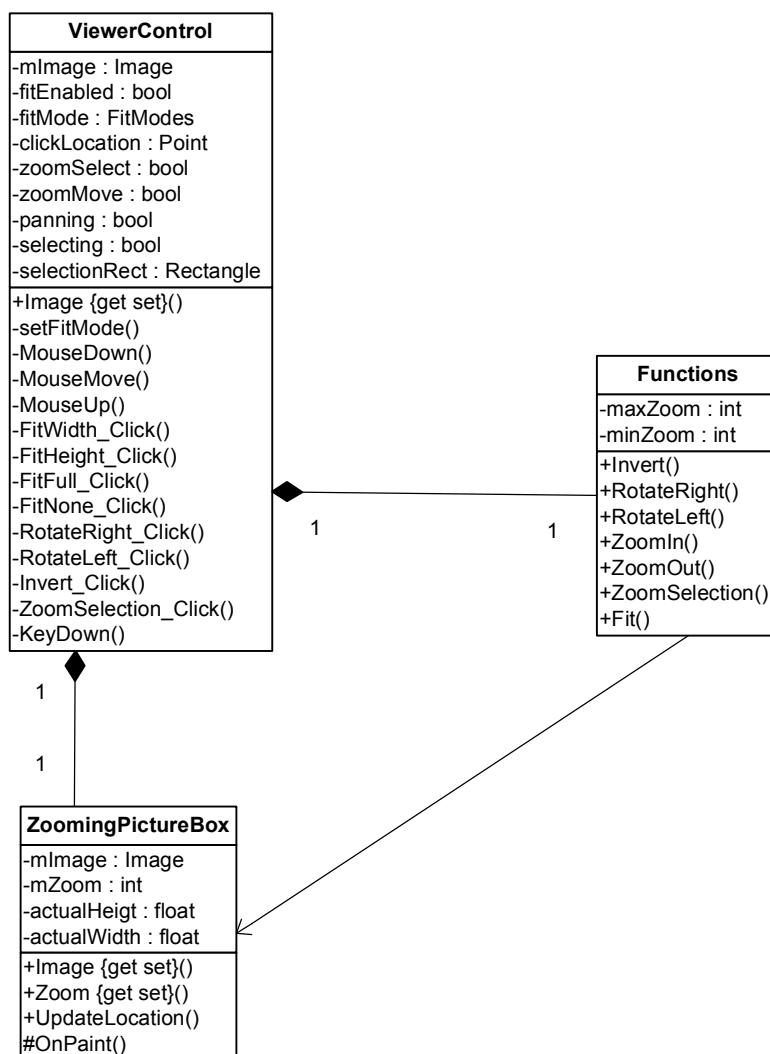
Door de linkermuisknop ingedrukt te houden in de bitmap en de muis te bewegen, kan de bitmap worden gepand. *Functional requirement 7*

Als de zoom-selection is ingeschakeld, kan door de linkermuisknop ingedrukt te houden een rectangle getekend worden. Bij het loslaten van de linkermuisknop wordt de geselecteerde rectangle in de bitmap ingezoomd weergegeven. *Functional requirement 1*

6 Details ontwerp

In dit hoofdstuk zijn de details van het ontwerp van de Image Viewer Control weergegeven. Dit is gedaan met behulp van een aantal diagrammen. In paragraaf 6.1 staat het klassendiagram en wordt het kort behandeld. In paragraaf 6.2 komen de volgordediagrammen aan bod van de verschillende functionaliteiten.

6.1 Klassendiagram



Figuur 2: Klassendiagram

ViewerControl Dit is het hoofdscherm van de Image viewer control. Dit is de user interface met de button- en mouse-events die kunnen worden aangeroepen door de eindgebruikers.

ZoomingPictureBox Deze klasse zorgt voor de gevraagde weergave van de eindgebruiker van een bitmap.

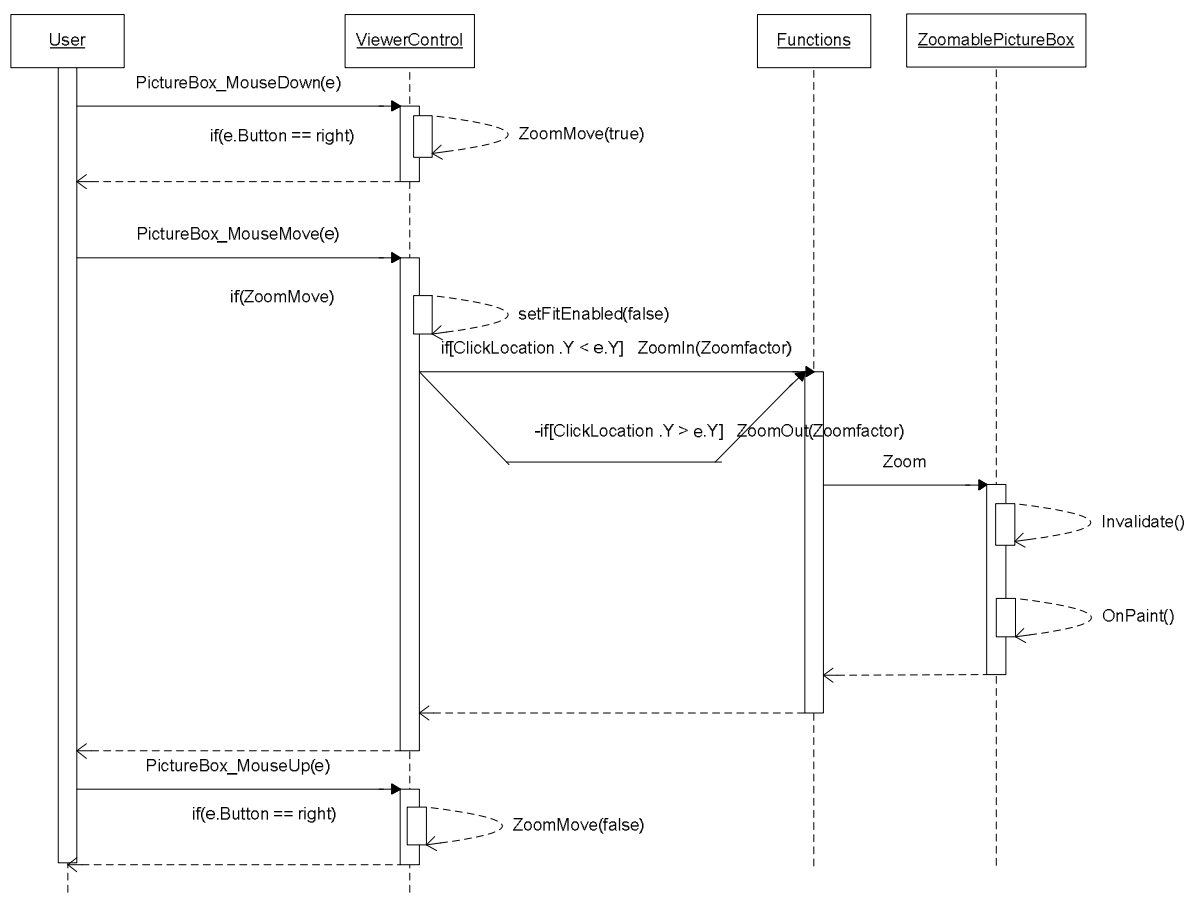
Functions In deze klasse zitten functies die ViewerControl kan gebruiken om de bitmap weer te geven zoals de eindgebruiker dat wilt.

De ViewerControl wordt opgebouwd uit een toolbar en daaronder de ZoomingPictureBox. In de klasse Functions staan de methodes om de weergave van de Image in de ZoomingPictureBox te veranderen. Bij elke keer als wordt gevraagd de weergave van de Image te veranderen zal de ZoomingPictureBox zichzelf opnieuw tekenen met de nieuwe weergave. De methode die dan wordt opgeroepen, berekent de juiste zoomfactor waarmee de Image moet worden getekend. Ook als er wordt gevraagd een andere deel van de Image weer te geven, wordt berekend vanaf welke positie van de Image dat moet worden gedaan. De nieuwe zoomfactor en positie worden dan aan de ZoomingPictureBox doorgegeven. Met behulp van deze informatie wordt in de OnPaint methode van de ZoomingPictureBox de Image opnieuw getekend.

6.2 Volgordediagrammen

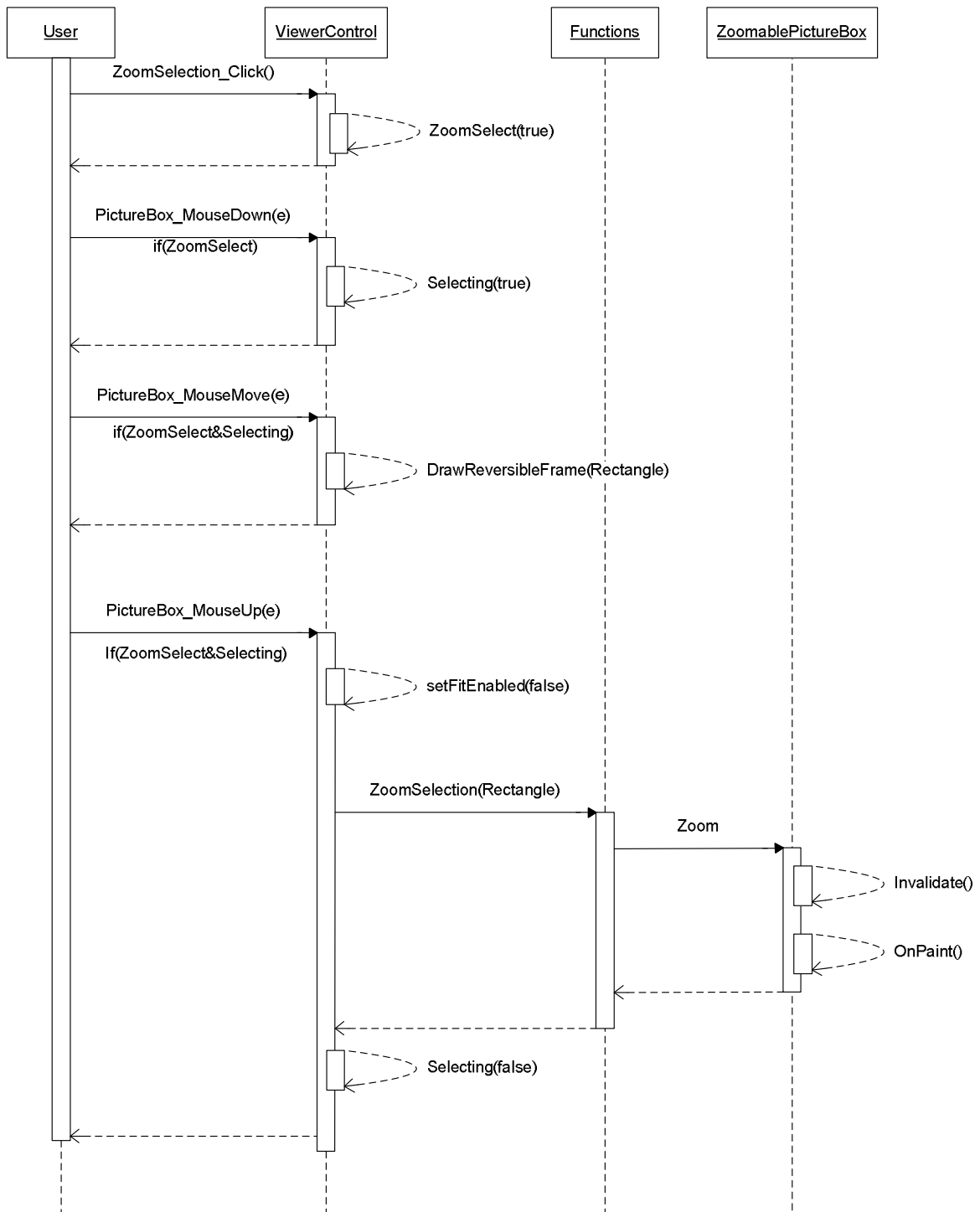
In deze paragraaf staan de volgordediagrammen van de requirements. Dit zijn zoomen, zoomen met een rectangle, roteren, inverteren, de fitmode en het pannen.

Zoomen



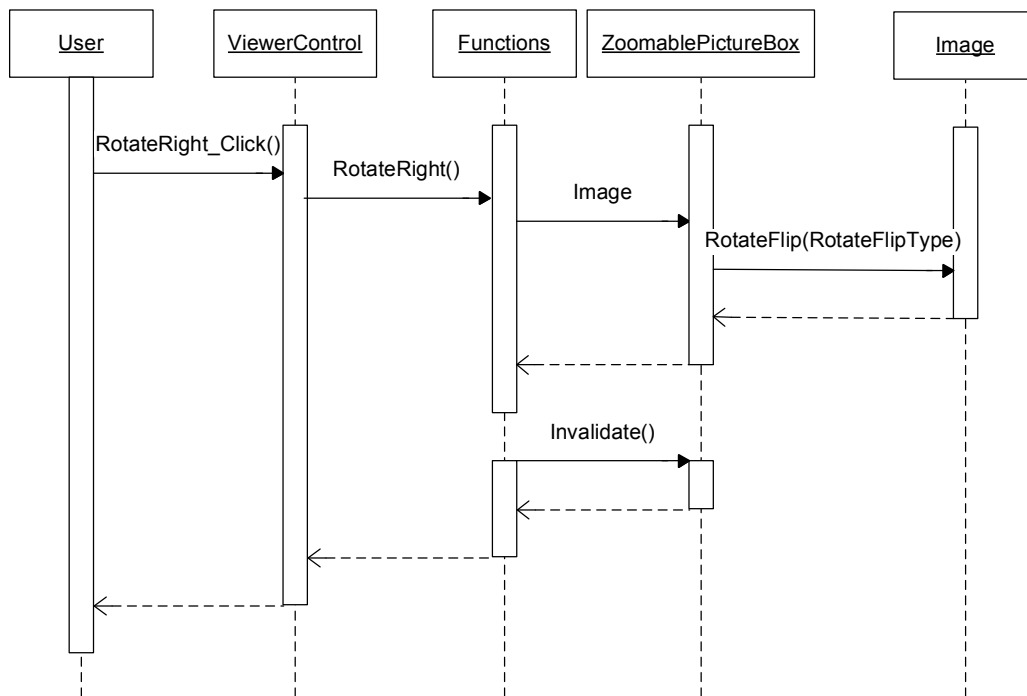
Figuur 3: Volgordediagram Zoomen

Zoomen met een rectangle



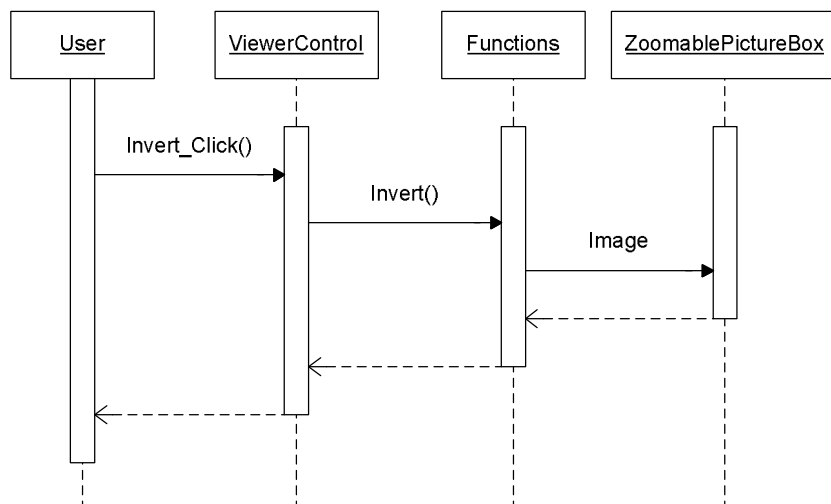
Figuur 3: Volgordediagram Zoomen met een rectangle

Roteren



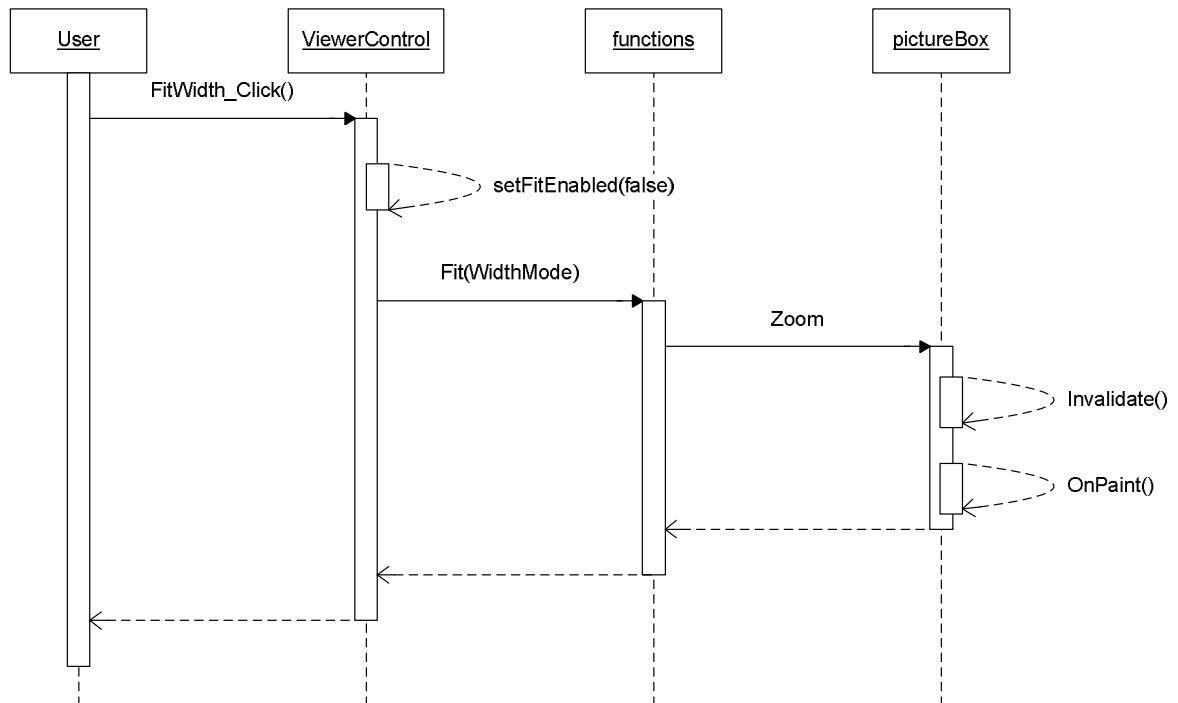
Figuur 3: Volgordediagram Roteren

Inverteren



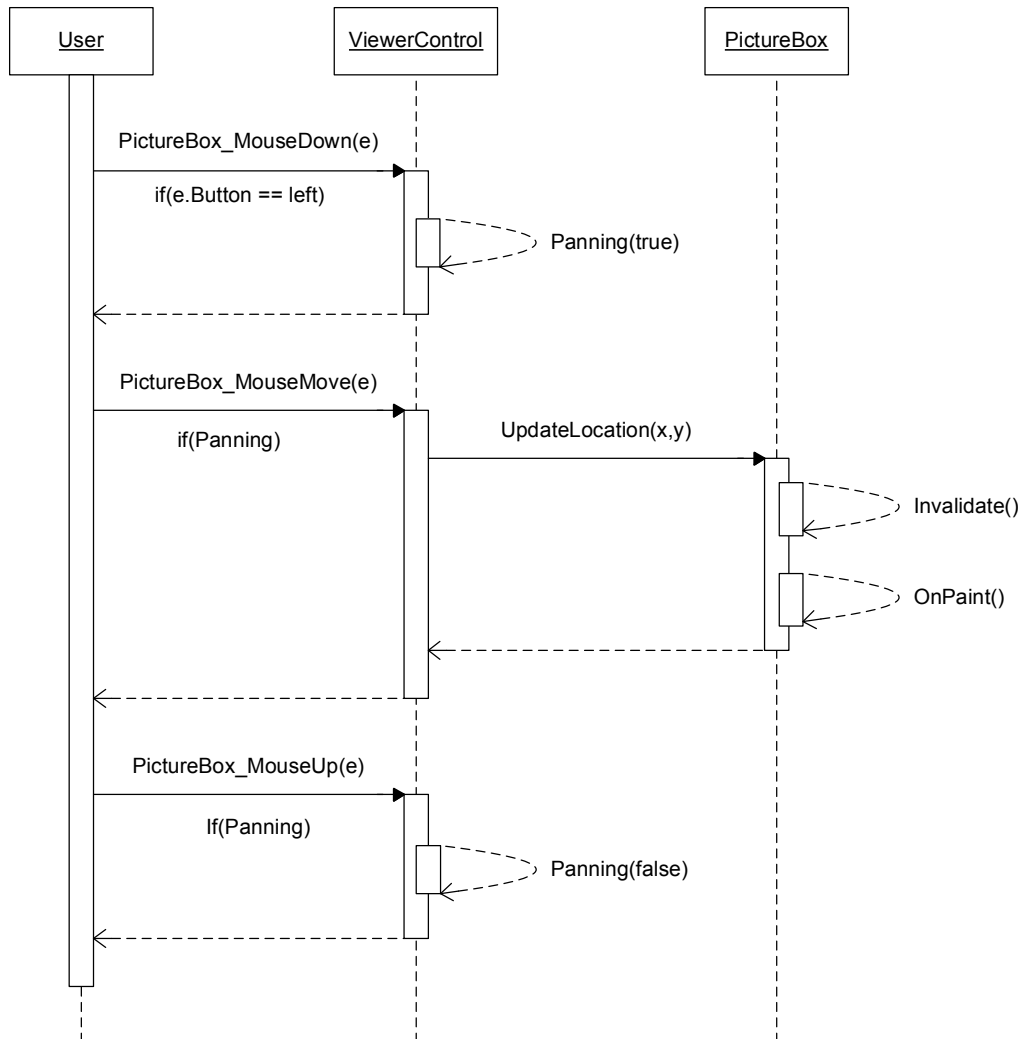
Figuur 3: Volgordediagram Inverteren

Fitmode:Width



Figuur 3: Volgordediagram Fitmode

Pannen



Figuur 3: Volgordediagram Pannen

7 Planning

Dit hoofdstuk geeft de planning van de ontwerpfase. Binnen paragraaf '8.1 Begrote en daadwerkelijke planning' is per taak aangegeven door wie of welke personen deze is uitgevoerd. Voor iedere taak is de vooraf begrote tijdsbesteding gegeven, met daarnaast de daadwerkelijk bestede tijd.

	<i>Kishen</i>	<i>Rashied</i>	<i>Begroot(uur)</i>	<i>Besteed(uur)</i>	<i>Datum indicatie</i>
Kennis verwerven (incl zoomen enz)	x	x	40	50	*
Planning: <i>ontwerp</i>	x		1	1	28-07-2008
Ontwerp					
UI ontwerp (prototype)	x		4	2	04-08-2008
Klassendiagrammen	x		8	4	04-08-2008
Sequence diagrammen		x	8	12	08-08-2008
Kritische reflectie en terugkoppeling					
Uitvoeren reflectie UI prototype	x	x	6	4	11-08-2008
Uitvoeren reflectie ontwerp	x	x	6	4	11-08-2008
Ontwerp					
Final UI	x		8	3	12-08-2007
Final ontwerp		x	16	10	12-08-2007
Design Document					
Redactie	x		8	16	*
<i>Introductie, literatuurlijst, bijlagen</i>					
Klassendiagrammen	x		16	4	20-08-2008
Sequence diagrammen		x	16	12	21-08-2008
UI	x		8	1	20-08-2008
Totaal			105	123	

Deadline: 'Design Document': 11-08-2008 - 18:00

Legenda

- x : 'betreffende persoon voert daadwerkelijk de taak uit en neemt volledig deel'
 grijs vakje: 'betreffende persoon voert taken uit maar heeft niet de hoofdverantwoordelijkheid'
 * : 'deadline voor betreffende taak staat niet vast op een enkele datum'

Bijlage H: SDD navigatie control

BSc-project IN3405



Design Document Navigatie control

Projectgroep:

Abdoelrasheed Jarmohamed
Kishen Gajadhar

Commissie:

Job Scheffers (Zorgverzekeraar DSW, begeleider)
B.R. Sodoyer (TU Delft, coördinator / begeleider)

Omgeving:

Zorgverzekeraar DSW
Technische Universiteit Delft
Faculteit EWI Technische Informatica
Variant Softwaretechnologie

Datum: 19 september 2008

Status: wachten op acceptatie

Versie: 1.0

©September 2008 Zorgverzekeraar DSW

Alle in dit verslag afgebeelde gegevens, waaronder teksten, foto's, illustraties, grafische materiaal, (handels)namen, logo's, waren- en dienstmerken, zijn in eigendom van zorgverzekeraar DSW en worden beschermd door auteursrecht, merkenrecht en/of enig ander intellectueel eigendomsrecht.

Inhoudsopgave

1. Inleiding	136
2. Overzicht ontwerp	137
3. User Interface	138
4. Details ontwerp	140

1. Inleiding

De analyse van de *navigatie control* hebben we afgerond. Tijdens die analyse hebben we de benodigde informatie verzameld die nodig is om de navigatie control te ontwikkelen. In de ontwerpfase zal aan de hand van deze informatie het ontwerp worden gemaakt. En aan de hand van het ontwerp zal dan geïmplementeerd worden. De ontwerpfase zal dus een belangrijke fase zijn, hierin zullen alle beslissingen worden genomen over de te ontwikkelen software. Verkeerde beslissingen en een verkeerd ontwerp zullen leiden naar een verkeerde of slechte implementatie.

In de ontwerpfase zal er dus worden voortgebouwd op de analysefase. De keuzes en eisen van de werkgever zullen worden vertaald naar een ontwerp. In het geval van een navigatie control zal er geen groot en ingewikkeld ontwerp nodig zijn. Dat wat het ontwikkelen van zo een control moeilijk maakt, zijn toch meer de algoritmes die moeten worden gebruikt voor de verschillende navigatie requirements.

In hoofdstuk 2 wordt een overzicht gegeven van het ontwerp, gevolgd door een gedetailleerde beschrijving van het user interface en het systeem respectievelijk in hoofdstuk 3 en 4.

2. Overzicht ontwerp

In dit hoofdstuk wordt globaal beschreven hoe het ontwerp van de navigatie control in elkaar zit.

User Interface

Het user interface van het navigatie control is precies dezelfde als dat van de viewer control met toevoeging van de navigatie knoppen. Immers de navigatie control is alleen een extensie van de viewer control. Zoals in het RAD van de navigatie control is beschreven zijn er drie navigatie niveaus met elk vier navigatie mogelijkheden. Dit wil zeggen dat een totaal van twaalf knoppen toegevoegd dienen te worden. Een klein lichtpuntje hierbij is dat de knoppen alleen getoond moeten worden als de betreffende navigatie mogelijk en zinvol is. De toe te voegen knoppen zullen in een *toolbar* geplaatst worden. Dit vanwege de *docking* mogelijkheden van de toolbar. Samen met de viewer control zal de toolbar dan toegevoegd worden aan een control welke de navigatie control moet voorstellen.

Systeem

Deze control heeft als hoofddoel het mogelijk maken van de navigatie tussen en binnen documenten. Om dit te kunnen realiseren dient eerst een bepaalde structuur te worden gehandhaafd. Dit zal gedaan worden met de klassen *MyDocument* en *DocumentAttachment*. De *MyDocument* klasse stelt precies een document, als beschreven in het RAD van de navigatie control, voor, met de uitzondering dat de pagina's van een document worden gegooid in een instantie van de *DocumentAttachment* klasse. Een bijlage wordt, zoals de naam ook aangeeft, voorgesteld door de *DocumentAttachment* klasse. In deze control wordt geen rekening gehouden met andere dan image bestanden. Daarom bevat elke *DocumentAttachment* één of meerdere images.

Om de afwijking met het originele structuur aan te geven: Een *Document* bestaat uit één of meerdere *Attachments* en een *Attachment* uit één of meerdere *Images*.

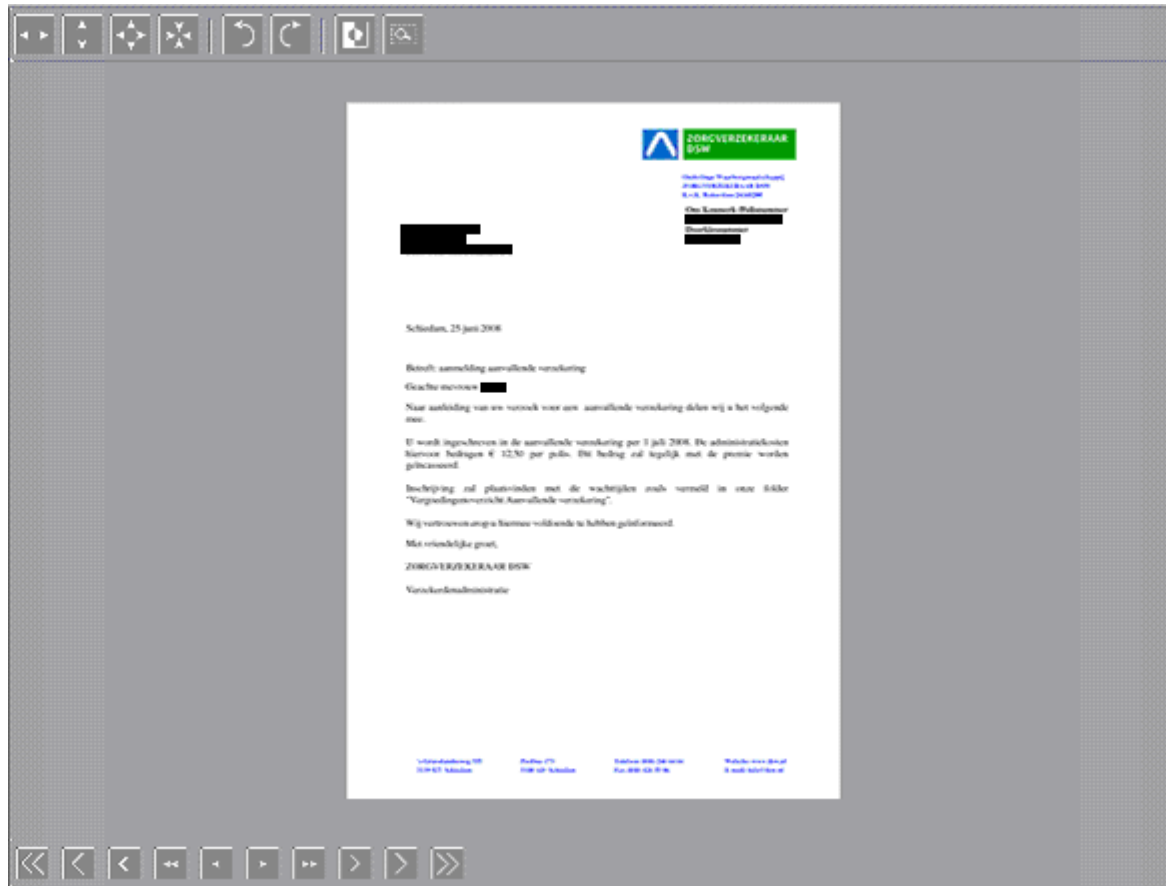
Om de navigatie te handhaven hoeft de control alleen bij te houden welke pagina van welke attachment uit welk document op dit moment getoond is. Aan de hand van die waarden kan gemakkelijk een volgende of vorige pagina, attachment of document getoond worden.

De navigatie control krijgt een rij van documenten als invoer. Bij het ontvangen van zo'n invoer wordt gecheckt welke navigatie type mogelijk is. Daarmee kunnen dan de betreffende knoppen getoond of juist niet getoond worden.

3. User Interface

De user interface van de navigatie control zal er heel simpel uitzien. Het bestaat alleen uit een knoppenbalk met daarboven de viewer control waar de image in zal worden getoond.

Alle functionaliteit van deze control is bereikbaar via de knoppen aan de onderkant van het venster.



Figuur 1: Hoofdscherm

Mogelijke uit te voeren acties met de knoppen



Door deze knoppen te drukken kan worden genavigeerd op pagina niveau. *Functional requirement 1*
De betekenis van de knoppen van links naar rechts: *Eerste pagina, Vorige pagina, Volgende Pagina* en *Laatste pagina*



Door deze knoppen te drukken kan worden genavigeerd op bijlage niveau. *Functional requirement 2*
De betekenis van de knoppen van links naar rechts: *Vorige bijlage* en *Volgende bijlage*



Door deze knoppen te drukken kan worden genavigeerd op document niveau.

Functional Requirement 3

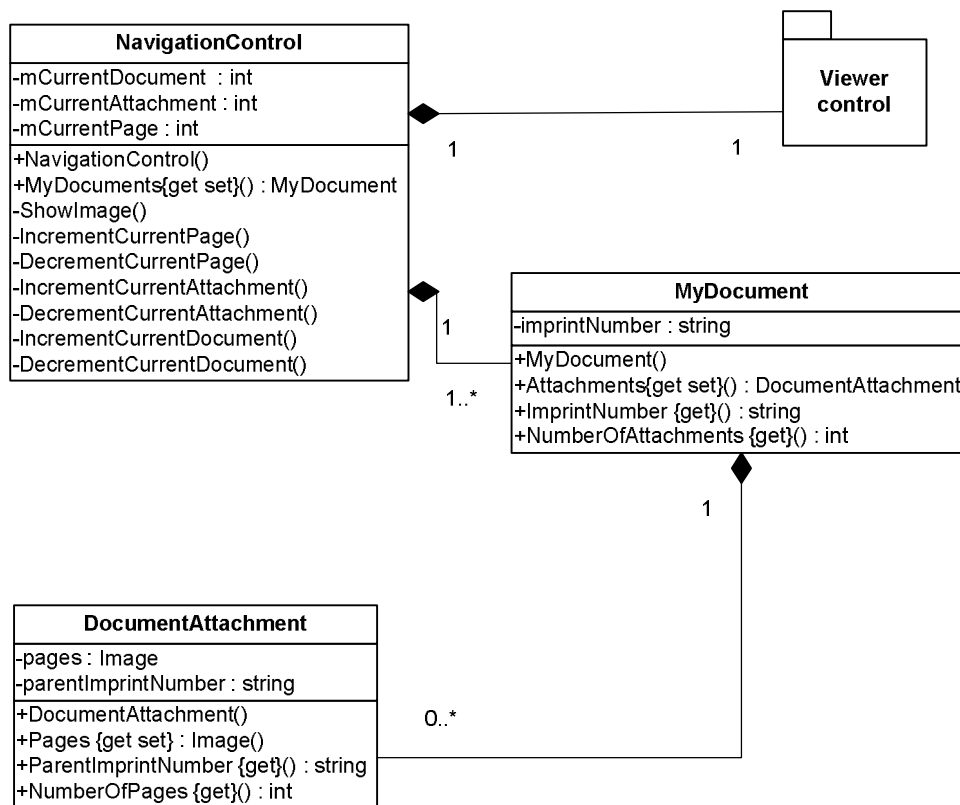
De betekenis van de knoppen van links naar rechts: *Eerste document*, *Vorige document*, *Volgende document* en *Laatste document*

Mogelijke uit te voeren acties met het keyboard

Naar de knoppen *PgUp* en *PgDn* wordt geluisterd. Bij het drukken van deze knoppen zal de pagina navigatie aangeroepen worden.

4. Details ontwerp

In dit hoofdstuk zijn de details van het ontwerp van de Navigatie Control weergegeven. Dit is gedaan met behulp van het klassendiagram, gevolgd door een uitleg behorende bij de klassen.



Figuur 2: Klassendiagram navigatie control

NavigatieControl Dit is het hoofdscherm van de navigatie control. Dit is de user interface met de button- en key events die kunnen worden aangeroepen door de eindgebruikers.

MyDocument Deze klasse stelt een document met één of meerdere DocumentAttachments voor.

DocumentAttachment Deze klasse stelt een bijlage met één of meerder pagina's (images) voor

De NavigatieControl wordt opgebouwd uit een toolstrip en daarboven de Viewer control. Bij elke keer als wordt gevraagd te navigeren naar een andere pagina, bijlage of document zal de navigatie control het betreffende image ophalen en ter tentoonstelling geven aan de viewer control.

Bijlage I: SDD converter control

BSc-project IN3405



Design Document Converter control

Projectgroep:

Abdoelrasheed Jarmohamed
Kishen Gajadhar

Commissie:

Job Scheffers (Zorgverzekeraar DSW, begeleider)
B.R. Sodoyer (TU Delft, coördinator / begeleider)

Omgeving:

Zorgverzekeraar DSW
Technische Universiteit Delft
Faculteit EWI Technische Informatica
Variant Softwaretechnologie

Datum: 26 september 2008

Status: wachten op acceptatie

Versie: 1.0

©September 2008 Zorgverzekeraar DSW

Alle in dit verslag afgebeelde gegevens, waaronder teksten, foto's, illustraties, grafische materiaal, (handels)namen, logo's, waren- en dienstmerken, zijn in eigendom van zorgverzekeraar DSW en worden beschermd door auteursrecht, merkenrecht en/of enig ander intellectueel eigendomsrecht.

Inhoudsopgave

1. Inleiding	136
2. Overzicht ontwerp	137
3. User Interface	138
4. Details ontwerp	140

1. Inleiding

De analyse van de *converter control* hebben we afgerond. Tijdens die analyse hebben we de benodigde informatie verzameld die nodig is om de converter control te ontwikkelen. In de ontwerpfase zal aan de hand van deze informatie het ontwerp worden gemaakt. En aan de hand van het ontwerp zal dan geïmplementeerd worden. De ontwerpfase zal dus een belangrijke fase zijn, hierin zullen alle beslissingen worden genomen over de te ontwikkelen software. Verkeerde beslissingen en een verkeerd ontwerp zullen leiden naar een verkeerde of slechte implementatie.

In de ontwerpfase zal er dus worden voortgebouwd op de analysefase. De keuzes en eisen van de werkgever zullen worden vertaald naar een ontwerp. In het geval van een converter control zal er geen groot en ingewikkeld ontwerp nodig zijn. Dit onder andere omdat het ontwerp van deze control geen user interface omvat.

In hoofdstuk 2 wordt een overzicht gegeven van het ontwerp, gevolgd door een gedetailleerde beschrijving van het systeem in hoofdstuk 3.

2. Overzicht ontwerp

In dit hoofdstuk wordt globaal beschreven hoe het ontwerp van de converter control in elkaar zit.

Deze control heeft als hoofddoel het ophalen van een document met gegeven *InfoImageId* en het laden van zo'n document in de navigatie control. Daarnaast dienen niet-image bestanden extern geopend te kunnen worden. Om dit te realiseren wordt gebruik gemaakt van een ActiveX control, *calactX*. Deze is ook gebruikt in de oude image viewer.

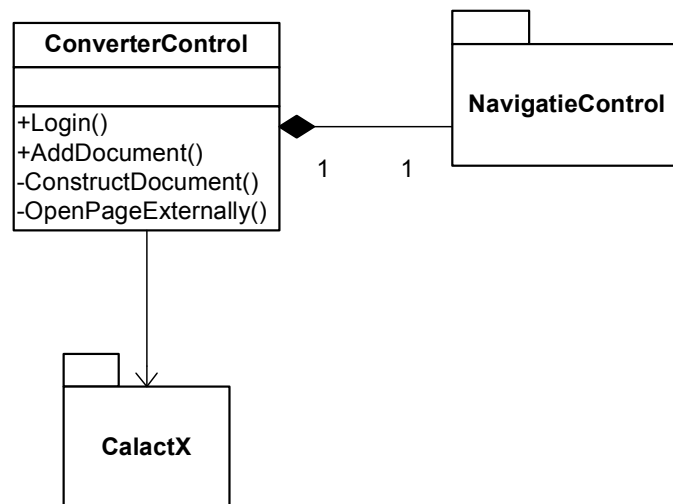
De converter control zal als volgt werken. Als invoer krijgt het een lijst van *InfoImageId*'s. Na in gelogd te zijn op *InfoImage* worden de documenten behorende bij de gekregen id's opgehaald. Per document wordt dan iedere pagina gelaadt in een *MyDocument* object. Voor niet-image bijlagen wordt een standaard image geladen, welke, als getoond, aan de gebruiker duidelijk maakt dat het gewenste pagina in een externe applicatie geopend zal worden.

Als de gebruiker via de navigatie control naar zo'n standaard image navigeert, wordt de converter control aangeroepen om het betreffende bestand via *calactX* extern te openen.

3. Details ontwerp

In dit hoofdstuk zijn de details van het ontwerp van de Converter Control weergegeven. Dit is gedaan met behulp van het klassendiagram, gevolgd door een uitleg behorende bij de klassen.

Het klassendiagram is te zien in *figuur 2*.



Figuur 2: Klassendiagram converter control

ConverterControl

De enige zelf gemaakte klasse van dit control. Qua user interface bestaat het uit een *user control* met daarin een instantie van de *navigatie* control. De klasse maakt ook gebruik van de *CalactX* namespace, het *ActiveX* control.

Bijlage J: Acceptatie & Test plan

BSc-project IN3405



Acceptatie & Test Plan

Viewer control

Projectgroep:

Abdoelrasheed Jarmohamed
Kishen Gajadhar

Commissie:

Job Scheffers (Zorgverzekeraar DSW, begeleider)
B.R. Sodoyer (TU Delft, coördinator / begeleider)

Omgeving:

Zorgverzekeraar DSW
Technische Universiteit Delft
Faculteit EWI Technische Informatica
Variant Softwaretechnologie

Datum: 21 augustus 2008

Status: wachten op acceptatie

Versie: 1.0

©Augustus 2008 Zorgverzekeraar DSW

Alle in dit verslag afgebeelde gegevens, waaronder teksten, foto's, illustraties, grafische materiaal, (handels)namen, logo's, waren- en dienstmerken, zijn in eigendom van zorgverzekeraar DSW en worden beschermd door auteursrecht, merkenrecht en/of enig ander intellectueel eigendomsrecht.

InhoudsOpgave

1. Inleiding.....	148
2. Acceptatie plan.....	149
3. Test plan	150
3.1 Aanpak	150
3.2 Te testen functionaliteiten	151
3.3 Test Cases	152

1. Inleiding

Voor dat met de implementatie begonnen wordt is het belangrijk de acceptatie criteria te formuleren. Ook is het belangrijk te specificeren hoe de test fase doorlopen zal worden, welke functionaliteit op welke manier getest moet worden. Immers het is de bedoeling een control met zomin mogelijke fouten op te leveren. Tijdens de implementatie zal daarom aan de hand van dit document zoveel mogelijk fouten afgehandeld worden.

Het doel van dit document is een overzicht te geven van de test fase van de te ontwikkelen control . Er wordt, in het acceptatie plan, eerst aangegeven onder welke voorwaarden het product geaccepteerd zal worden door de opdrachtgever. Vervolgens wordt, in het test plan, de aanpak van het testen besproken, gevolgd door een specificatie van de te testen functionaliteiten. Uiteindelijk worden de verschillende test cases uitgewerkt.

2. Acceptatie plan

In dit hoofdstuk worden de eisen voor de acceptatie van de Image Viewer Control besproken. Er zal dus worden aangegeven wanneer de opdrachtgever de opgeleverde systemen accepteert.

Acceptatie door Opdrachtgever

Voor dit project werd de opdracht opgedeeld in verschillende controls namelijk een *Image Viewer*, een *Doc* en een *Map* control (zie RAD). Als belangrijkste eis voor DSW geldt een compleet en correct werkende *Image Viewer* control. Met de overige controls wordt gestart als er voldoende tijd beschikbaar is.

De volgende eisen gelden voor elk van de aparte controls:

Implementatie

Ten eerste moeten alle *must have* (functionele) requirements geïmplementeerd zijn. Voor de non-functionele requirements geldt dit voor de *quality* requirements. Wanneer het systeem met alle *must have* requirements opgeleverd wordt, wordt aan de eerste eis voor acceptatie voldaan.

Testen

Ten tweede moeten de in het testplan(zie hoofdstuk 3) beschreven test gevallen geslaagd zijn. Voornamelijk de testcases voor de *must have*s moeten slagen. Er geldt dat in dat geval alle testgevallen uitgevoerd moeten zijn.

Commentaar

Bij code moet commentaar worden gegeven zoals beschreven binnen de DSW Standaards. Er moet bijvoorbeeld bij iedere functie bovenaan commentaar gegeven worden alsook bij elke ingewikkeld stukje code.

Documentatie

Voor documentatie geldt dat het commentaar bij iedere functie/methode volstaat. Er is niet noodzakelijk behoefte aan documentatie.

3. Test plan

In dit hoofdstuk volgt het test plan. Hierin wordt gespecificeerd hoe we zullen testen, de aanpak, wat getest zal worden en uiteindelijk de testcases.

3.1 Aanpak

Automatisch en Handmatig testen vanaf GUI

Automatisch testen wordt gedaan door een programma. De te testen functies worden aangeroepen door het testprogramma en voorzien van verschillende invoer waarden. De controle vindt plaats door de verkregen uitvoer te vergelijken met de verwachte. De verwachte uitvoer moet natuurlijk ook gespecificeerd worden. Het voordeel van automatisch testen ligt in consistentie van de tests tijdens en na implementatie.

Handmatig testen vanaf de GUI ‘Graphical User Interface’ betekent dat gebruikers het systeem testen door invoer te geven in invoervelden, acties uit te voeren en het resultaat te ontvangen. Voor gegeven invoer kan hierbij dus ook de gegeven uitvoer door het systeem worden vergeleken met verwachte uitvoer. De testcases worden in dit geval handmatig door de gebruiker afgewerkt.

Voor het uitvoeren van automatische tests bestaat voor Java-code het zogenaamde JUnit. Voor automatisch testen binnen de .Net omgeving bestaat NUnit. Deze is geïntegreerd in Visual Studio, waardoor test klassen gegenereerd kunnen worden. Voor ons rest dan nog het specificeren van in- en uitvoer waarden.

Automatische tests zullen na belangrijke wijzigingen in de implementatie worden doorlopen. Hierbij worden de specifieke testcases voor het gewijzigde onderdeel in de implementatie doorlopen. Daarnaast zal iedere dag of enkele dagen de complete ‘testsuite’ automatisch worden doorlopen, om zo het complete systeem te testen.

Handmatige testcases zullen door de ontwikkelaars zelf en eindgebruiker(s) gedaan worden door deze te doorlopen en te controleren. Deze handmatige testcases zullen door de ontwikkelaars gedurende ontwikkeling meerdere malen doorlopen worden. Eindgebruikers zullen deze tests uitvoeren in het kader van de ‘acceptatie’ waarin bepaald wordt of de gebruiker het opgeleverde systeem zal accepteren. Alle testcases moeten in dit geval slagen. Deze ‘acceptatie’ zal aan het einde eenmalig plaatsvinden.

Functionele en structurele tests

Er kan bij testen een onderscheid worden gemaakt tussen functioneel (black-box) en structureel (white-box) testen. *Functioneel* testen betekent dat voor het kiezen van testwaarden de implementatie van de te testen functie en/of methode niet relevant is. Het is slechts van belang te weten wat voor de gegeven invoer de verwachte uitvoer is, ongeacht de implementatie van de functie en/of methode.

Structureel testen houdt juist wel rekening met de implementatie van de betreffende functie en/of methode. Door naar de implementatie te kijken, kunnen specifieke slimme testwaarden worden gekozen die inspelen op de implementatie. Er kan bijvoorbeeld worden ingespeeld op het bepaald aantal keren doorlopen van *loops*, uitvoeren *if* statements en dergelijke. Er kunnen met dit type tests specifieke fouten in de implementatie worden gevonden.

Bij dit project zal een combinatie van beide test principes gebruikt worden, het zogenaamde *grey-box testing*. Dit elimineert de beperkingen van het gebruik van een test principe. Voor de structurele tests

zullen de testwaarden tijdens implementatie aangepast worden. De waarden voor het testen van de methoden en de klassen worden gekozen door middel van ‘domein analyse’. Hierbij zal rekening worden gehouden met pre- en post condities. De testwaarden voor de functionele tests worden gebaseerd op willekeurige testwaarden die vooraf realistisch worden gekozen vanuit de requirements.

Coverage door testen

Coverage geeft aan hoeveel procent van de code daadwerkelijk uitgevoerd is door uitvoering van de testcases. Door het gebruik van bijvoorbeeld *if* statements is het mogelijk dat een deel van de code niet bereikt wordt. Dat gedeelte van de code zou wel een fout kunnen bevatten. Door coverage te meten kan tot zekere hoogte worden gegarandeerd dat de code voldoende getest is. Een percentage van 100% geeft echter nog niet de garantie van foutvrije code. Het percentage garandeert wel dat alle code in dit geval is geraakt door het uitvoeren van de testcases.

NCover is een veel gebruikte *coverage* tool voor C#. Dit zal in ons geval ook gebruikt worden. Voor de verschillende testcases zullen de testwaarden zo worden gekozen dat geneste *loops* en *if* statements allemaal worden geraakt. Hier is er sprake van *structureel* testen aangezien naar de code gekeken wordt om test waarden te bepalen, die een hogere coverage garanderen. Tijdens de implementatie zullen de test waarden daarom ongetwijfeld aangepast worden.

3.2 Te testen functionaliteiten

In deze paragraaf worden de functionaliteiten die per control getest zullen worden, opgesomd. Ook wordt aangegeven als die tests automatisch of handmatig en functioneel of structureel zijn.

Viewer control

Te testen functionaliteit:

- zoomen
- roteren
- inverteren
- fitmodes: height, width, full & none
- pannen

De tests op bovenstaande functies zullen op zowel handmatige en automatische wijze uitgevoerd worden. De handmatige tests zullen inhouden dat via het GUI de aanroep naar een bepaalde functie zal plaatsvinden, vervolgt door een inspectie naar correctheid. Voor het automatische gedeelte zullen de individuele methoden en hulpmethoden van elk van deze functies getest worden.

De automatische tests zullen, afhankelijk van de te testen methoden, vallen onder zowel black- als white-box testen. De handmatige tests zullen vallen onder black-box of functioneel testen.

Onder de non-functionele eisen zijn er twee testbare gevallen. Die zijn de *performance* en de *Response time* requirements. Deze zullen beide op handmatige wijze getest worden.

Navigatie control

De belangrijkste functie van dit control is het navigeren. Dat zal automatisch en handmatig getest worden. Verder zullen we op automatische wijze het toevoegen van documenten aan de map testen.

Converter control

Omdat dit control over het algemeen alleen functies van het (werkende) AcitveX control aanroept, zal het alleen op handmatige wijze getest worden. Test documenten zullen worden bekeken met de oude image viewer en daarna met de nieuwe. De twee image viewers zouden dan een aantal images in dezelfde volgorde moeten laten zien.

3.3 Test Cases

In deze paragraaf volgen de test cases voor elk van de controls.

Viewer control

De test cases voor de viewer control

Test case 1	Zoomen-1
Doel:	De controle op juistheid van het formaat van het image.
Invoer:	Een incorrecte zoom factor (< het minimum of > het maximum).
Verwachte uitvoer:	Het image wordt op de minimale of maximale zoom factor getoond.
Faal/Slaag criteria:	Slaagt als het image op de minimale of maximale zoom factor getoond is.

Test case 2	Zoomen-2
Doel:	De controle op juistheid van het formaat van het image.
Invoer:	Een correcte zoom factor (tussen het minimum en maximum).
Verwachte uitvoer:	Het image wordt op de gespecificeerde zoom factor getoond.
Faal/Slaag criteria:	Slaagt als het image op de gespecificeerde zoom factor getoond is.

Test case 3	Roteren
Doel:	De controle op het correct roteren van het image.
Invoer:	Er wordt naar links of rechts geroteerd.
Verwachte uitvoer:	Het image is naar links of rechts geroteerd.
Faal/Slaag criteria:	Slaagt als het image naar links of rechts geroteerd is.

Test case 4	Inverteren
Doel:	De controle op het correct inverteren van het image.
Invoer:	Het image wordt geïnverteerd.
Verwachte uitvoer:	Het image is geïnverteerd.
Faal/Slaag criteria:	Slaagt als het image geïnverteerd is.

Test case 5	FitNone
Doel:	De controle op de juistheid van het formaat van het image in fitNone mode.
Invoer:	Het image wordt <i>gefit</i> in de None mode.
Verwachte uitvoer:	Het image wordt op zijn originele formaat getoond, met een zoomfactor van 100% dus.
Faal/Slaag criteria:	Slaagt als het image op zijn originele formaat getoond is.

Test case 6	FitWidth
Doel:	De controle op de juistheid van het formaat van het image in fitWidth mode.
Invoer:	Het image wordt <i>gefit</i> in de Width mode.
Verwachte uitvoer:	Het image wordt zodanig getoond dat het qua breedte precies in het venster past.
Faal/Slaag criteria:	Slaagt als het image qua breedte precies in het venster past.

Test case 7	FitHeight
Doel:	De controle op de juistheid van het formaat van het image in fitHeight mode.
Invoer:	Het image wordt <i>gefit</i> in de Height mode.
Verwachte uitvoer:	Het image wordt zodanig getoond dat het qua hoogte precies in het venster past.
Faal/Slaag criteria:	Slaagt als het image qua hoogte precies in het venster past.

Test case 8	FitFull
Doel:	De controle op de juistheid van het formaat van het image in fitFull mode.
Invoer:	Het image wordt <i>gefit</i> in de Full mode.
Verwachte uitvoer:	Het image wordt zodanig getoond dat het qua hoogte én breedte precies in het venster past.
Faal/Slaag criteria:	Slaagt als het image qua hoogte én breedte precies in het venster past.

Test case 9	Pannen
Doel:	De controle op de juistheid van de locatie van het image.
Invoer:	Een nieuwe locatie
Verwachte uitvoer:	Het image wordt geplaatst op de gegeven locatie.
Faal/Slaag criteria:	Slaagt als het image op de gegeven locatie geplaatst is.

Navigatie control

De test cases voor de navigatie control

Test case 1	Navigeer naar volgende pagina-1
Doel:	Controle of de volgende pagina is getoond
Invoer:	Er wordt van een laatste image uit een niet-laatste document naar de volgende image genavigeerd.
Verwachte uitvoer:	De volgende image, de eerste van het volgende attachment en document, is getoond
Faal/Slaag criteria:	Slaagt als de volgende image getoond is.

Test case 2	Navigeer naar volgende pagina-2
Doel:	Controle of dezelfde pagina wordt getoond
Invoer:	Er wordt van een laatste image uit een laatste document naar de volgende image genavigeerd.
Verwachte uitvoer:	Dezelfde image is getoond
Faal/Slaag criteria:	Slaagt als dezelfde image is getoond

Test case 3	Navigeer naar vorige pagina-1
Doel:	Controle of de vorige pagina is getoond
Invoer:	Er wordt van een eerste image uit een niet-eerste document naar de vorige image genavigeerd.
Verwachte uitvoer:	De vorige image, de laatste van het vorige attachment en document, is getoond
Faal/Slaag criteria:	Slaagt als de vorige image getoond is

Test case 4	Navigeer naar vorige pagina-2
Doel:	Controle of dezelfde pagina wordt getoond
Invoer:	Er wordt van een eerste image uit een eerste document naar de vorige image genavigeerd.
Verwachte uitvoer:	Dezelfde image is getoond
Faal/Slaag criteria:	Slaagt als dezelfde image is getoond

Test case 5	Document toevoegen-1
Doel:	Controle of het document is toegevoegd en getoond
Invoer:	Een document wordt toegevoegd wanneer nog geen documenten staan getoond
Verwachte uitvoer:	Een nieuwe document array is aangemaakt met daarin het toegevoegde document. Het document wordt ook getoond
Faal/Slaag criteria:	Slaagt als het toegevoegde document getoond is.

Test case 6	Document toevoegen-2
Doel:	Controle of het document is toegevoegd en getoond
Invoer:	Een document wordt toegevoegd wanneer al een document is getoond
Verwachte uitvoer:	Het document wordt toegevoegd aan het eind van de document array en wordt getoond
Faal/Slaag criteria:	Slaagt als het toegevoegde document getoond is.

Test case 7	Document toevoegen-3
Doel:	Controle of het document is getoond
Invoer:	Een document wordt toegevoegd en het document al voorkomt in de array van documenten
Verwachte uitvoer:	Geen enkel document wordt toegevoegd, maar het gewenste document wordt getoond
Faal/Slaag criteria:	Slaagt als het gewenste document getoond is.

Converter control

De test cases voor de converter control

Test case 1	Inloggen
Doel:	Controle of ingelogd is op InfoImage
Invoer:	Een gebruikersnaam, password en domein
Verwachte uitvoer:	Er is ingelogd met de gegeven gebruikersnaam
Faal/Slaag criteria:	Slaagt als ingelogd is op InfoImage

Test case 2	Document ophalen en tonen
Doel:	Controle of het juiste document is opgehaald en getoond
Invoer:	Een infoImageId van het gewenste document
Verwachte uitvoer:	Het gewenste document is opgehaald met de eerste pagina daarvan getoond
Faal/Slaag criteria:	Slaagt als het gewenste document opgehaald en de eerste pagina daarvan getoond is.

Test case 3	Bestand extern openen
Doel:	Controle of het juiste bestand extern is geopend
Invoer:	Een infoImageId van een document met een <i>MSOffice</i> document als bijlage
Verwachte uitvoer:	In plaats van de bijlage is een standaard informatie image getoond en de gewenste bijlage is extern geopend
Faal/Slaag criteria:	Slaagt als de standaard informatie image getoond is en de gewenste bijlage extern geopend is