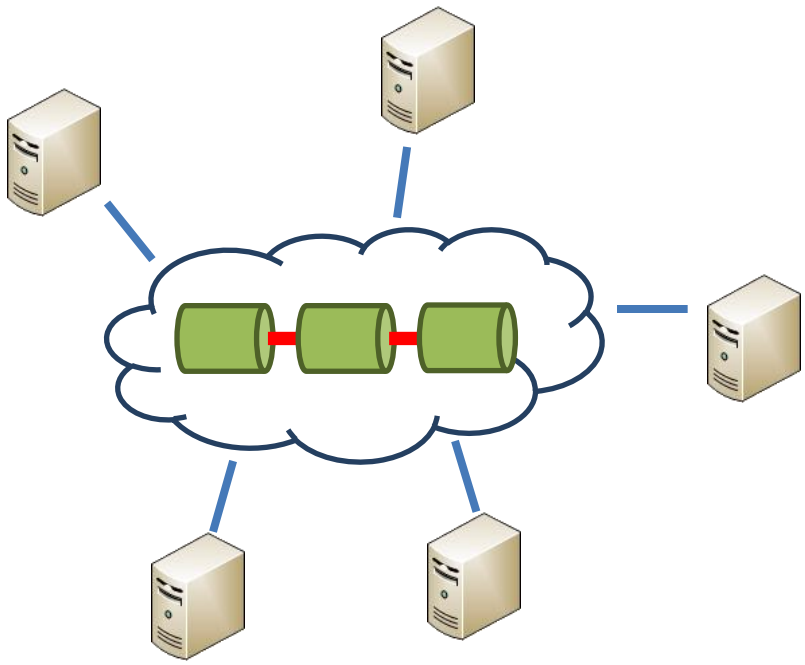
MEMORIA MEDIA

V.I.P.

Video Internet Pipeline



Versie 1.0

Jasper Lammers
Eindhoven, 07-06-2011

Memoria Media
Kanaaldijk Zuid 13
5611 VA Eindhoven
Nederland

www.memoriamedia.nl

T +31 40 7853177
F +31 40 7853178
info@memoriamedia.nl

V.I.P.

"Video Internet Pipeline"

Student:	Jasper Lammers
Student nummer:	2108351
Begin studie:	2007
Afstudeer periode:	07-02-2011 t/m 24-06-2011
Opleiding:	Technische Informatica, voltijd
Bedrijf:	Memoria Media
Locatie:	Eindhoven
Bedrijfsbegeleider:	Erwin Wullems, Algemeen Directeur
Technisch begeleider:	Henk Jooren
Docentbegeleider:	Peter Lambooij
Date:	07-06-2011
Document release:	01
Document status:	Finished

Voorwoord

Dit verslag is geschreven naar aanleiding van de afstudeerstage die plaatsvindt in het vierde leerjaar van de opleiding Technische Informatica bij de Fontys Hogeschool Eindhoven. In de periode van februari 2011 tot en met juni 2011 heb ik stage gelopen bij Memoria Media in Eindhoven. Daar heb ik onderzoek verricht, een ontwerp gemaakt en een deel van de implementatie uitgevoerd voor een framework dat video filtering kan uitvoeren met behulp van meerdere computers.

In dit verslag geef ik toelichting over de opdracht, mijn werkwijze en zullen de keuzes die zijn gemaakt worden verantwoord. Er wordt verslag gedaan van het hele ontwikkeltraject, van het onderzoek in het begin tot het opgeleverde eindproduct. De stage beleving zal aan bod komen in het hoofdstuk evaluatie.

Ik wil mijn bedrijfsbegeleider Erwin Wullems bedanken voor zijn betrokkenheid en begeleiding. Daarnaast wil ik Henk Jooren, mijn technische begeleider bedanken voor zijn technische ondersteuning. Ook wil ik mijn dank uiten naar Peter Lambooij, mijn docentbegeleider, voor het bewaken van het proces en zijn waardevolle feedback en hulp bij het schrijven van het afstudeerverslag.

Jasper Lammers
Eindhoven, 07-06-2011

Inhoudsopgave

Samenvatting

Memoria Media maakt met behulp van een zelf ontwikkeld camerasysteem films van uitvaarten die het bedrijf live bewerkt en vervolgens over het internet kan laten zien. Na vijf jaar deze oplossing gebruikt te hebben wil het bedrijf het bestaande hardware systeem gaan vervangen door software.

In 2010 heeft Memoria Media al een prototype voor deze software laten ontwikkelen. Dit prototype voldeed echter niet helemaal aan de functionele eisen en stabiliteits eisen die hieraan werden gesteld. Ook was het nodig om een library die in dit oude prototype gebruikt werd te vervangen door een goedkoper alternatief. Het was dan ook de afstudeeropdracht om te onderzoeken wat er gedaan moet worden om bij de ontwikkeling van een nieuw software systeem wel de gewenste functionaliteit en stabiliteit te krijgen, en waardoor de library het beste vervangen kon worden. Na dit onderzoek moest er een nieuw software systeem ontworpen en geïmplementeerd worden, om uiteindelijk het oude prototype te kunnen vervangen.

Het grootste deel van de afstudeerperiode is besteed aan het onderzoeken van mogelijkheden om dit project het beste uit te voeren. Hierbij zijn technieken zoals DirectShow en patronen zoals cloud computing en pipeline architectuur onderzocht. Uit deze onderzoeken bleek dat het software systeem het beste op meerdere computers kan draaien, die samen computerkracht delen om taken uit te voeren.

Om dit te kunnen realiseren is een framework ontworpen. Hierbij is het onderzoek toegepast, en onder andere rekening gehouden met herbruikbaarheid en uitbreidbaarheid. Hiervoor is het “plug-in” patroon meerdere malen terug te vinden. Dit framework wordt het V.I.P. (Video Internet Pipeline) framework genoemd.

Uiteindelijk is er een implementatie gemaakt van het V.I.P. framework. Omdat het onderzoek toch groter en belangrijker bleek dan oorspronkelijk was gedacht, is de prioriteit van de implementatie lager geworden. Dit heeft er voor gezorgd dat de implementatie niet meer helemaal binnen de afstudeer periode paste. Om deze reden moest er een stukje overdracht plaatsvinden wat het bedrijf in staat moet stellen de implementatie te kunnen voltooien.

Hoe het oude prototype en de library precies vervangen worden door het V.I.P framework is in het tabel weergegeven.

	Naam systeem	(Digitale video) library
Oud	Oude prototype	Medialooks
Nieuw	V.I.P Framework	DirectShow

Wel is er toegekomen om een systeem te maken wat een cloud kan vormen en een pipeline kan draaien. Dit systeem kan data bewerken volgens een gegeven patroon (de pipeline) en wordt aangestuurd via een netwerk of internet.

Summary

Using a self-developed camera system, Memoria Media makes movies of funerals. These movies are edited on the spot and are streamed over the internet. After five years of using this solution, the company wants to replace this hardware system with software.

In 2010, Memoria Media already developed a prototype for this software. However, this did not conform to the functionality and the stability that was desired. Besides this, the prototype uses a library which had to be replaced by a cheaper alternative. It was the graduation assignment to research what had to be done to develop a software system that would conform to the functionality and stability norm, and which library would be best to use as a replacement for the original. After this research, a new software system had to be designed and implemented. This software system should then eventually replace the old prototype.

The biggest part of the graduation period was spend researching the best options to realize the project. This research included techniques such as DirectShow and patterns as cloud computing and the pipeline architecture. The conclusion that could be drawn from these researches was that the software system could best be distributed on multiple computers that would share their computer power to complete tasks.

To realize this, a framework was designed. Doing this, the research was used and it was designed to be reusable and extensible. For this, the “plug-in” pattern was applied multiple times. The name of the framework is V.I.P. (Video Internet Pipeline).

Eventually, an implementation of the V.I.P. framework was made. Because the research was bigger and more important than initially planned, the priority of this implementation became lower. Because of this the complete implementation of the project could not be realized in the time that was reserved for the graduation period. This made it necessary to transfer the project to the company, so they can finish it.

How the old prototype and library were replaced by the V.I.P. framework is made visible in the table.

	Name systeem	(Digital video) library
Old	Old prototype	Medialooks
New	V.I.P Framework	DirectShow

What was eventually implemented was a system which could form a cloud that was able to support a pipeline. This system is able to manipulate data in a provided pattern (the pipeline) and can be controlled over a network or over the internet.

Verklarende woordenlijst

Term	Uitleg
AV	Audio/Video
AVI	Audio Video Interleave. Een verpakkingsformaat voor digitale audio en video.
Client	Een client is een applicatie of een computersysteem met toegang tot een ander systeem, de server, via een netwerk.
Cloud computing	Cloud computing is het via het internet beschikbaar stellen van hardware, software, gegevens en andere hulpbronnen. Google.nl is een mooi voorbeeld van een cloud computing systeem. De gebruiker geeft een opdracht aan een cloud systeem. Dit zorgt ervoor dat er snel gereageerd kan worden op miljoenen opdrachten binnen een korte tijd.
Codec	Een apparaat of programma wat kan encoderen en/of decoderen.
Decoding	Het decoderen van (gecodeerde) informatie. Het tegenovergestelde van Encoding.
Encoding	Het coderen van informatie.
Framework	Een (software)framework is een geheel van softwarecomponenten dat gebruikt kan worden bij het programmeren van applicaties.
Guid	Globally Unique Identifier
Het oude prototype	Applicatie die eerder door Memoria Media ontwikkeld is om het hardware systeem te vervangen.
Node	Een zelfstandig blok in een pipeline.
OS	Operating System (besturingssysteem).
Parsen	Een proces waarbij (digitale) informatie wordt geanalyseerd en wordt omgezet naar een ander formaat.
Peer	Gelijke.
Real time	Een taak in een door software gecontroleerd systeem wordt realtime genoemd (of "uitvoerbaar in reële tijd, in real time") als de gecombineerde reactie- en uitvoertijd van de taak korter is dan de maximale tijd die is toegestaan, rekening houdend met invloeden van buitenaf. In de video wereld houdt dit in dat de uitvoer van de data de juiste framerate behoudt.
Renderen	Het bereiden en presenteren van data om weer te geven aan de gebruiker. Bijvoorbeeld het genereren van een plaatje op het scherm of in een bestand. Of het afspelen van een geluid.
Scene	Een scene staat het toe om meerdere streams in een andere stream te zetten. In de afbeelding hieronder is een voorbeeld te zien van een scene. Linksonder staat een stream over de rest van het beeld heen.

	
SDK	Software Development Kit.
Server	Een server is een computer die, of een programma dat diensten verleent aan clients.
SOA	Service Oriënted Architecture.
SOAP	Simple Object Access Protocol.
TCP	Transmission Control Protocol.
Thread	Een thread (Engels voor draad) op een computer is een proces dat binnen een proces uitgevoerd wordt. Met behulp van threads kan een computerprogramma verschillende taken "tegelijkertijd" uitvoeren.
Transitie	Een transitie is de overgang tussen twee (video)streams. Dit kan met veel verschillende effecten zoals fade, schuiven, hard cut etc. In de afbeelding is een voorbeeld te zien van een transitie. Links is een stream van een hek te zien terwijl rechts vogels zijn weergegeven. De transitie zorgt voor de overgang tussen deze streams. 
UDP	User Datagram Protocol.
VIP	Video Internet Pipeline.
WCF	Windows Communication Foundation.

1. Inleiding

Digitale video wordt steeds populairder. Zo is het kijken van filmpjes op YouTube tegenwoordig heel gebruikelijk. Maar niet alleen wordt hier voor amusement gebruik van gemaakt. Zo maakt het bedrijf Memoria Media gebruik van digitale video over het internet om uitvaarten toegankelijk te maken voor mensen die deze niet kunnen bijwonen. Momenteel wordt hier een hardware oplossing voor gebruikt. In 2010 heeft Memoria Media een prototype laten ontwikkelen dat het mogelijk maakt om het streamen en bewerken van digitale video volledig met behulp van software uit te voeren. Dit prototype voldoet nog niet aan de eisen die gesteld worden aan de functionaliteit en de stabiliteit. De afstudeeropdracht is bedoeld om te onderzoeken wat er gedaan moet worden om bij de ontwikkeling van een nieuw software systeem wel de gewenste functionaliteit en stabiliteit te krijgen.

Dit verslag geeft inzicht in de achtergronden, werkwijze en resultaten van deze afstudeeropdracht. In het volgende hoofdstuk zal meer informatie worden gegeven over het bedrijf Memoria Media. Hoofdstuk 3 beschrijft de projectdoelen, de opdracht en de werkwijze die toegepast is om het doel te bereiken.

Tijdens de uitvoering van het afstudeerproject is vooral veel aandacht geschonken aan het onderzoeksaspect. Het onderzoek wordt beschreven in hoofdstuk 4. De uitwerking van de opdracht wordt vervolgens in hoofdstuk 5 behandeld.

Tot slot wordt dit verslag afgesloten met een conclusie en een evaluatie.

2. Het bedrijf: Memoria Media

2.1 Het bedrijf

Memoria Media is een klein bedrijf in Eindhoven dat in 2006 is opgericht door Erwin Wullems en Bart Robben. Later is er nog een werkneemster bijgekomen.

Memoria Media is een bedrijf dat als dienst het online volgen van uitvaarten mogelijk maakt. Zo kunnen mensen die door omstandigheden niet aanwezig kunnen zijn toch de uitvaart meebeleven. Ook wordt er een dvd geleverd waarop de uitvaart te zien is.

Het bedrijf heeft een prettige en informele werksfeer. Doordat het een klein bedrijf is zijn mensen goed met elkaar bekend en vertrouwd, en werken de werkgevers en de werknemers goed naast elkaar.

2.2 Visie

Uitvaart registratie voor iedereen beschikbaar maken.

2.3 Doel

Marktleider in Nederland worden en middels franchise formule een internationale speler zijn.

2.4 Dienst

De dienst die Memoria Media levert is het op afstand kunnen volgen van een uitvaartdienst. Hiervoor zijn bij een crematorium meerdere op afstand bestuurbare camera's geplaatst. Vervolgens kunnen de werknemers bij Memoria Media de beelden van deze camera's live mixen (overgangen maken en dergelijke) en de camera's besturen om een wat dynamischere film te maken.

Tegelijkertijd kunnen mensen inloggen en live meekijken met de uitvaart. Mochten er mensen zijn die niet kunnen komen bijvoorbeeld door ziekte of doordat ze in het buitenland zitten, dan is het voor hen toch mogelijk om de uitvaart op afstand bij te wonen.

Na afloop van de dienst maakt Memoria Media eventueel nog wat aanpassingen en verbeteringen in de opname, en zal er een dvd voor de nabestaanden worden gebrand.

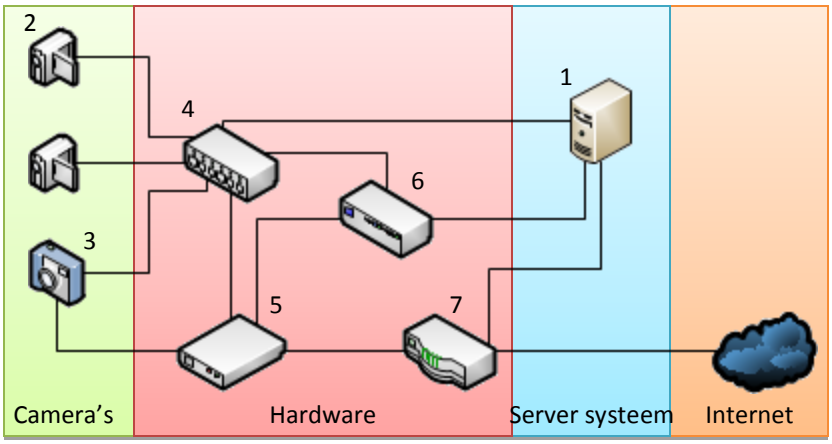
2.5 Technische begeleiding

Memoria Media is geen technisch bedrijf. Tijdens het uitvoeren van de afstudeeropdracht was er technische begeleiding verzorgd door iemand buiten het bedrijf. Deze technisch begeleider, genaamd Henk Jooren wordt vaker door Memoria Media ingehuurd voor software opdrachten of voor onderhoud aan de hardware systemen.

3. De opdracht

3.1 Aanleiding

Op dit moment gebruikt Memoria Media een simpel systeem wat met behulp van software een hardware mixer aanstuurt om transities te doen, en een video server om de camera's mee te kunnen aansturen. Dit systeem is weergegeven in figuur 3.1.1.



Figuur 3.1.1 - Het hardware systeem dat op het moment nog steeds actief in gebruik is.

1. Server systeem (Hier draait software op gemaakt door Henk Jooren)
2. Statische camera
3. Beweegbare camera
4. Video verdeler
5. Video server (Axis241Q
6. AV verdeler
7. Router

Het huidige systeem kost 20.000 euro wat erg duur is. Om dit goedkoper te maken en te vervangen is er in opdracht van Memoria Media een softwarepakket (het oude prototype[1]) ontwikkeld dat de videomixer kan vervangen. De aanzet van dit softwarepakket is in principe goed, maar niet stabiel en maakt gebruik van een softwarepakket (Medialooks[2]) dat ze in verband met kosten niet willen gebruiken. Daarom wordt deze software nog niet gebruikt.

In Figuur 3.1.2 is een overzicht te zien van de drie systemen die in dit verslag aan de orde zijn.

Systeem	Opmerking
Product A - Oorspronkelijk systeem (Hardware systeem wat wordt aangestuurd met software)	Momenteel in gebruik
Het oude prototype (Geheel software)	Niet in gebruik – niet stabiel
Te maken systeem (Product C of V.I.P. systeem)	Moet onderzocht/ontworpen/gemaakt worden door afstudeerder

Figuur 3.1.2 - Een overzicht van de drie systemen die in het verslag aan de orde zijn.

3.2 Doelstelling

Het doel van de opdracht is om een systeem te maken wat alle functionaliteit van het oude prototype biedt en hiernaast nog een aantal extra zaken te implementeren. Zo moet het nieuwe systeem uitbreidbaar zijn met meerdere camera's, en te gebruiken zijn op mobiele apparaten bijvoorbeeld een iPhone.

Hiernaast moet de Medialooks library wat in het oude prototype gebruikt wordt in dit nieuwe systeem vervangen worden door een goedkoop alternatief.

[1] Zie hoofdstuk 4.3 - Het oude prototype
[2] Zie hoofdstuk 4.2 - Medialooks library

3.3 De opdracht

Tijdens het traject is de opdracht een aantal keren veranderd. Dit kwam voornamelijk door uit onderzoek resulterende gesprekken met de klant. Zo werd elk keuzemoment met de klant besproken. De oorspronkelijke opdrachtomschrijving luidde als volgt:

Gegeven dat er nu een applicatie bestaat die door middel van een library, geleverd door Medialooks video en audio bewerking kan doen, moet er worden onderzocht of deze functionaliteit vervangen kan worden door een andere library, die ook geïmplementeerd kan worden op andere OS-en, met voorkeur Mac-OS.

Naar aanleiding van een interview met de klant, waarin voornamelijk werd gekeken wat echt het gewenste resultaat moest zijn, bleek de bovenstaande opdrachtformulering niet geheel tot het gewenste resultaat te leiden. Na met de klant te hebben overlegd is er een nieuwe opdracht geformuleerd die beter paste bij de wensen van de klant.

Maak een systeem dat AV mixing doet dat vergelijkbaar is met de functionaliteit die de Medialooks library biedt. Zorg er hierbij voor dat dit systeem gebruikt kan worden op meerdere OS-en en mobiele apparaten zoals een iPhone. Ook moet de Medialooks library vervangen worden door een alternatieve library.

Ook moest er tijdens de afstudeerstage veel aandacht aan documentatie geschonken worden, met nadruk tijdens het onderzoek.

Na het definiëren van de opdracht, is samen met de klant een planning opgesteld. Deze planning is in overleg met de klant in de loop van het project een aantal keer aangepast. De redenen worden in de loop van dit document duidelijk uitgelegd, maar de voornaamste oorzaak was dat, na een onderzoek er nieuwe opties en keuzes aan het licht kwamen, die de opdracht beïnvloedden.

Bovengenoemde opdrachtformulering is de formulering die geschreven is vóór het traject (onderzoek/ontwerp etc) doorlopen is.

3.4 Werkwijze

De eerste weken van de afstudeerstage zijn gebruikt voor het onderzoeken van de bestaande applicatie (het oude prototype) en het onderzoeken en vastleggen van de requirements. Om op een gestructureerde manier te kunnen werken is het project opgedeeld in verschillende fasen. In het project initiation document is gedefinieerd welke activiteiten in iedere fase zijn uitgevoerd.

De ontwikkelingsfase is door de afstudeerder ingedeeld in een onderzoek en uitwerking. Deze twee stappen zijn ook duidelijk terug te vinden in de afstudeerscriptie[1]. Verder komen niet alle fases als hoofdstukken in dit verslag terug, de noemenswaardige resultaten en keuzes die gemaakt zijn tijdens deze fases, staan wel in het verslag.

Tijdens de uitvoering van de ontwikkelingsfase is er wekelijks een gesprek geweest met de klant en de technische begeleider. Op deze manier is er veel sturing bij het uitvoeren van het project mogelijk geweest.

In overleg met de klant is besloten om zo veel mogelijk binnen het project in C# of Visual Basic te doen, gebruik makende van het .NET framework, versie 3.5 of versie 4.0. De reden hiervoor is dat de technisch begeleider die het project tevens over gaat nemen als de afstudeerperiode voorbij is het meest bedreven is in deze talen.

Omdat de afstudeerder zelf goed bedreven is in C#, is er voor gekozen om voornamelijk deze taal toe te passen.

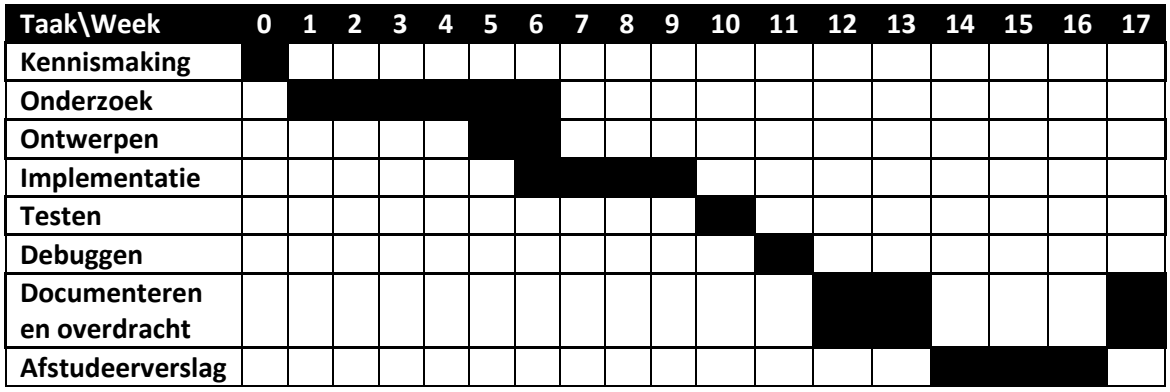
[1] Zie hoofdstuk 4 - Het onderzoek en hoofdstuk 5 - De uitwerking

Planning

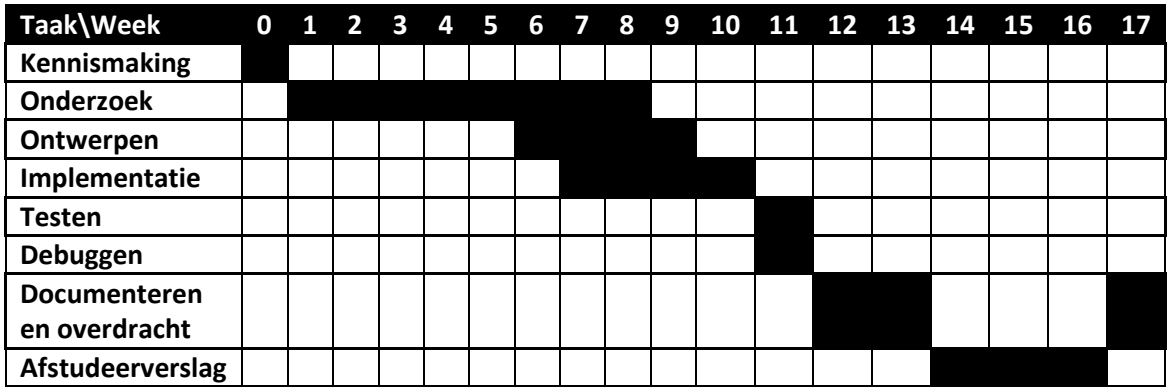
Voor het begin van het project is er een planning opgesteld voor het eindproduct. Er werd toen vanuit gegaan dat het project binnen de looptijd van de stage helemaal afgerond kon worden.

Na een korte tijd bleek dit echter niet het geval te zijn. Tijdens de eerste onderzoeken kwam de noodzaak tot meer onderzoeken naar boven en werd er in overleg met de klant voor gekozen om de onderzoekfase uit te breiden, en de planning aan te passen.

In de figuren 3.4.1 en 3.4.2 is te zien hoe de oorspronkelijke planning verschilt van de uiteindelijke uitvoering. Dit lijkt niet veel te zijn aangepast, maar behalve dat het onderzoek groter is geworden, is het eindproduct voor de afstudeerperiode kleiner geworden om het onderzoek meer ruimte te geven.



Figuur 3.4.1 – De oorspronkelijke planning



Figuur 3.4.2 – De uiteindelijke uitvoering

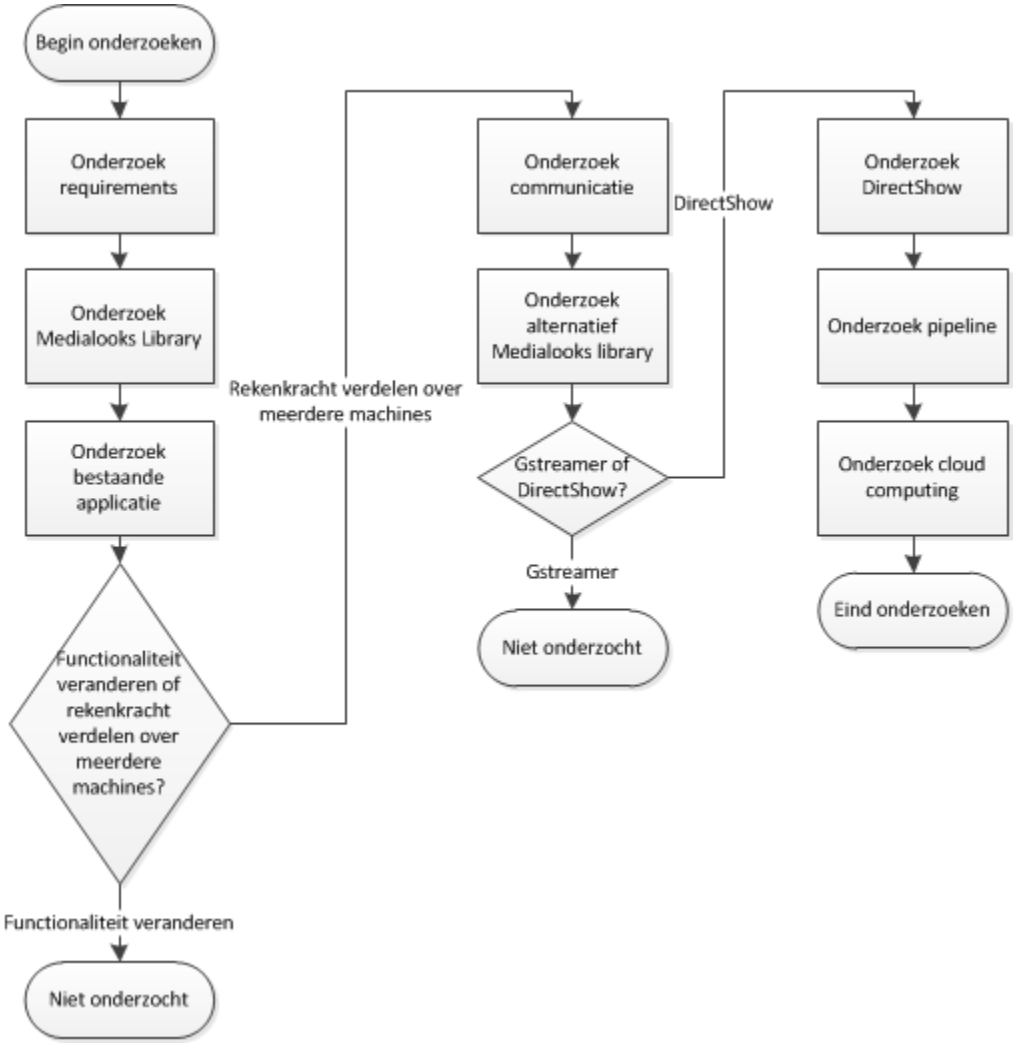
De prioriteit van het project lag dus bij het onderzoek, en vervolgens het ontwerp van de opdracht. De implementatie kreeg lage prioriteit, en zou gedaan worden indien er tijd over was, en afhankelijk van het resultaat van het onderzoek en het ontwerp.

Uiteindelijk is er wel aan de implementatie toegekomen en is er een demonstreerbaar programma gerealiseerd.

4. Het onderzoek

4.1 Inleiding

Een groot deel van dit project bestond uit onderzoeken die verricht moesten worden. In figuur 4.1.1 is het pad te zien wat genomen is tijdens de onderzoeksfase. Hier is ook te zien dat er keuzes zijn gemaakt naar aanleiding van verschillende onderzoeksresultaten. Deze keuzes zijn gemaakt in overleg met de klant. In de volgende hoofdstukken worden alle onderzoeken even kort behandeld, en wordt kort toegelicht waarom bepaalde keuzes zijn gemaakt.



Figuur 4.1.1 - Onderzoek flowchart

4.2 Medialooks library

In het oude prototype wordt gebruik gemaakt van de Medialooks library. Het is onderdeel van de opdracht om deze library te vervangen. Dit onderzoek is uitgevoerd om te bepalen wat de Medialooks library precies inhoudt. Op welke manier deze library het beste vervangen kan worden is later onderzocht[1]. De reden hiervoor is dat een alternatief af kon hangen van de uitkomsten van andere onderzoeken.

Medialooks

Medialooks is een bedrijf dat zich specialiseert in software die video en audio bewerking uitvoert. Complete producten worden niet door Medialooks geleverd. Ze maken plug-ins, filters en SDK's (software development kits).

Voor dit onderzoek is er gekeken welk product er in het oude prototype is toegepast. De informatie voor dit onderzoek is verkregen van de website van Medialooks.

Vision Mixer SDK

Vision mixer sdk is het product dat door Memoria Media gebruikt wordt in het oude prototype[2]. Deze sdk levert een library die de volgende kenmerkende functionaliteiten aanbiedt:

1. Real-time mixen van audio/video van live bronnen, streams en bestanden
2. Frame-nauwkeurig mixen van files
3. Playlist ondersteuning
4. Bewerkbaar transparant masker voor elke individuele video output
5. 2d en 3d transformaties
6. Transities
7. Scenes, het opslaan van scenes en het vloeiend overgaan van scenes
8. Tekst en plaatjes overlays
9. Ingebouwde audio/video synchronisatie
10. Ondersteuning voor verschillende input formaten
11. Windows media streaming
12. Opslaan naar avi of wmv bestanden
13. RTSP, MMS & http Windows media stream output
14. Meerdere optionele plug-ins

De library is gemaakt met het Microsoft platform DirectShow.

[1] Zie hoofdstuk 4.6 - Alternatief Medialooks library
[2] Zie hoofdstuk 4.3 - Het oude prototype

Feedback

Na de uitkomsten van dit onderzoek besproken te hebben met de klant, bleek dat de library toch meer functionaliteit bood dan oorspronkelijk bedoeld werd met de opdracht. Het zou al voldoende zijn als het product dezelfde AV-mixing functionaliteiten had als het oude prototype. De overige functionaliteiten zijn wel zeer gewenst.

Naar aanleiding van dit onderzoek was de opdracht dus al veranderd. Welke van de functionaliteiten van de Video Mixer SDK gebruikt worden in het oude prototype wordt in het volgende hoofdstuk beschreven.

Conclusies

- Met de “Medialooks library” wordt de library bedoeld die geleverd wordt bij het product “Vision Mixer SDK” van het bedrijf Medialooks
- Niet alle functionaliteiten die geboden worden door deze library hoeven in het project te worden geïmplementeerd. De functionaliteiten die zijn toegepast in het oude prototype zijn voldoende.
- Er moet nog onderzocht worden welke functionaliteiten die de Medialooks library aanbiedt daadwerkelijk gebruikt worden in het oude prototype.

4.3 Het oude prototype

Het oude prototype is eerder door de klant ontwikkeld om het hardware camerasysteem te vervangen. Het oude prototype wordt niet gebruikt omdat deze niet stabiel genoeg is.

Uit het vorige onderzoek bleek dat het systeem wat gemaakt moest worden de functionaliteit moet hebben die in het oude prototype wordt aangeboden. Om deze reden is er een onderzoek gedaan naar het oude prototype waaruit moet blijken wat deze functionaliteit precies is. Ook is er gekeken welke functionaliteit van de Vision Mixer SDK er gebruikt wordt in het oude prototype.

Er is begonnen met het uitproberen van deze software om een goed idee te krijgen wat deze precies kan, en waar deze precies voor bedoeld is. Vervolgens is er in de code van de software gekeken om te bepalen welke functionaliteiten van de Vision Mixer SDK er gebruikt worden in deze software.

Functionaliteit en gebruik Vision Mixer SDK

Het oude prototype heeft de volgende functionaliteiten:

1. Er kunnen tot vier input streams gekozen worden.
2. Een input stream kan een video bestand zijn, of een video driver/device.
3. Het volume van de audio van elke individuele input kan worden veranderd.
4. De applicatie heeft één output.
5. De output stream kan naar een video bestand worden opgeslagen (wmv).
6. De output stream kan over het internet worden gestreamed. Dit kan zowel direct (http) als worden gepushed met mms.
7. De input streams kunnen met een transitie met elkaar worden verwisseld.
8. Er kunnen meerdere input streams tegelijk zichtbaar zijn in de output door gebruik te maken van een scene.
9. Alle inputs kunnen worden gepauzeerd.
10. Het streamen/opnemen kan worden gestart en gestopt.
11. Het lokale volume van de output kan worden veranderd.
12. Het streamingvolume van de output kan worden veranderd.
13. De snelheid waarmee van scene wordt gewisseld of waarmee een transitie gedaan wordt kan worden veranderd.

In figuur 4.3.1 is in een tabel te zien welke functionaliteiten van de Vision Mixer SDK gebruikt worden om de functionaliteiten van het oude prototype te verwezenlijken. Als we deze functies op een rij zetten blijven de volgende functies over:

Functie in het oude prototype	Functie Vision Mixer SDK
1	1, 10
2	1
3	1, 9
4	Geregeld binnen applicatie.
5	11, 12
6	13
7	6
8	7
9	Geregeld binnen applicatie.
10	Geregeld binnen applicatie
11	1, 9
12	1, 9
13	6, 7

Figuur 4.3.1

- Real-time mixen van audio/video van live bronnen, streams en bestanden
- Transities
- Scenes, het opslaan van scenes en het vloeiend overgaan van scenes
- Ingebouwde audio/video synchronisatie
- Ondersteuning voor verschillende input formaten
- Windows media streaming
- Opslaan naar avi of wmv bestanden
- RTSP, MMS & http Windows media stream output

Bovenstaande punten moeten sowieso ondersteund worden door de library die gekozen moet worden om de Vision Mixer SDK library te vervangen.

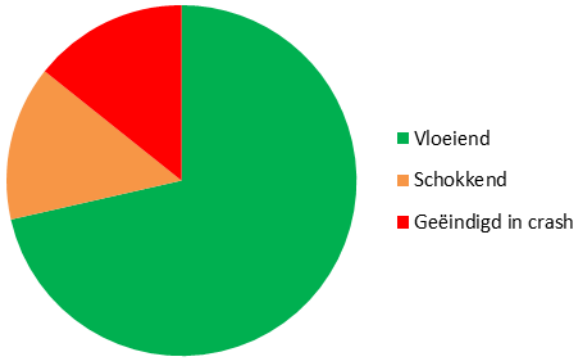
Gebruik en bevindingen

De grootste reden dat het oude prototype niet gebruikt wordt, is het feit dat deze niet stabiel is. Om deze reden is het onderzoek vervolgd met het kijken onder welke omstandigheden de software niet goed meer functioneert. De reden hiervoor is om te leren waar grote gevaren zitten als er gewerkt wordt met digitale video, en om te kijken hoe deze ontweken kunnen worden bij het ontwerpen van het systeem.

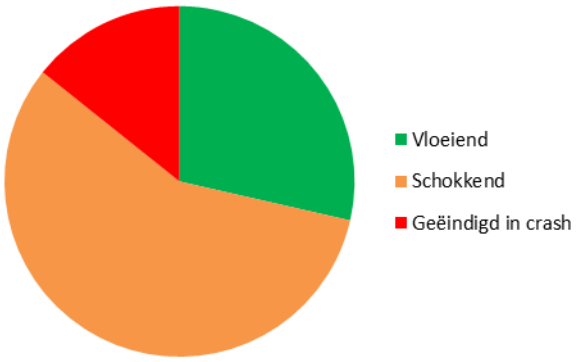
Stabiliteits Metingen

Het onderzoek is uitgevoerd door een aantal metingen te verrichten met verschillende type video input, en verschillende handelingen. De metingen zijn op te delen in twee scenario's. Het eerste scenario past alleen transities en scene veranderingen toe, het tweede scenario past hiernaast ook encodering toe.

Voor beide scenarios zijn er 7 metingen gedaan. In de tabellen in figuur 4.3.2 en figuur 4.3.3 is te zien hoe veel van de metingen goed verliepen (groen), bij hoeveel metingen de applicatie begon te schokken (oranje) en bij hoeveel metingen de applicatie helemaal vastliep/crashte (rood). In bijlage 2 is een overzichtstabel met de metingen te zien. Hier staan ook de (technische)details van de metingen uitgelegd.



Figuur 4.3.2 - Resultaten van metingen zonder encoding

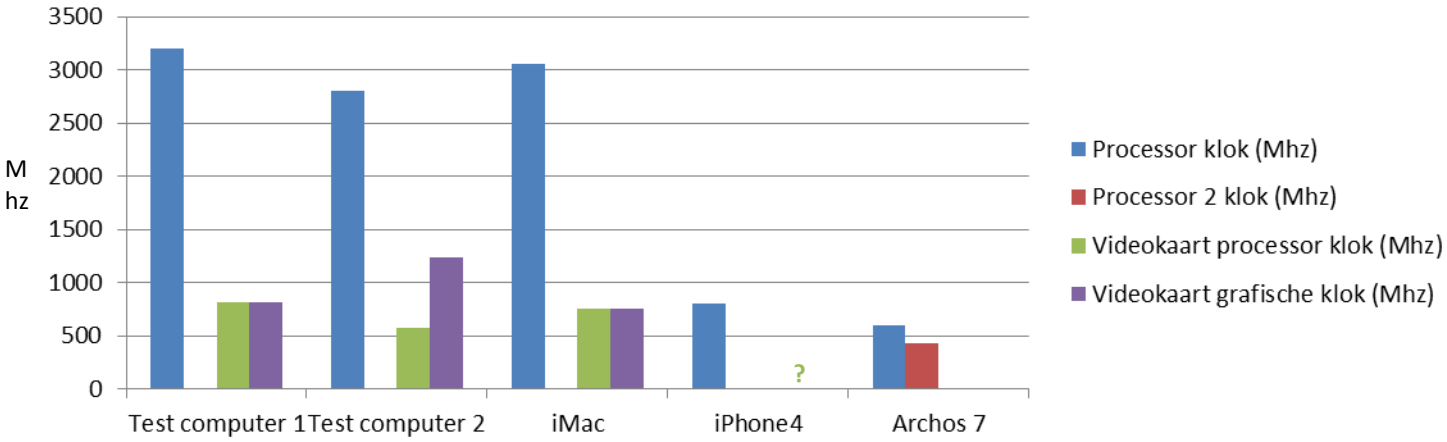


Figuur 4.3.3 - Resultaten van metingen met encoding

Wel viel op dat de programma's stabieler draaiden naarmate de prioriteit van het programma omhoog gezet werd, of het programma getest werd op een snellere computer. Hieruit kon de conclusie getrokken worden dat het aan de snelheid van de computer lag dat het programma niet stabiel draaide. Hoewel de programma's stabieler gingen draaien op een snellere computer, was dit bij lange na nog niet stabiel genoeg.

Hardware

Uit de stabiliteits metingen die gedaan waren kon de conclusie getrokken worden dat de testcomputers simpelweg niet genoeg rekencapaciteit hadden om de bewerkingen snel genoeg (real time) uit te voeren. Als dit al het geval was, was het meteen de vraag of de hardware[1] die gewenst was voor het nieuwe systeem wel de rekencapaciteit zou bieden om de bewerkingen goed uit te voeren. In figuur 4.3.3 zijn een aantal apparaten met de CPU (Central Processing Unit) en GPU (Graphics Processing Unit)die ze bevatten op een rijtje gezet[2].



Figuur 4.3.3 - CPU en GPU van verschillende apparaten op een rij

Uit de grafiek blijkt dat de computers waarop de bestaande software getest was, van ongeveer hetzelfde niveau, of meer rekencapaciteit in zich hadden. Dit betekende eigenlijk dat het een slecht idee was om het systeem, met de eisen die eraan gesteld werden, te programmeren op de apparaten.

Er waren nu twee opties. De eerste optie was om de opdracht zo aan te passen dat de rekenkracht die nodig was om deze uit te voeren, verminderd zou worden. Dit zou als gevolg hebben dat er bepaalde functionaliteit niet zou kunnen worden uitgewerkt. De tweede optie was om de rekenkracht te distribueren.

[1] Zie hoofdstuk 3.2 - Doelstelling & hoofdstuk 3.3 - De opdracht
[2] Zie bijlage 3 - Hardware specificaties

Keuze

Samen met de klant is uiteindelijk besloten om voor de laatste keuze te gaan. Er zou een systeem moeten komen om de berekeningen op uit te voeren. Vervolgens moet er dan met (mobiele) apparaten op dit systeem ingelogd kunnen worden om dit aan te sturen. Op deze manier zou het lijken dat het op de apparaten uitgevoerd werd.

Conclusies

- Het oude prototype is gemakkelijk in gebruik door een goede en heldere user interface.
- Het oude prototype is niet stabiel.
- Het oude prototype is te zwaar voor een standaard pc en heeft krachtigere hardware nodig om goed te draaien.

Keuzes

Uit het onderzoek wat gedaan is naar het oude prototype zijn er een aantal keuzes genomen in overleg met de klant.

- In plaats van het lokaal uitvoeren van AV-mixing wordt dit op afstand gedaan door een server.
- De server waar de AV mixing op wordt uitgevoerd kan bestaan uit meerdere computers die samen werken.
- Eén of meerdere clients kunnen inloggen op de server om deze te bedienen.

4.4 Communicatie

Zoals uit hoofdstuk 4.3 bleek was er voor gekozen om bij het uitwerken van het systeem gebruik te maken van meerdere computers die door middel van een netwerk met elkaar verbonden zijn. Bij het maken van zo'n systeem moet er rekening worden gehouden met het feit dat een netwerk bepaalde imitaties heeft. Om deze reden is er een onderzoek gedaan om er achter te komen welke beperkingen zich kunnen voordoen bij het uitvoeren van dit project.

Server-client communicatie

Bij de requirements[1] staat al gesteld dat er bij de communicatie tussen de client en de server gebruik gemaakt moet worden van webservices. Dit wordt gedaan middels SOAP (simple object access protocol) berichten. Dit is prima voor het gewenste systeem, omdat er gemakkelijk gepraat kan worden met de server door gebruik te maken van web-methodes.

Server

Omdat er veel data tussen de computers aan de server-kant gaat, was het verstandig om te kijken of dit wel mogelijk is. En als dit niet mogelijk is, hoe dit op te lossen is.

TCP

Voor het communiceren van de grote datapakketten is gekozen het TCP (transmission control protocol) te gebruiken. Dit omdat dit de algemeen geaccepteerde standaard is, en ook het versturen van grote hoeveelheden data ondersteunt.

Mocht dit uiteindelijk te langzaam blijken, dan moet er worden gekeken naar het UDP protocol. De vraag hier is wel dat er een risico is dat ge-encodeerde streams corrupt aankomen. In het geval dat UDP gebruikt moet gaan worden zou hiervoor moeten worden getest.

Service Oriënted Architecture

In overleg met de klant is besloten om het framework als service aan te bieden. Hiervoor wordt SOA (Service Oriënted Architecture) gebruikt. Dit houdt in dat data die clients met de server willen communiceren aan een bepaald contract moeten voldoen.

Om dit toe te passen is in overleg met de klant besloten om WCF (Windows Communication Foundation) te gebruiken. De reden hiervoor is dat dit een Framework is wat hier goed kan worden toegepast. Daarnaast wil de technisch begeleider dit framework graag verder gebruiken wanneer de afstudeerder het project overdraagt.

Een bijkomend voordeel van WCF is dat ook beveiliging als extra laag in dit framework zit.

[1] Zie bijlage 4 - Requirements

Opties voor data overdracht

Als de datastroom van AV beelden te groot is om real time te kunnen verzenden zijn er een aantal mogelijkheden om dit op te lossen. Deze mogelijkheden zijn hieronder toegelicht.

Downgrading

De eerste mogelijkheid is om bij het decoden van de video (audio kan ook, maar is relatief een kleine hoeveelheid data vergeleken met video) de data in een compacter formaat op te slaan. Ook kan er voor gekozen worden om de resolutie van het bronbestand aan te passen.

Encoding/Decoding

Een tweede mogelijkheid is om de stream voor het verzenden telkens te encoderen, en na het ontvangen te decoderen. Hierbij wordt de grootte van de datastroom ingeruild voor rekentijd. Of dit een goede methode is, zal van de situatie af hangen. Dit hangt af van de codec, de computer en het netwerk. Het is wel een goede optie om te onderzoeken mocht er echt een probleem ontstaan bij het verzenden van data.

Non-realtime

De laatste optie zou zijn om te accepteren dat het netwerk de load niet real-time kan verwerken. Voor het genereren van bestanden zou dit geen probleem zijn. Maar voor live streams is dit geen optie.

Hoeveelheid data

Iedere netwerkverbinding heeft een beperkte hoeveelheid dataverkeer. Om uit te rekenen wat de mogelijkheden zijn betreft het verzenden van digitale video, moet uitgerekend worden hoeveel data er in een video gaat. Dit hangt echter af van een aantal factoren.

1. Resolutie

   De resolutie van de video is het aantal pixels in één videoframe. Zo heeft een frame met een resolutie van 640*480 in de breedte 640 pixels en in de hoogte 480 pixels.

2. Framerate

   De framerate van een video is hoe veel frames er per seconde in de video zit. Standaard is dit 24p. Er zijn echter twee manieren om dit aan te geven.

   Progressive scan

   De p achter sommige frame rate standaarden staat voor progressive scan. Dit houdt in dat bij het weergeven van de video de frames in hun geheel worden ververst.

   Interlaced

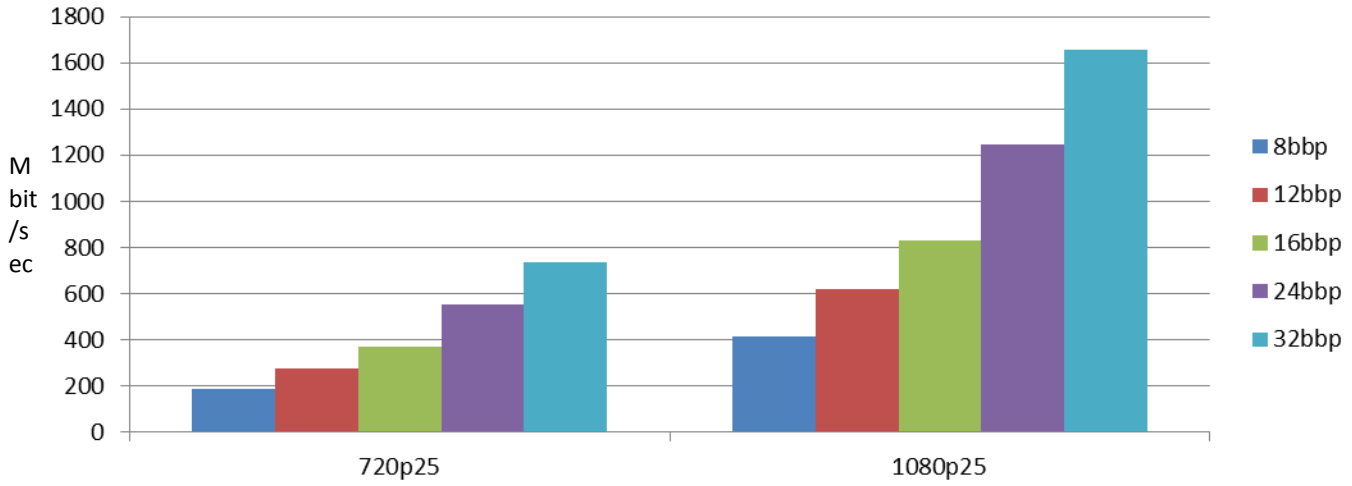
   De i achter sommige frame rate standaarden staat voor interlaced. Dit houdt in dat bij het weergeven van de video frames niet het hele frame wordt ververst. Eerst worden de even lijnen ververst, en vervolgens de oneven lijnen. Er is dus nooit een heel frame tegelijk zichtbaar, maar steeds twee halve frames. Dit gebeurt echter zo snel dat een mens het ervaart als gewoon beeld.

   Resolutie en framerate worden vaak samen afgekort. Hierbij word de hoogte van de resolutie weergegeven in combinatie met de framerate. Een interlaced video met een resolutie van 1920*1080 met een frame rate van 60i wordt geschreven als 1080i60.

3. Bits per pixel

   Bij digitale video verschilt het ook hoeveel data er gebruikt wordt voor het opslaan van één pixel. Dit ligt meestal van 1 tot 32 bit.

Deze drie factoren vermenigvuldigd geeft het totaal aantal bits per seconde van een bepaalde video. In de grafiek in figuur 4.4.1 is een grafische weergave te zien van de formaten die bij Memoria Media het meeste zullen voorkomen. In bijlage 5 is meer informatie te vinden over AV data en is een grafiek opgenomen waar meer data formaten tegen elkaar zijn uitgezet.



Figuur 4.4.1 - Grafische weergave van data verkeer tegen data opslag weergegeven voor de gangbare formaten 720p25 en 1080p25. Links is weergegeven hoeveel Mbit het per seconde zou kosten om de data over te dragen.

Uit de resultaten die te zien zijn in figuur 4.4.1 is samen met de klant overlegd hoe er verder gehandeld moest worden. De klant heeft toen besloten om hier tijdens het maken van dit project geen extra tijd aan te besteden, en dit voor een eventuele uitbreiding open te laten[1].

Verder moeten makers van filters en de beheerder van het netwerk vooral rekening houden met de capaciteiten van het netwerk. Omdat deze factoren sterk van elkaar afhankelijk zijn kan dit beter per situatie op elkaar af worden gestemd.

Conclusies

- Commando communicatie over het netwerk zal gedaan worden met het soap protocol.
- AV data communicatie over het netwerk zal gedaan worden met het TCP protocol.
- Indien het TCP protocol toch niet voldoet aan de eisen zal het UDP protocol worden toegepast.
- Er hoeft bij het maken van het framework geen rekening gehouden te worden met de netwerksnelheid. Dit moet worden gedaan in overleg tussen de beheerder van het netwerk en de maker van de filters.

[1] Zie bijlage 6 - Workload

4.5 Alternatief Medialook library

Omdat de functionaliteit van het oude prototype beschikbaar moet zijn in het product zonder gebruik te maken van de Medialooks library die in het oorspronkelijke product gebruikt werd, moest er een alternatief worden gevonden voor de Medialooks library. Het onderzoek dat in dit hoofdstuk wordt beschreven geeft de mogelijke alternatieven voor deze library, wat de voor- en nadelen zijn van de verschillende oplossingen, en welke oplossing het meest geschikt is voor het systeem.

Onderzoek

De library die de Medialooks library zou moeten gaan vervangen dient de functionaliteit te bieden die het oude prototype[1] biedt.

Met dit in het achterhoofd zijn tijdens het onderzoek de volgende kandidaten gevonden om de Medialooks library te vervangen. Deze producten zijn gevonden door te zoeken op het internet.

- DirectShow
- VideoLan
- G-Streamer
- Media Foundation
- LeadTools Multimedia SDK
- Visioforge SDK's

Al deze kandidaten bieden de functionaliteit aan die door het oude prototype ook aangeboden wordt, en kunnen gebruikt worden om de Medialooks library te vervangen.

Nu moest er één van deze producten gekozen worden om te gebruiken in het eindproduct. Om dit goed te kunnen doen waren er door de afstudeerder en de klant een aantal kenmerken opgesteld die van belang zijn voor het eindproduct. Er is hierbij ook rekening gehouden met de toekomst. Er moest rekening gehouden worden met de licentie, als Memoria Media het product wil gaan verkopen. In figuur 4.5.1 is een overzicht te zien van de producten, samen met de kenmerken.

Product	Draait op Windows 7	Draait op Windows server	Gratis	Verkoopbaar	Support***	Open-source
DirectShow	Ja	Ja	Ja	Ja	Uitstekend	Nee
VideoLan	Ja	Ja	Ja	Nee	Goed	Ja
G-Streamer	Ja*	Ja*	Ja	Ja	Matig voor Windows	Ja
Media Foundation	Ja	Ja	Ja	Ja	Goed	Nee
LeadTools Multimedia SDK	Ja	Ja	Nee	Ja**	Beschikbaar, kwaliteit onbekend	Nee
Visio forge SDK's	Ja	Ja	Nee	Ja		Nee

Figuur 4.5.1 - De verschillende producten en de kenmerken

*G-Streamer heeft een Windows versie (OSSBuild)

**Verkoopbaar met speciale licentie waar weer voor betaald moet worden

***Kwaliteit van community

[1] Zie hoofdstuk 4.4 - Het oude prototype, Functionaliteit

Omdat het belangrijk was voor de klant dat het product gratis is, vallen de LeadTools en de Visio forge SDK's af.

Er bleven nu nog vier mogelijke kandidaten over die bruikbaar zijn voor het eindproduct. Nu moest er gekeken worden welk van deze producten het beste is voor de toepassing in het eindproduct. Na de vorige kenmerken was het belangrijk dat het product verkoopbaar is. Dit geeft de klant in de toekomst meer flexibiliteit, en voorkomt vervelende verassingen. Hierdoor valt ook VideoLan af, en blijven G-streamer, Media Foundation en DirectShow over.

Deze drie producten leken prima geschikt voor het project. G-streamer bleek echter een minder goede support te hebben voor Windows developers. Er is een forum, maar hier is nog vrij weinig activiteit. Dit komt omdat G-streamer nog relatief nieuw is, en de OSS-build, die voor windows is, nog steeds een work in progress is. Daarnaast zijn DirectShow en Media Foundation van Microsoft en speciaal geschreven voor Windows. Dit maakt het aannemelijker dat deze producten stabieler en beter getest zijn voor dit OS dan een nieuw framework als de OSS-build van G-streamer. Dit maakt G-streamer een minder verstandige keuze in vergelijking met DirectShow en Media Foundation.

Uiteindelijk is er voor DirectShow gekozen boven Media Foundation. De reden hiervoor is dat Media Foundation in verhouding tot DirectShow een stuk jonger is, en toch minder ervaren gebruikers kent of het development forum. Een laatste argument voor het kiezen van DirectShow is dat de library van Medialooks die nu gebruikt wordt, ook werkt met DirectShow. Dit maakt de keuze veiliger omdat het zeker is dat dezelfde resultaten te behalen zijn.

Media Foundation, de opvolger van DirectShow, was niet gekozen omdat het niet op alle gewenste OS-en zou kunnen draaien. Wel is het een goed alternatief mogen deze eisen er niet zijn.

Conclusies

- DirectShow is de beste keuze om de Medialooks library mee te vervangen.
- Goede alternatieven zouden G-streamer en Media Foundation zijn.

4.6 DirectShow

Zoals in het vorige hoofdstuk wordt toegelicht, was er besloten om DirectShow te gebruiken bij het project. Om deze reden is er een onderzoek gedaan naar de opbouw van DirectShow om een inzicht te krijgen hoe DirectShow gebruikt moet worden, en om een inzicht te krijgen in een goede manier om met audio en video om te gaan. Dit zou ook nuttig zijn bij het ontwerpen van de rest van het systeem.

Dit onderzoek is gebaseerd op de documentatie die te vinden is op de website van Microsoft. De precieze bronnen zijn te vinden in de Literatuurlijst.

Systeem overzicht

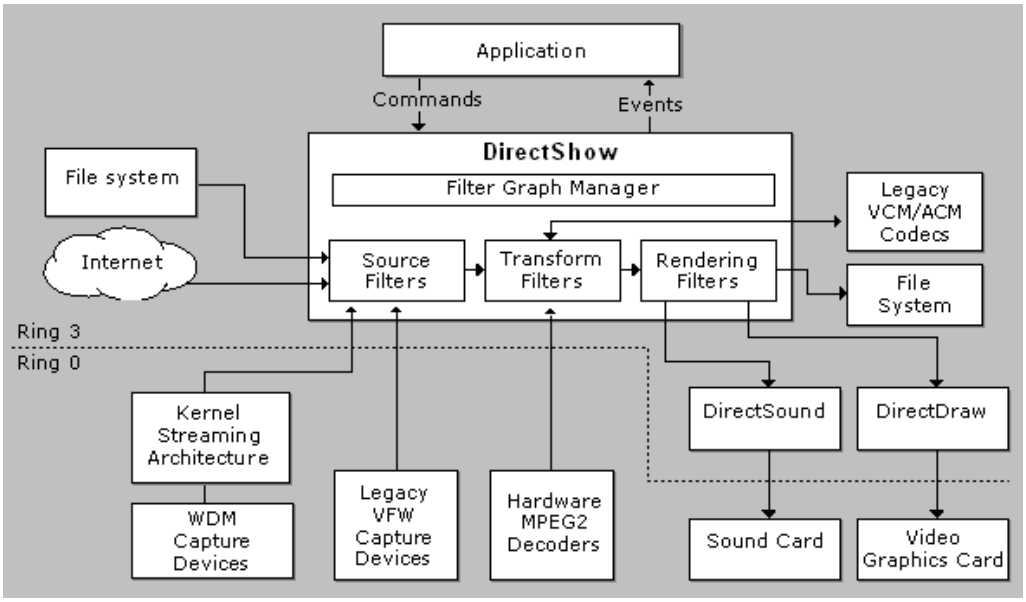
Wanneer er met video en audio gewerkt wordt, komen de volgende uitdagingen naar boven:

- Multimedia heeft vaak grote hoeveelheden data, die snel verwerkt moeten worden.
- Audio en video moeten worden gesynchroniseerd zodat starten, stoppen en afspelen hiervan tegelijk gebeurt.
- Data kan van veel verschillende bronnen komen zoals lokale bestanden, netwerken en camera's.
- Data komt in verschillende formaten, zoals AVI, MPEG etc.
- De programmeur weet meestal niet wat er voor hardware in de computer zit waar het eindproduct op gaat draaien.

DirectShow is ontworpen om om te kunnen gaan met al deze uitdagingen. Het hoofddoel van DirectShow is om het maken van een multimedia applicatie gemakkelijker te maken. Dit doel wordt bereikt door complexe logica zoals data transport, hardware moeilijkheden en synchronisatie te isoleren en afzonderlijk uit te voeren.

DirectShow maakt zo veel mogelijk gebruik van DirectDraw en DirectSound. Deze technologieën renderen de data op de video- en geluidskaart van de computer. Voor synchronisatie maakt DirectShow gebruik van timestamps die in de datapakketen worden gestopt. Om verschillende formaten te ondersteunen maakt DirectShow gebruik van een combinatie van één of meerdere verschillende software componenten die filters genoemd worden.

In figuur 4.6.1 is een overzicht te zien van hoe DirectShow is opgebouwd.



Figuur 4.6.1 - Opbouw DirectShow

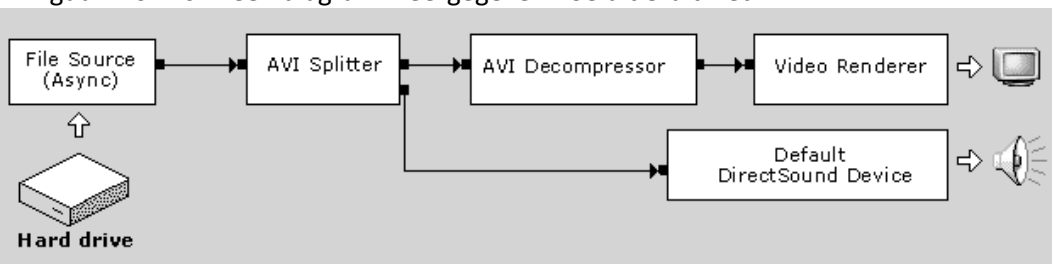
DirectShow filter graph

DirectShow maakt gebruik van een modulaire architectuur, waarbij elke stap van data verwerking wordt uitgevoerd door een filter. DirectShow bevat zelf een verzameling van standaard filters, maar ontwikkelaars kunnen ook zelf filters schrijven voor het DirectShow framework.

Als voorbeeld is hieronder weergegeven hoe een AVI (audio video Interleave) bestand vanaf de harde schijf wordt afgespeeld:

- Lees de ruwe data van een bestand als byte stream (File Source filter).
- Kijk in de avi header en parse de data stream naar losse frames en audio monsters.
- Decodeer de video frames (afhankelijk van het compositie patroon moet er een decoder gekozen worden).
- Teken de video frames (Video Renderer filter).
- Verzendt de audio monsters naar de geluidskaart (Default DirectShound Device filter).

In Figuur 4.6.2 is in een diagram weergegeven hoe dit eruit ziet.



Figuur 4.6.2 - Filter graph

Zoals het diagram laat zien zijn de verschillende filters verbonden met andere filters. Tussen deze verbindingen gaat data van de ene filter naar de andere filter. Zo'n verzameling van verbonden filters wordt een DirectShow filter graph genoemd. Zo'n filter graph voldoet aan pipeline architectuur. Hierover wordt meer verteld in het volgende hoofdstuk.

DirectShow filters kunnen worden ingedeeld in verschillende categoriën:

- Een ***source filter*** geeft data aan de DirectShow filter graph. Deze data kan van een bestand, een netwerk, een camera of een andere bron komen.
- Een ***transformatie filter*** neemt een input stream, behandeld de data, en maakt een output stream. Voorbeelden van transformatie filters zijn encoders en decoders.
- ***Render filters*** zijn te vinden aan het einde van een DirectShow filter graph. Deze krijgen data en presenteren het aan de gebruiker. Een video render filter tekent bijvoorbeeld video frames op het scherm, een audio render filter stuurt geluid naar de geluidskaart en een bestand-schrijf render filter schrijft data naar een bestand.
- Een ***splitter filter*** split de input stream op in twee of meerdere outputs. Vaak wordt de data hierbij ook geparsted. Een voorbeeld hiervan is de AVI splitter, die een data stream opdeelt naar losse video en audio data.
- Een ***mux filter*** neemt twee of meerdere inputs, en maakt hier één output van. Deze filter doet dus het tegenover gestelde van een splitter filter.

Een filter kan tot meer dan één type tegelijk behoren. Het ASF Reader filter dient zo bijvoorbeeld zowel als source filter en als splitter filter.

4.7 Pipelines

Na het vorige onderzoek betreffende DirectShow was er veel kennis opgedaan over het streamen van digitale video onder Windows. Het onderdeel dat het meest opviel was de filtergraph, en hoe deze heel erg overeen lijkt te komen met het pipeline architectuur patroon.

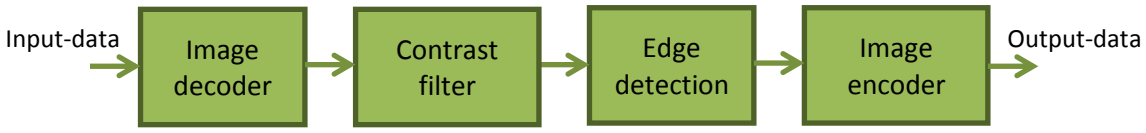
Om deze reden is er ook een onderzoek gestart om dit patroon verder te onderzoeken. Het hele onderzoek is te vinden in bijlage 7. In dit hoofdstuk wordt een korte samenvatting gegeven van het pipeline architectuur patroon.

Dit onderzoek is gebaseerd op informatie van het internet en een presentatie die Philips Healthcare gegeven had aan de afstudeerder.

Nodes

Een pipeline wordt toegepast als er verschillende acties, of filters achter elkaar moeten plaatsvinden. Het is een rij logische blokken. Deze blokken worden nodes genoemd.

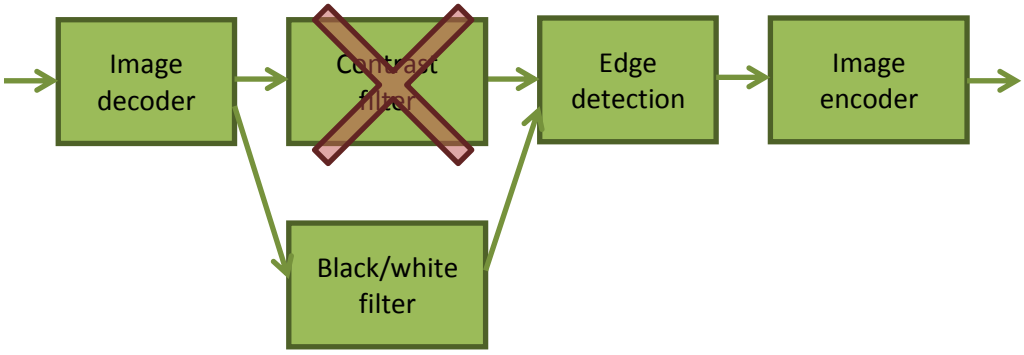
In figuur 4.7.1 is een voorbeeld te zien van een image-processing pipeline, de groene pijltjes geven de flow van data weer. De data die in een node gaat is input-data, de data die uit een node komt, output data. Elke node doet zijn eigen werk, niet wetende wat de rest van de pipeline doet, en het resultaat wordt doorgegeven aan de volgende node. Het werk dat zo'n node precies doet kan van alles zijn; van



Figuur 4.7.1 - Een image-processing pipeline

een simpel optel sommetje tot complexe beeldbewerking.

Eén van de grote voordelen van een pipeline is dat er heel gemakkelijk nodes kunnen worden veranderd, zonder dat dit andere nodes beïnvloedt. In figuur 4.7.2 is te zien hoe de contrast filter uit het voorbeeld hierboven vervangen wordt door een zwart/wit filter. Het eind resultaat van de pipeline zal hierdoor veranderen, maar de andere nodes worden hierdoor niet beïnvloed, en kunnen gewoon hun eigen werk voortzetten.



Figuur 4.7.2 - Nodes kunnen gemakkelijk vervangen worden binnen een pipeline. De rest wordt niet beïnvloed

Latency

De latency (of processing time) van een (simpele) pipeline is de tijd die data erover doet om vanaf het moment dat het in de pipeline gestopt wordt, naar de andere kant te gaan.

De tijd die data er over doet om van het begin naar het einde van de pipeline te gaan, is uit te rekenen door de tijd van elke node bij elkaar op te tellen. In figuur 4.7.5 is de tijd die data erover doet om door de pipeline te gaan dus 1 + 1 + 1 = 3 seconden.

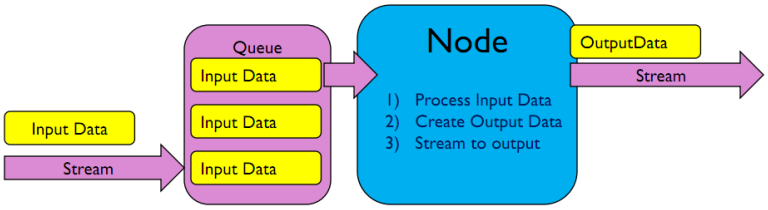
Figuur 4.7.5 - Drie nodes die elk één seconde over een taak doen

Een belangrijk voordeel van een pipeline is, dat een pipeline verschillende stukken data tegelijk kan verwerken. Als de eerste node van de pipeline klaar is met de data, en zijn data doorgeeft aan de volgende node, kan er wel al nieuwe data de pipeline in. Dit betekent dat bovenstaande pipeline 3 seconden over een stuk data doet, maar wel elke seconde nieuwe data kan accepteren, en leveren. De latency van de pipeline is 3 seconde en de throughput is 1 seconde.

Hoe kleiner nodes gemaakt worden, hoe meer data er dus tegelijkertijd berekend kan worden (hogere throughput), en hoe sneller de dataflow zal worden. Ter vergelijking: nu kunnen er in 5 seconden 3 stukken data helemaal door de pipeline gaan, maar als de nodes zouden worden samengevoegd, zouden dezelfde 3 stukken data er 9 seconden over doen.

Queue

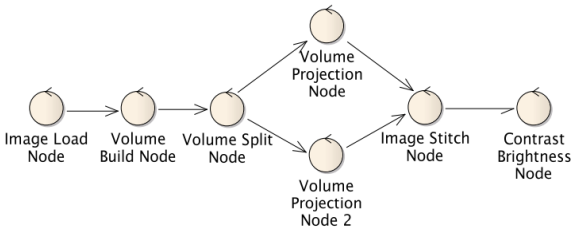
Het kan voorkomen dat er schommelingen omstaan in de throughput. Zo kunnen er sommige momenten meer data aangeboden worden, of kunnen nodes tijdelijk trager worden. De pipeline kan dan verstoppen, of wordt niet optimaal gebruikt.



Figuur 4.7.3 – Een node met een queue

Een oplossing hiervoor is het gebruik maken van een queue (rij). Als een node nog bezig is met zijn taak, maar er komt wel al nieuwe data, dan wordt deze in de rij geplaatst. Wanneer de node dan klaar is met zijn taak, pakt hij de volgende data die in de rij staat, en gaat daar aan werken. In figuur 4.7.3 is weergegeven hoe dit eruit ziet.

De Input-data gaat de queue in, en de node pakt een nieuw stuk data uit de queue wanneer deze klaar is met de huidige taak. Als de node dan een moment meer data krijgt of wat langzamer werkt dan gemiddeld loopt de queue vol. Komt er weer wat minder data binnen of werkt de node wat sneller, dan kan deze vorige taken afhandelen en is deze toch bezig.



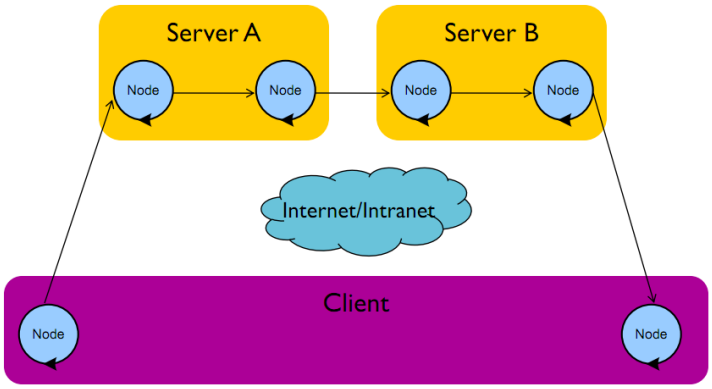
Figuur 4.7.4 – De pipeline kan opsplitsen om bottlenecks te voorkomen of een complexe berekening op te delen.

Belangrijk voor optimale performance is het dat de nodes ieder alsnog het beste even lang over het verwerken van data kunnen doen. Als er een node aan het einde twee maal zo lang doet over zijn taak dan de voorliggende nodes, zal de queue vol blijven lopen. Er zijn wat standaard oplossingen die kunnen worden toegepast in een zo'n geval. Zo kan een node die te lang over zijn taak doet bijvoorbeeld worden opgedeeld in twee nodes die allebei een deel doet van de oorspronkelijke taak. Of de data kan worden opgedeeld zodat er twee verschillende nodes met dezelfde logica aan hetzelfde probleem kunnen werken. Deze laatste oplossing wordt in figuur 4.7.4 afgebeeld.

Pipeline en distributed computing

Het principe van het gebruiken van distributed computing combineert heel natuurlijk met een pipeline. Aangezien nodes heel gemakkelijk uit elkaar kunnen worden gehaald, en zelf niet bewust hoeven te zijn van hun bijdrage in het grote geheel, is het ook erg gemakkelijk om een node bijvoorbeeld op een andere computer te laten draaien. Zo kan een pipeline gemakkelijk worden verdeeld over meerdere computers. In figuur 4.7.6 staat afgebeeld hoe dit werkt.

De data gaat oorspronkelijk bij de client de pipeline in. Maar zodra deze uit de eerste node komt, wordt deze naar server A gestuurd. Deze zal, samen met server B, wat andere berekeningen uitvoeren. Uiteindelijk wordt het resultaat weer naar de client gestuurd, die wat met de data kan doen. De client hoeft op deze manier niet veel rekenkracht te hebben.



Figuur 4.7.6 - Een pipeline gedistribueerd worden

Een bijkomend voordeel is dat er op deze manier ook op een slimme manier omgegaan kan worden met de hardware die gebruikt wordt voor de pipeline. Omdat specifieke nodes vaak één taak hebben, kan er voor bepaalde nodes hardware speciaal worden aangeschaft voor de taak die ze uitvoeren.

Conclusies

Ter conclusie van het pipeline patroon nog even de behandelde voor- en nadelen van het gebruik van een pipeline op een rijtje.

Voordelen

- Nodes zijn gemakkelijk te verwisselen.
- Een pipeline is gemakkelijk te verdelen over meerdere processen/computers.
- Een pipeline kan meerdere taken tegelijk doen (wel in verschillende nodes).

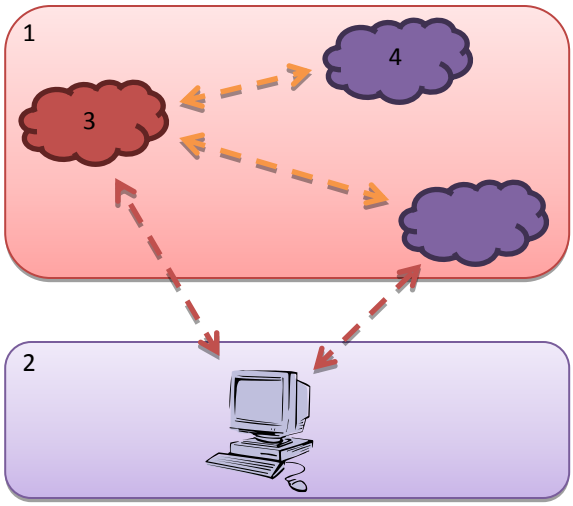
Nadelen

- Eén node kan de hele pipeline blokkeren.

4.8 Cloud computing

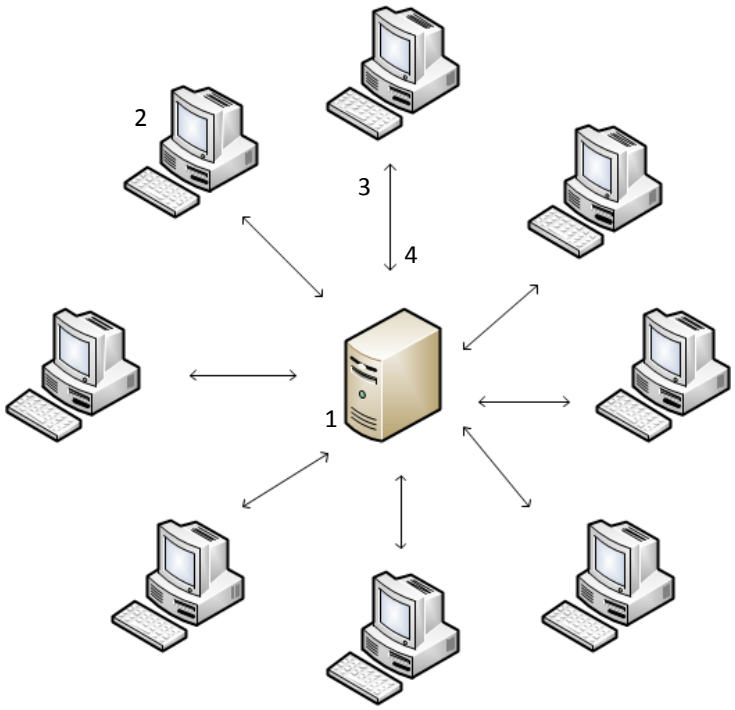
Omdat er uit het onderzoek voorgaande van het oude prototype[1] gebleken is dat één standaard PC niet genoeg rekenkracht heeft om de video filter taken uit te voeren, was er voor gekozen om dit op meerdere computers uit te voeren. Om deze reden is er een onderzoek gestart naar cloud computing.

Dit hoofdstuk gaat over de cloud van de server kant van het project. Er wordt in dit hoofdstuk over masters en slaves gesproken. Deze termen verwijzen naar instanties binnen een cloud. In figuur 4.8.1 is dit grafisch weergegeven.



Figuur 4.8.1 - Overzicht van het systeem binnen en buiten de cloud.
1. Server. Hier draait de cloud.
2. Client. Bijvoorbeeld een iPhone.
3. Master. Leider van de cloud.
4. Slave. Volger binnen de cloud.

Er zijn vele manieren om een cloud in te construeren. Om het verhaal duidelijk te houden beginnen we met een standaard voorbeeld, en gaan we stap voor stap een cloud architectuur opbouwen dat het beste gebruikt kan worden voor dit project.



Figuur 4.8.2 - Een voorbeeld van een simpel cloud computing systeem
1. Master
2. Slave
3. Verzoek voor berekening gaat naar een slave.
4. Vervolgens wordt het resultaat terug gestuurd naar de master.

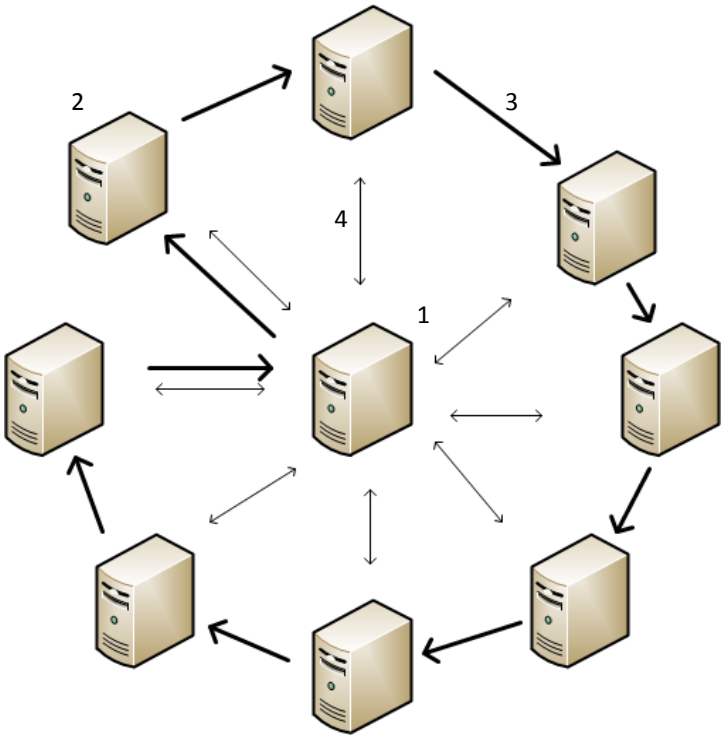
Als voorbeeld gaan we uit van een centrale master die berekeningen laat uitvoeren door een grote hoeveelheid computers om zo snel complexe berekeningen te kunnen doen. Een dergelijk architectuur is te zien in figuur 4.8.2.

Dit is een goed begin van een cloud architectuur. Echter moet het server systeem om kunnen gaan met enorme hoeveelheden data. Het is een probleem als al deze data (frames) elke keer door de master heen zouden moeten. De master heeft zelf ook maar een beperkte down- en upload. Dit is niet praktisch.

[1] Zie hoofdstuk 4.4 - Het oude prototype, sub-hoofdstuk Hardware

Pipeline

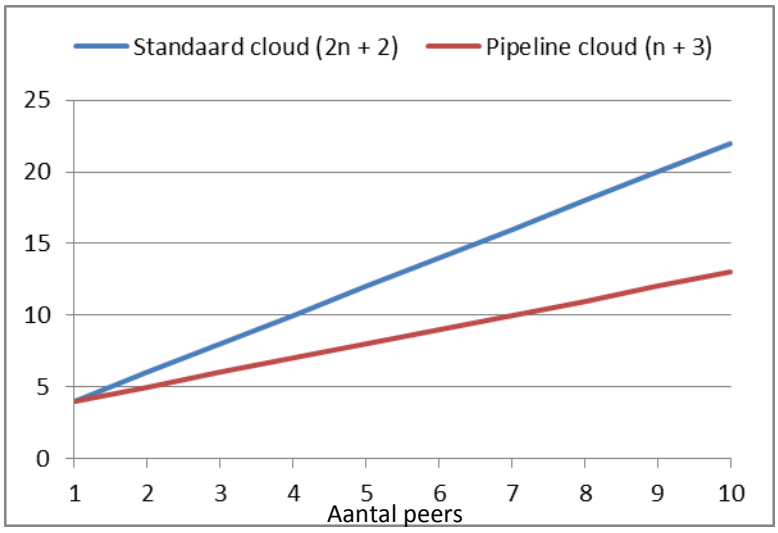
Hier viel ook het pipeline patroon toe te passen. Het voordeel van dit patroon is dat taken zijn opgedeeld in nodes. Elke node doet dezelfde taak telkens weer, en er kan gezegd worden dat deze in de meeste gevallen even lang over hun taak zullen doen. Dit betekent dat het systeem maar één keer hoeft te kijken waar berekeningen het beste kunnen worden uitgevoerd om de snelste cloud te krijgen (behalve als fysieke systemen uitvallen of worden toegevoegd. Hier wordt later op terug gekomen). Een groot voordeel hiervan is dat het systeem weet waar taken worden uitgevoerd. En omdat er een pipeline is gedefinieerd, weet het systeem ook in welke volgorde deze taken moeten worden uitgevoerd. Hier kan voordelig gebruik van worden gemaakt door de systemen nu direct aan elkaar te verbinden.



Figuur 4.8.3 - Het aangepaste cloud systeem
1. Master
2. Slave
3. Video/audio data
4. Overige data (bijvoorbeeld om de cloud in stand te houden)

Het systeem is te zien in figuur 4.8.3.

De dikke lijnen vertegenwoordigen de grote datapakketten zoals videoframes. De kleine pijlen zijn kleine hoeveelheden data om de connectie tussen de master en slaves te regelen. De computers vormen nu dus een pipeline, met de fysieke computers als nodes.



Figuur 4.8.4 - Het aantal data transporten tussen peers voor een standaard cloud en een pipeline cloud

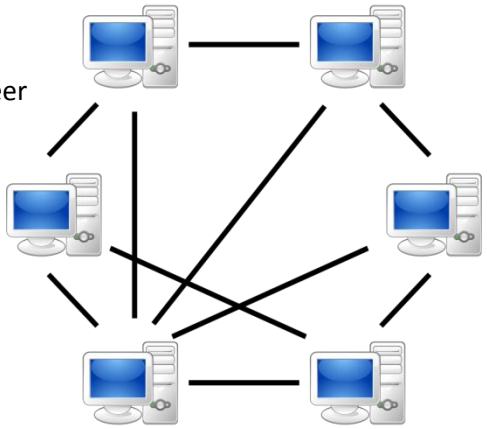
Doordat er nu een pipeline gebruikt wordt, hoeft de data een stuk kortere weg af te leggen. Hoeveel dit scheelt met de standaard cloud implementatie is goed te zien in de grafiek in figuur 4.8.4.

Er doet zich nu echter wel een nieuw probleem voor. Zoals in het onderzoek naar de pipeline architectuur is gebleken, is een zwakte van dit patroon, dat als er een node uitvalt, of op een andere manier zijn werk niet meer doet, het hele systeem niet meer kan werken. Het was echter een eis van de klant dat er een computer uit de cloud getrokken moet kunnen worden en dat het systeem dan nog steeds door blijft werken. Echter is dat nu niet mogelijk. Als er een computer kapot gaat of stroom verliest, dan wordt de pipeline onderbroken, en zal het systeem niet meer werken.

Peer to peer

Om te voorkomen dat het systeem niet meer werkt als er één computer uitvalt of vastloopt, was er besloten om een peer-to-peer netwerk toe te passen.

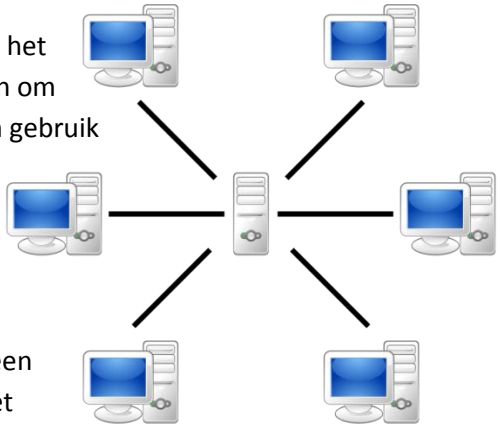
Een peer to peer netwerk is een vorm van distributed computing waarin verschillende peers (peer=gelijke) met elkaar verbonden staan en een netwerk vormen. Hierbij heeft elke peer gelijke rechten. Hoe zo'n netwerk eruit ziet wordt in figuur 4.8.5 weergegeven.



Figuur 4.8.5 - Een puur peer to peer

In zo'n netwerk is niet iedere peer verbonden met iedere andere peer. Als peers elkaar willen vinden dan moet dan soms via andere peers gebeuren.

Er is echter een variant op dit puur peer to peer netwerk. Soms is het gewenst dat er één centrale plek is waar mensen kunnen inloggen om informatie te krijgen over het peer to peer netwerk. Er wordt dan gebruik gemaakt van een master-based peer to peer netwerk. De master houdt dan een lijst bij van alle peers in het netwerk, en registreert ook nieuwe peers. In figuur 4.8.6 is een afbeelding van een dergelijk netwerk weergegeven.



Figuur 4.8.6 - Een master-based peer to peer netwerk.

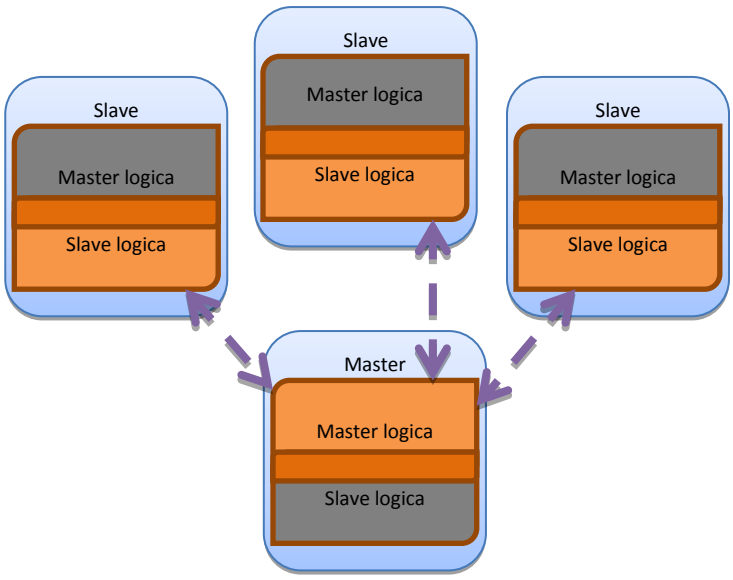
In tegenstelling tot een puur peer to peer netwerk is er hier wel een bottleneck te vinden: als de master uitvalt, werkt het systeem niet meer naar behoren. Wel is het voordeel dat er één toegangspunt is tot het netwerk.

Robust

Om uiteindelijk een robust systeem te maken dat niet stopt met werken als er een computer uitvalt of vastloopt, is bedacht om de twee type peer to peer netwerken met elkaar te combineren. Het eerste type netwerk heeft geen bottlenecks. Aan de andere kant is het handig als ons systeem één master heeft die de slaves in de gaten houdt, wat weer typerend is voor het tweede type peer to peer netwerk.

Wel is er voor gekozen om het systeem simpel te houden. Deze keuze is goed te realiseren omdat ons netwerk relatief klein blijft, wat het mogelijk maakt om één master alle slaves in de gaten te laten houden.

Het resulterende systeem is in principe een master-based peer to peer netwerk. Echter zijn de peers niet compleet afhankelijk van de server. Elke peer bevat de mogelijkheid om zowel master als slave te worden. Als de master dan uitvalt neemt een andere peer zijn taak over. In figuur 4.8.7 is grafisch weergegeven hoe dit eruit ziet.



Figuur 4.8.7 – Peers bevatten zowel master als slave logica.

Hoe dit systeem uiteindelijk geïmplementeerd is, wordt uitgelegd bij het hoofdstuk over het cloud framework[1]. Ook staan hier meer details over in bijlage 8.

[1] Zie hoofdstuk 5.3 – Cloud Framework

Implementatie

Toen besloten was dat er cloud computing geïmplementeerd ging worden, werd het tijd om te kijken of er goede opties zijn om dit te doen. Aan de hand van een onderzoek er een aantal producten gevonden. Deze producten staan in de tabel in figuur 4.8.8.

Product	Gratis?	Draait lokaal	Bottleneck	Robuust (fail tolerant)	Open source?	Inzetbaar voor veel data	Inzetbaar voor veel rekenkracht
Mpapi	Ja	Ja	Ja*	Nee**	Ja	Ja	Ja
Appistry	Nee	Nee	Ja*	Ja	Nee	Hangt van pakket af	Hangt van pakket af
Windows Azure	Nee	Ja	Ja*	Ja	Nee	Hangt van pakket af	Hangt van pakket af
Alchemi	Ja	Ja	Ja*	Nee**	Ja	Ja	Ja
Utilify	Ja	Ja	Ja*	Nee**	Ja	Ja	Ja
Amazon EC2	Nee	Nee	Ja*	Ja	Nee	Hangt van pakket af	Hangt van pakket af

Figuur 4.8.8 - Overzicht van gevonden producten die behulpzaam zijn voor cloud implementatie.
*Bottleneck ligt bij de client van de cloud; als deze uitvalt worden er geen opdrachten meer aan de cloud gegeven, en werkt het systeem niet meer.
**Uit de code van deze producten blijkt dat als een deel van de cloud weigert de opdracht uit te voeren, de master toch op antwoord blijft wachten, en dus vastloopt.

Uit alle producten in de tabel lijken Mpapi, Alchemi en Utilify de beste opties voor dit project. Na wat verder in te gaan op deze producten is de conclusie dat er het beste kan worden uitgegaan van het Mpapi en Utilify framework. Deze keus is gemaakt omdat deze producten compleet open-source zijn en goede voorbeelden bevatten, in tegenstelling tot Alchemi, die geen goede sdk levert. Omdat de wensen voor dit project vrij specifiek zijn moet er wel nog het één en ander aangepast worden, maar Mpapi en Utilify zijn goede voorbeelden/uitgangspunten.

Conclusies

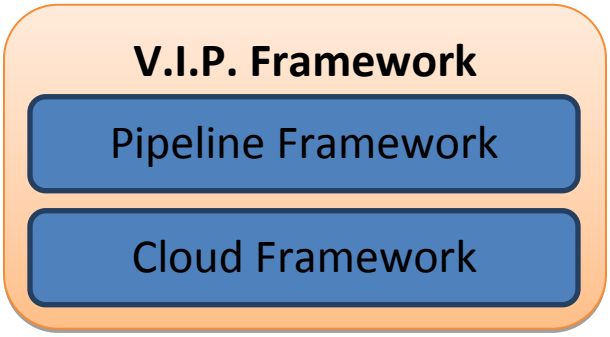
- Voor dit project moet cloud computing worden toegepast.
- Er moet een soort pipeline van computers gemaakt worden binnen de cloud.
- Om het systeem fail-tolerant te maken moet een variatie van peer to peer logica worden toegepast.
- Het framework voor dit project moet nog steeds zelf worden geschreven, maar kan geïnspireerd zijn op het Mpapi en Utilify framework.

5. De uitwerking

5.1 Inleiding

Toen het onderzoek achter de rug was, was het tijd om het systeem te ontwerpen en te implementeren. Voor de implementatie van het systeem is een framework ontworpen. Dit framework is in staat om een cloud te maken, en vervolgens een pipeline op deze cloud te laten draaien. Dit framework is V.I.P. Framework genoemd, en implementeert logica dat zowel aan de server als aan de client kant gebruikt kan worden.

Dit framework is intern weer opgesplitst in twee losse frameworken. Het ene framework zorgt voor de pipeline implementatie en het ander voor de cloud. Hier is voor gekozen om het overzichtelijk en dynamisch te houden. In het vervolg kan het framework gemakkelijk worden uitgebreid en veranderd worden. In figuur 5.1.1 is een grafische weergave te zien van de opbouw van het V.I.P. framework. In bijlage 9 zijn ook wat klasse diagrammen en interfaces toegevoegd.



Figuur 5.1.1 – Het V.I.P. framework bestaat weer uit twee andere frameworks: het pipeline framework en het cloud framework.

In dit hoofdstuk worden beide frameworks los behandeld en wordt vervolgens uitgelegd hoe ze zijn samengevoegd. Vervolgens wordt er ook nog wat verteld over hoe DirectShow geïmplementeerd moet worden. Uiteindelijk wordt er verteld hoe de daadwerkelijke implementatie van het framework verlopen was, en hoe ver de afstudeerder gekomen is met het implementeren van het complete systeem.

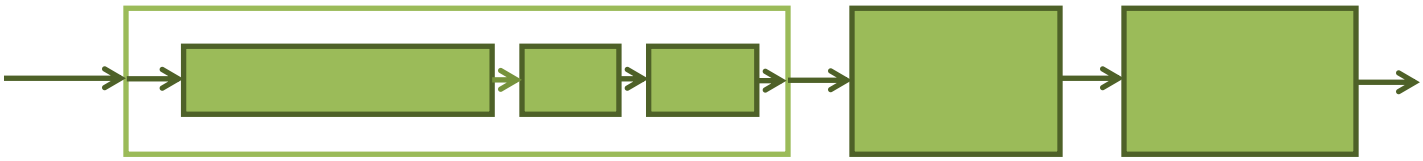
5.2 Pipeline Framework

Allereerst is er een pipeline framework ontwikkeld. De bedoeling van dit framework was om een interface aan te bieden aan de gebruikers van het systeem, en ook om op de verschillende peers in de cloud te kunnen draaien. Hoe dit uiteindelijk in elkaar zit is goed te zien in figuur 4.8.6 in het hoofdstuk over de cloud pipeline[1].

Composite Patroon

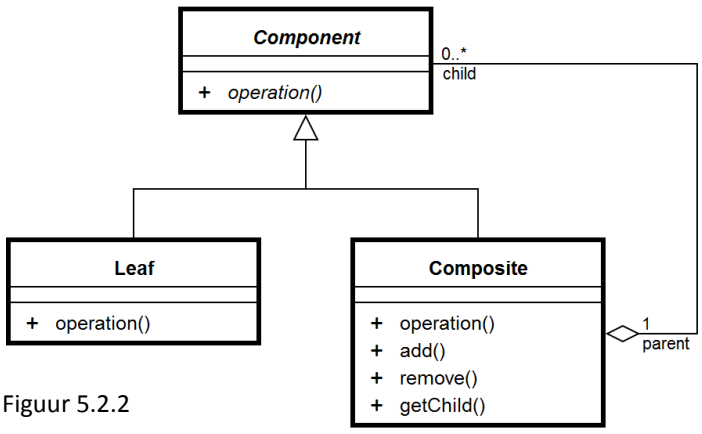
Er is bij het ontwerp voor gekozen om het composite patroon toe te passen bij het maken van het pipeline framework. Dit maakt het framework flexibel en staat gebruikers en programmeurs toe om pipelines te hergebruiken of zelfs als plug-ins aan te bieden.

Het idee van het composite patroon is dat een programma geen verschil hoeft te zien tussen een enkel object, en samengevoegde objecten.



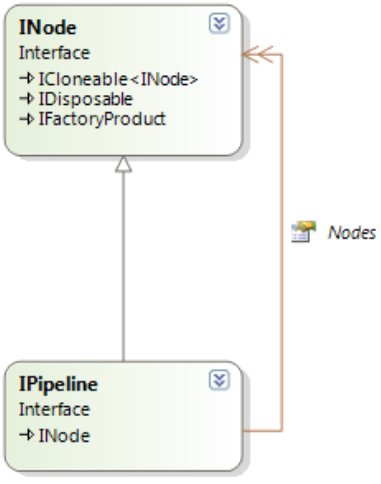
Figuur 5.2.1

Zoals te zien is in figuur 5.2.1, bevat de eerste node in deze pipeline weer een hele pipeline. Het programma maakt dus geen onderscheid tussen één enkele node, en een verzameling van samengevoegde nodes. Een pipeline, dat eigenlijk niet meer is dan een verzameling nodes, wordt op zichzelf weer gezien als een node. In figuur 5.2.2 is dit beschreven als UML diagram.



Figuur 5.2.2

Zoals je hier kunt zien, erft de composite van de interface component, en beheert de composite ook een collectie met componenten. Dit houdt in dat composite naast een collectie van het object leaf (wat ook van component erft), ook andere composite objecten kan beheren. Dit is precies hetzelfde als dat een pipeline behalve een bepaalde type pipeline node, ook een andere pipeline zou kunnen bevatten.



Figuur 5.2.3

Om dit mogelijk te maken, moet de pipeline zelf een instantie zijn van pipeline node. Hier is in het ontwerp ook rekening mee gehouden. In figuur 5.2.3 is een UML diagram te zien van hoe de pipeline in het systeem uiteindelijk erft van een Node.

[1] Zie hoofdstuk 4.8 - Cloud Pipeline

Plugins

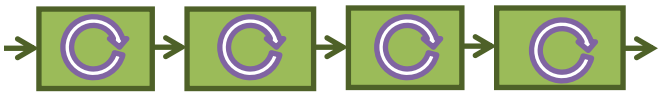
Bij het onderzoek werd duidelijk dat nodes binnen een pipeline gemakkelijk kunnen worden vervangen door andere nodes, en onafhankelijk van elkaar een taak kunnen uitvoeren. Dit brengt nog een belangrijk voordeel met zich mee. Omdat nodes in principe losse blokjes zijn, hoeft het framework niet van te voren te weten wat voor taken er worden uitgevoerd.

Om deze reden is het plug-in patroon toegepast. Het framework staat nodes toe die aan bepaalde voorwaardes voldoen. Hiernaast moet er wel door een node worden aangegeven wat voor type data deze accepteert, en wat voor type data deze produceert.

Multithreading

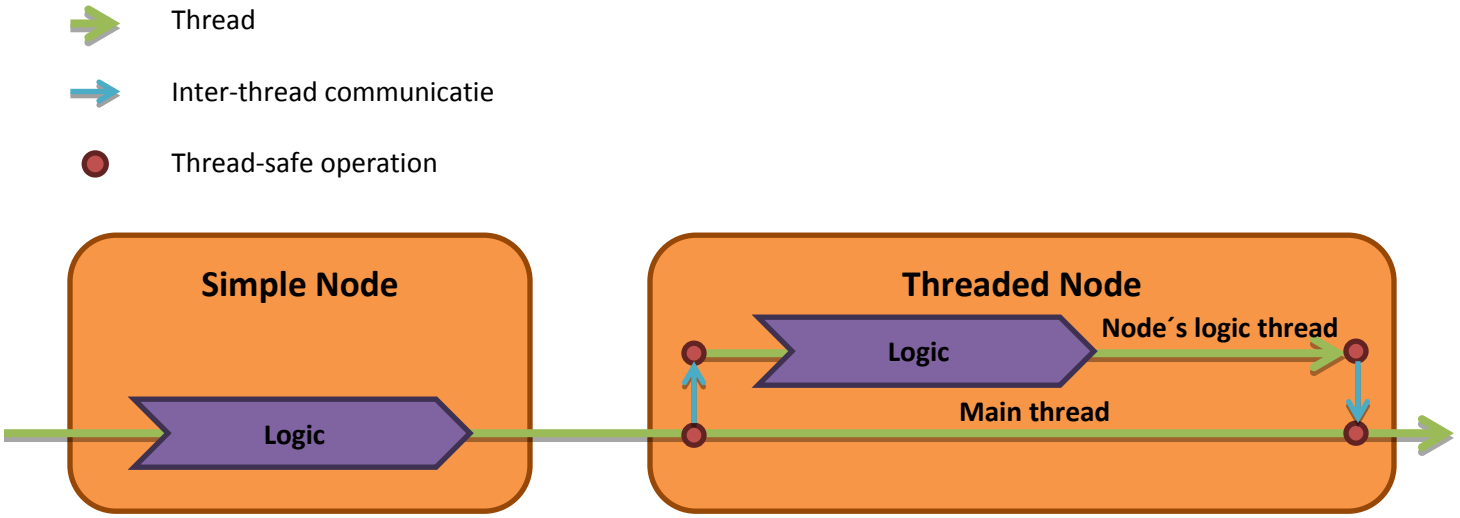
Als er plug-ins gebruikt worden weet het framework echter niet van te voren wat er binnen de pipeline gaat gebeuren. Het zou kunnen dat plug-in nodes vastlopen, of heel lang over hun taak doen.

Om dit op te vangen worden er ook queues toegepast in het framework. Hiernaast wordt elke node op een losse thread uitgevoerd. Dat wil zeggen dat in het geval er een node wat langzamer wordt, de andere nodes door kunnen gaan.



Figuur 5.2.4

In figuur 5.2.5 is te zien hoe dit werkt. De paarse pijlen stellen nog onbekende logica voor, die mogelijk vast kan lopen.

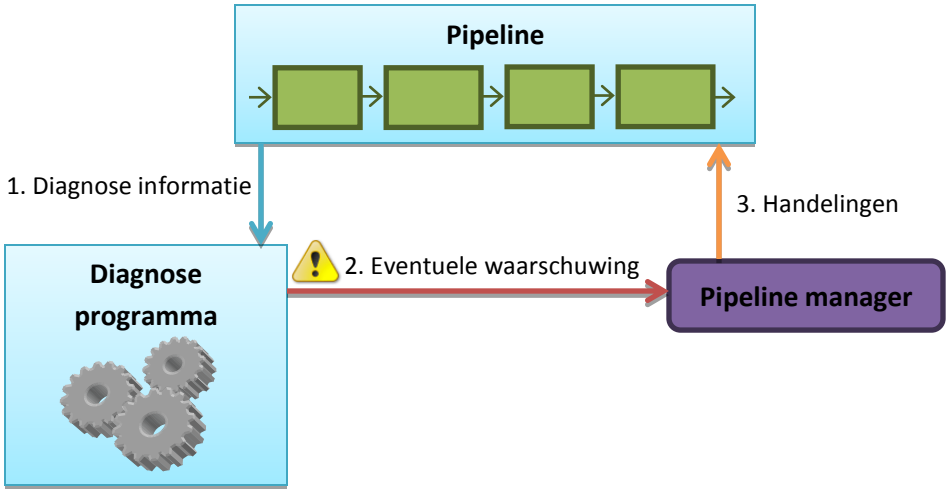


Figuur 5.2.5

Diagnose

Als een node echt vastloopt, kan het gebeuren dat een pipeline niet meer kan functioneren (een opstopping). Om dit af te vangen moet er binnen het framework naast de pipeline ook een diagnose programma draaien. Deze houdt bijvoorbeeld in de gaten hoe lang nodes gemiddeld over hun werk doen, en kan een waarschuwing geven als er iets mis is in de pipeline. Deze diagnose programma's kunnen ook als plug-ins worden aangeboden aan het framework. In figuur 5.2.6 is een overzicht weergegeven van hoe dit in zijn werk gaat.

Onder diagnose informatie valt bijvoorbeeld de tijd die een datapakket erover doet om door een node te gaan.



Figuur 5.2.6

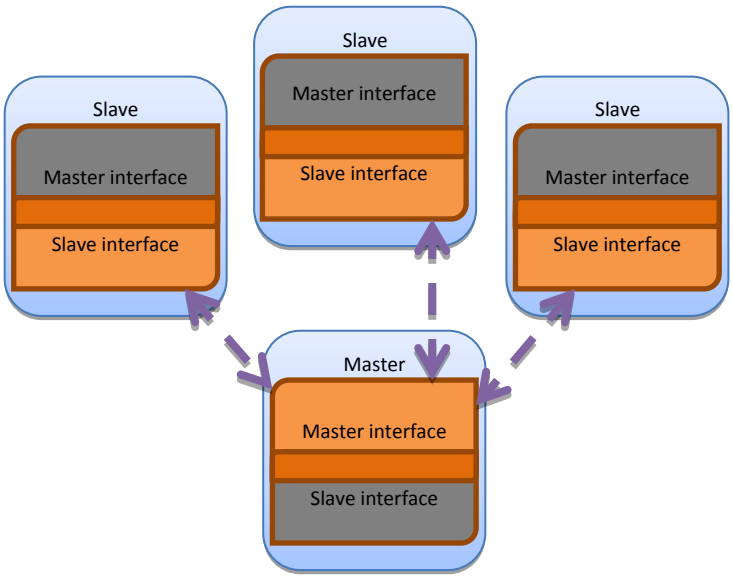
5.3 Cloud Framework

Het cloud framework moet er uiteindelijk voor zorgen dat het systeem op een veilige en vooral gemakkelijke manier op meerdere computers kan draaien. Het cloud framework bestaat uit twee delen. Zo is er het deel wat op de cloud gaat draaien, en het deel wat op de client (bijvoorbeeld een iPhone) gaat draaien. Hieronder wordt voor beide onderdelen uitgelegd hoe ze ontworpen zijn.

Cloud instanties

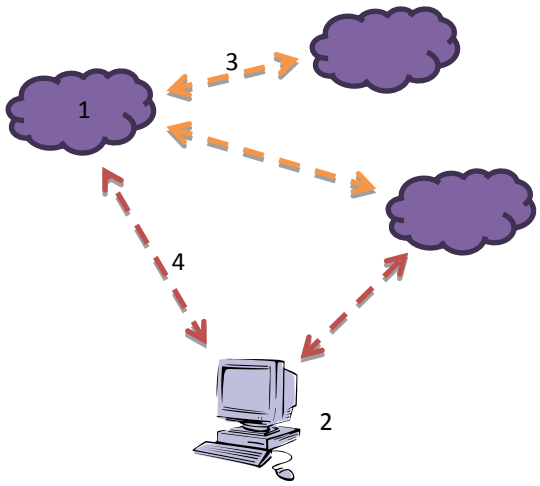
Het cloud deel van het cloud framework zorgt ervoor dat er een cloud gemaakt kan worden, en dat deze stabiel en intact blijft. Voor het opzetten van een cloud is eenmalig hulp nodig van de client. Zo moet er verteld worden welke computers mee mogen doen in de cloud. Verder kan de cloud zelfstandig intact blijven door middel van interne communicatie. Uiteindelijk kan er ook gebruik gemaakt worden van een UDP (User Datagram Protocol) broadcast om cloud instanties te vinden.

Eén deel (-proces) van de cloud die gevormd wordt, wordt een “cloud instantie” genoemd. Een cloud instantie is in individueel proces, dat samen met andere cloud instanties een cloud vormen. Cloud instanties zijn peer processen, wat betekent dat ze gelijk zijn aan elkaar, en dezelfde taken kunnen uitvoeren. In figuur 5.3.1 is te zien hoe dit werkt.



Figuur 5.3.1 - Cloud instanties zijn peers
De gewenste functionaliteit van een peer wordt actief wanneer deze nodig is.

Communicatie tussen cloud instanties wordt interne cloud communicatie genoemd, communicatie naar clients die met de cloud praten (bijvoorbeeld om deze te gebruiken of aan te sturen) wordt externe cloud communicatie genoemd. In figuur 5.3.2 staat dit afgebeeld.



Figuur 5.3.2 - Cloud instanties en communicatie kanalen.

1. Een cloud instantie
2. Client die met cloud communiceert
3. Interne cloud communicatie
4. Externe cloud communicatie

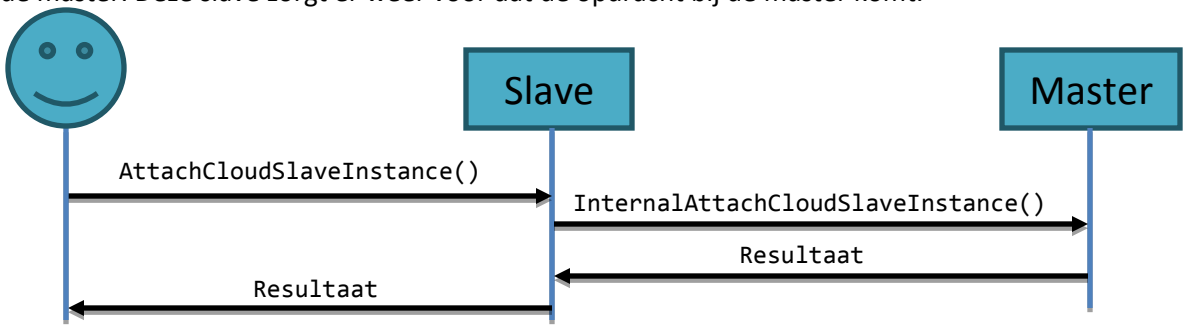
Een cloud instantie heeft drie mogelijke toestanden waar deze zich in kan verkeren:

- Undefined
  De cloud instantie hoort niet bij een cloud.
- Master
  De cloud instantie is de master/leider van een cloud.
- Slave
  De cloud instantie is een slave/volger van de cloud.

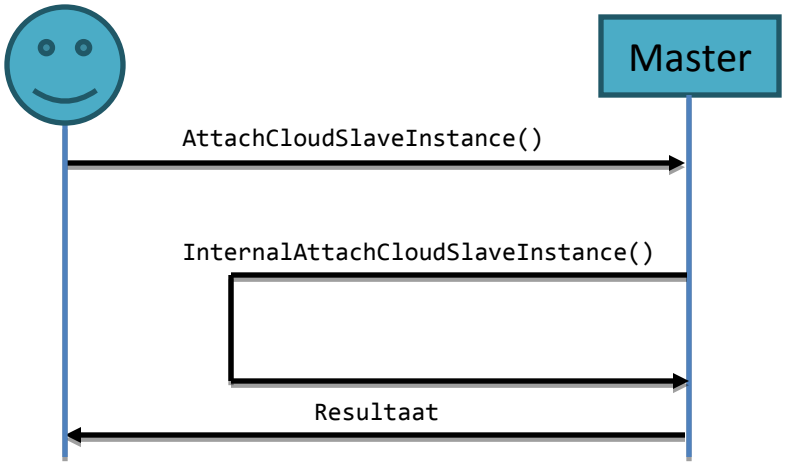
Een belangrijke reden waarom cloud computing gebruikt wordt voor dit project, is om het systeem robuust te maken. Er mag dan ook geen single point of failure zijn. Een groot gevaar is natuurlijk dat er een programma vastloopt of dat er zelfs een computer vastloopt of kapot gaat.

Binnen een cloud geeft de master opdrachten aan de slaves. Ook houd de master in de gaten of alle slaves goed draaien en er niets uitvalt. Op deze manier zou het een probleem worden als de master zelf stopt met werken. Om te voorkomen dat de cloud dan niet meer functioneert, houden ook slaves in de gaten of de master nog draait. Als deze niet meer draait wordt er ingegrepen. Hoe dit precies in zijn werk gaat wordt duidelijk uitgelegd in bijlage 8.

Tot slot worden opdrachten binnen de cloud rond-gecommuniceerd om de illusie te geven dat een cloud één object is. Dit wordt met interne cloud communicatie geregeld. In Figuur 5.3.3 staat een simpel sequentie diagram waarin een externe client (de actor) een opdracht geeft aan een slave, in plaats van de master. Deze slave zorgt er weer voor dat de opdracht bij de master komt.



Figuur 5.3.3 - Communicatie met een slave wordt intern naar de master door gecommuniceerd.



Figuur 5.3.4 - Ook directe communicatie met de master wordt intern afgehandeld met een interne methode

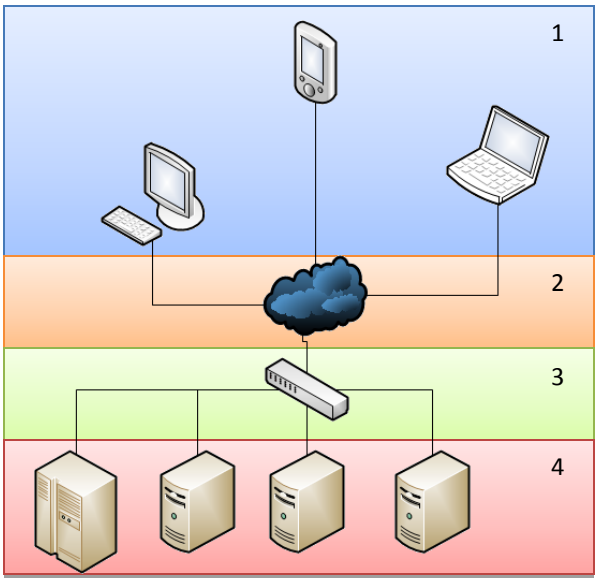
Client

De client van een cloud is het programma wat kan communiceren met een cloud. Dit kan zijn om een pipeline in de cloud te zetten, informatie uit de cloud te verkrijgen en nog vele andere taken uit te voeren.

Over het algemeen kan de client een verbinding hebben met iedere cloud instantie. Ook kunnen er meerdere clients verbonden zijn met de cloud.

Wat de gebruiker niet te zien krijgt, is dat zodra een client een verbinding maakt met een cloud instantie, deze alle informatie over de andere cloud instanties ook doorspeelt naar de client. Op deze manier kan de client automatisch verbinding zoeken met een andere cloud instantie als de oorspronkelijke cloud instantie uitvalt.

Uiteindelijk is het de bedoeling dat de clients via het internet in verbinding komen te staan met de cloud. In Figuur 5.3.5 is dit weergegeven.



Figuur 5.3.5
1. Clients
2. Internet
3. Switch
4. Cloud - verschillende computers die samen een cloud vormen

Dit systeem bevat echter nog twee grote bottlenecks: het internet en de switch. Het systeem is wel zo opgezet dat mocht het internet uitvallen, de cloud gewoon het werk blijft doen. Alleen het streamen over het internet, of het lezen van bronnen die van het internet afkomen is niet meer mogelijk. Een grotere bottleneck is de switch. In overleg met de klant is er bewust voor gekozen om niet na te denken over deze bottleneck, maar meer te focussen op het cloud systeem.

5.4 V.I.P. Framework

Toen zowel het pipeline framework als het cloud framework ontworpen waren was het tijd om de frameworks samen te voegen tot één werkend geheel: het V.I.P. Framework. Hierin was de gedachte om het de gebruiker van het framework zo gemakkelijk mogelijk te maken, zodat ze zich zelf niet hoeven te verdiepen in het optimaliseren hiervan. In het geval van dit systeem is dat het maken en besturen van de pipeline. Uiteindelijk is het systeem zo opgezet dat de gebruiker enkel een pipeline interface[1] heeft om tegen te praten. Hij kan dan acties doen zoals het toevoegen en verwijderen van nodes en connecties.

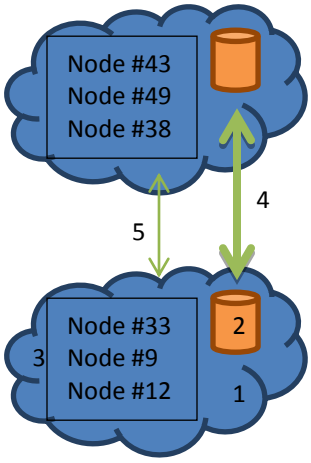
Echter moest er intern nog wel het één en ander gebeuren om dit werkend te krijgen. In dit hoofdstuk wordt hier aandacht aan besteed.

Plugins

Allereerst moest ervoor gezorgd worden dat er een pipeline, die door de gebruiker gedefinieerd is, zo opgedeeld kan worden dat deze goed op de cloud draait. Zoals besproken in het vorige hoofdstuk en te zien is in de attachment[2] functioneert de cloud robust doordat alle cloud instanties op de hoogte zijn van alle data die relevant is in de cloud. Dit moest ook toegepast worden voor de pipeline binnen de cloud. Verder was het pipeline framework ontworpen om uit plug-ins te bestaan. Dit wil in principe zeggen dat er bij elke aanpassing op de pipeline nieuwe plug-ins moeten worden geüpload en gedownload binnen de cloud. Aangezien het niet bekend is hoe groot deze plug-ins kunnen zijn, is dit niet erg veilig. Zo zou kunnen dat het up- of downloaden van een relatief grote plug-in ander dataverkeer binnen de cloud belemmert.

Om deze reden is de pipeline intern gescheiden van de plug-ins. De cloud zorgt er voor dat iedere instantie van de cloud alle plug-ins download wanneer hier tijd voor is. Natuurlijk worden plug-ins die op het moment nodig zijn wel direct gedownload. Verder is de pipeline binnen de cloud niet meer dan een structuur/skelet met verwijzingen (guid's) naar deze plug-ins, die in zichzelf erg weinig data bevat.

Als er nu iets in de pipeline verandert, hoeft alleen de pipeline structuur te worden gecommuniceerd binnen de cloud. De kans dat de nodige plug-ins er nog niet zijn is nog steeds aanwezig, maar een stuk kleiner. In figuur 5.4.1 is weergegeven hoe dit werkt.



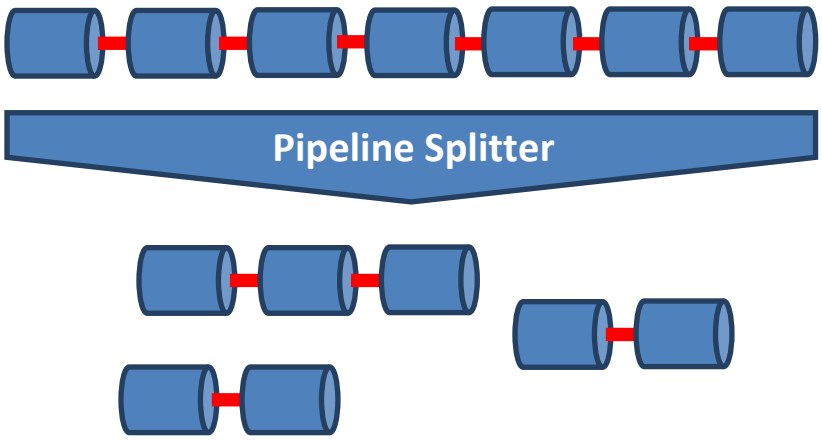
Figuur 5.4.1

1. Cloud instantie
2. Database met plugins
3. Structuur van pipeline
4. Ftp communicatie voor plugin synchronisatie
5. Kleine hoeveelheid communicatie voor pipeline structuur

[1] Zie Hoofdstuk 5.2 - Pipeline Framework
[2] Zie bijlage 7 - Fail tolerantie

Pipeline opdelen

Verder is het van belang dat de pipeline die door de gebruiker geleverd wordt, op een juiste manier verdeeld wordt over de cloud. Dit wordt gedaan door een “pipeline splitter” object. Deze objecten zijn ook als plug-ins te geven aan het framework en er draait er altijd één op de server. Afhankelijk van de plug-in kan zal zo’n object kijken naar de computers die binnen de cloud draaien, en de soort nodes die in de pipeline zitten om een goede opdeling te maken van de pipeline. Het zou zelfs kunnen voorkomen dat er eerst een soort test gedaan wordt om te kijken welke computers het beste om kunnen gaan met welke nodes.



Figuur 5.4.2

Uiteindelijk word er een lijst met nodes naar elke cloud instantie gestuurd die vervolgens een werkende pipeline kan maken van deze informatie.

Pipeline diagnose & node communicatie

Om ervoor te zorgen dat de pipeline voor de client hetzelfde reageert als dat hij op de computer zelf zou draaien is er ook rekening gehouden met de communicatie van en naar de pipeline. Zo communiceert de cloud nog steeds diagnose informatie naar de client, en kan er ook nog naar nodes worden gecommuniceerd.

De diagnose data die wordt terug gecommuniceerd is wel aangepast op de situatie. In de situatie waar de pipeline lokaal draait wordt er een pointer naar de geheugenplek van de data meegestuurd. Dit is echter niet mogelijk als de pipeline op de cloud draait. En het versturen van de data zal voor veel te veel verkeer zorgen[1]. Er is daarom voor gekozen om naast de diagnose data alleen de .ToString() waarde van de data terug te sturen. De client is zo wel op de hoogte van de locatie en status van bepaalde datapakketten. Deze communicatie kan uitgeschakeld worden indien gewenst.

Voor het communiceren naar de pipeline moet de maker van een bepaalde node plug-in “toestemming geven”. Hiervoor moet namelijk de attribute “PipelineCommunicationPropertyAttribute” toegevoegd worden aan de property die moet kunnen veranderen tijdens het functioneren van de pipeline. In figuur 5.4.3 is een stukje c# code te zien waar dit is toegepast op een property genaamd AddValue.

```
[PipelineCommunicationProperty("AddValue", AllowRuntimeUpdate = true)]
public double AddValue
```

Figuur 5.4.3

[1] Zie hoofdstuk 4.5 - Communicatie

5.5 DirectShow filters

Uiteindelijk is het natuurlijk de bedoeling dat er video informatie gefilterd gaat worden. Uit de onderzoeken was gebleken dat DirectShow hier het beste framework voor is. Nu is het van belang om DirectShow in het ontworpen framework te laten werken. In dit hoofdstuk wordt uitgelegd wat voor proces hier bij kwam kijken.

Data toegang

DirectShow maakt gebruik van een filtergraph[1] om video te filteren. Zo'n filtergraph kan op één computer draaien. Echter is het voor ons systeem van belang dat we zo'n FilterGraph kunnen verdelen over meerdere computers. Om dit voor elkaar te krijgen moeten speciale Source en Render filters[1] ontwikkeld worden die in plaats van bijvoorbeeld een bestand lezen en schrijven, de rest van het framework toegang geven tot de video stream die in het geheugen staat.

De source filter noemen we Injection filter, deze staat toe om van buiten bytes in de filter te "injecteren", waarna deze data door de filtergraph kan reizen.

De render filter wordt al door het DirectShow framework aangeboden. Hier kunnen we de SampleGrabber filter voor gebruiken. Deze doet in principe het tegenover gestelde van de injection filter, en staat toe van buiten af data uit de filterGraph te halen.

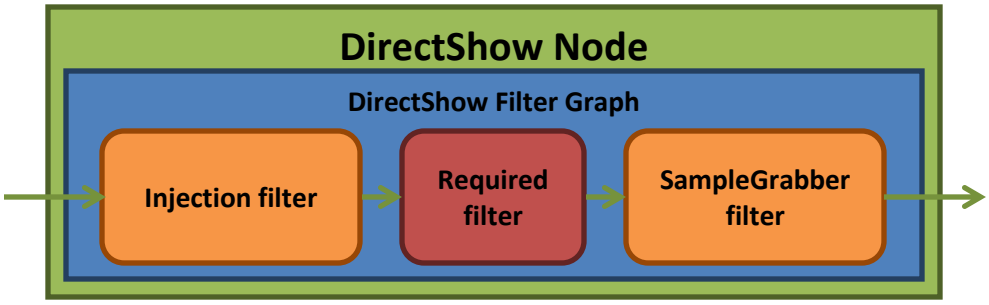
Taal

DirectShow filters worden standaard in de taal C++ geschreven. Er zijn mogelijkheden om dit in C# te doen, maar dit wordt op het officiële DirectShow ontwikkel forum[2] afgeraden. Daarom is in overleg met de klant en technisch begeleider besloten dit toch in C++ uit te voeren mocht de tijd dit toelaten. Als deze filters eenmaal ontwikkeld zijn en naar behoren werken kunnen deze overal worden toegepast en hoeft de opvolger hier niets meer aan te doen.

[1] Zie hoofdstuk 4.7 DirectShow, sub-hoofdstuk DirectShow filter graph
[2] Zie literatuurlijst

DirectShow Node

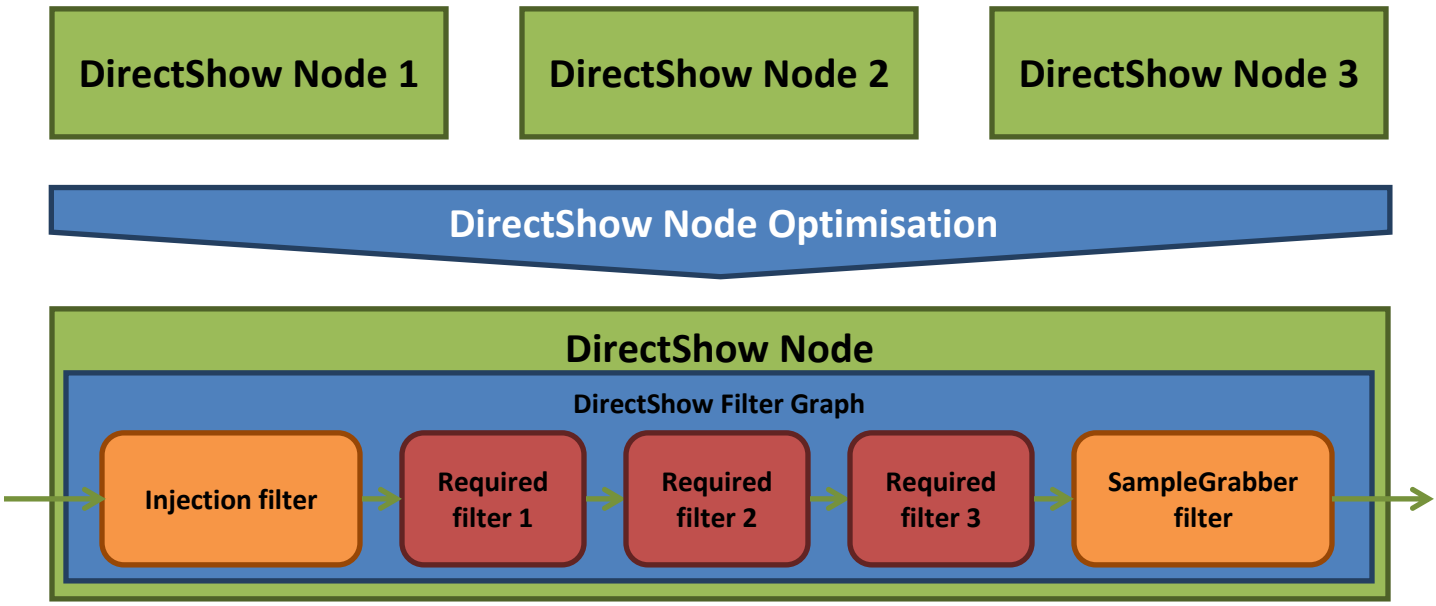
Nu is het de bedoeling om DirectShow binnen het ontworpen framework te gaan draaien. Dit doen we door het in een node te plaatsen. Over het algemeen is het de bedoeling dat er van elke DirectShow



Figuur 5.5.1 - De directshow node
Deze heeft een FilterGraph met daarin altijd een Injection filter, en een Samplegrabber filter.

filter een node gemaakt kan worden. Zo'n node is opgebouwd zoals in figuur 5.5.1 te zien is.

Het kan natuurlijk voorkomen dat verschillende DirectShow nodes achter elkaar op dezelfde computer moeten draaien. Het is dan niet nodig om drie injection filters en drie sampleGrabber filters te gebruiken. De DirectShow Node Optimisation zorgt er dan voor dat deze nodes samen als één DirectShow node gaan draaien. In figuur 5.5.2 is te zien hoe dit werkt.



Figuur 5.5.2 - Meerdere DirectShow nodes worden samengevoegd tot één DirectShow node

5.6 Implementatie

Na het ontwerpen van het framework, is er begonnen met het implementeren hiervan. Binnen de geplande implementatie periode[1] was het mogelijk om het grootste deel hiervan te implementeren. Het enige waar nog niet aan toegekomen is, is het implementeren van het DirectShow gedeelte van het framework[2].

Om uiteindelijk de bestaande applicatie te vervangen moest het framework zijn opgenomen in een server systeem. Dit server systeem is ook geïmplementeerd. Uiteindelijk kunnen er dan clients gemaakt worden die dit systeem kunnen aansturen of uitlezen. In figuur 5.6.1 is een overzicht te zien van de producten die momenteel zijn geïmplementeerd.

Product	Status
V.I.P. framework	Niet af
• **Pipeline framework**	Af
• **Cloud framework**	Af
• **Directshow filters**	Niet af
Server applicatie	Af
Client applicatie	Niet af

Figuur 5.6.1 - Overzicht van producten en voortgang.

Vervolg

Met de klant is afgesproken dat het afronden van de stage prioriteit heeft. Als er echter nog tijd over is om te werken aan de implementatie van het product dan moet er geconcentreerd worden op een applicatie die gebruikt kan worden als demonstratie. Hiervoor moet er een client gemaakt worden die een pipeline in het framework zet en moet er data door de cloud heen kunnen die door deze pipeline wordt verwerkt. Helemaal mooi zou zijn als dit ook al kan met een implementatie van DirectShow.

Naast de implementatie is de opvolger van het systeem op de hoogte gebracht van de werking van het systeem en wat er nog moet worden geïmplementeerd.

[1] Zie hoofdstuk 3 - De opdracht
[2] Zie vorig hoofdstuk

6. Conclusie en aanbevelingen

Conclusie

De V.I.P. afstudeeropdracht is naar tevredenheid en boven de verwachtingen van de klant uitgevoerd.

Hoewel het onderzoek groter bleek te zijn dan vooraf was voorzien, bleek dit geen probleem voor de opdracht. De klant was meer gebaat bij een groter onderzoek. Zelfs met een onderzoek wat meer tijd ingenomen had dan vooraf was gepland, was er nog tijd over om een ontwerp te maken, en een groot deel hiervan te realiseren.

Door een cloud computing model te combineren met een pipeline model is het mogelijk gebleken om een video verwerkings pipeline door meerdere computers samen te laten uitvoeren. Hierdoor wordt rekenkracht van meerdere computers gebundeld, en kunnen voor specifieke taken computers worden gekozen die deze het beste kunnen uitvoeren.

Voor het uiteindelijke systeem is gekozen om WCF te gebruiken en dit te combineren met DirectShow voor de AV bewerking. Het systeem is aan te spreken als web-service.

Uiteindelijk moet het oude product worden vervangen door het V.I.P. framework wat ontwikkeld is als resultaat van deze afstudeerstage.

De afstudeerperiode is binnen Memoria Media afgesloten met een presentatie voor alle belanghebbenden.

Aanbevelingen

Tijdens het onderzoeken en implementeren van het project zijn er een aantal punten voorbij gekomen waar niet naar gekeken is omdat er geen tijd voor was, of omdat de klant hier geen prioriteit aan gegeven had. Een aantal van deze punten kunnen echter wel van belang zijn voor het vervolg van het traject. Om deze reden worden ze hier genoemd.

Er wordt aangeraden dat er verder onderzocht wordt of DirectShow misschien vervangen kan worden door Media Foundation. Media Foundation is over het algemeen nieuwer dan DirectShow en zal in de toekomst waarschijnlijk beter ondersteund worden.

Verder wordt aangeraden dat er onderzocht wordt hoe er omgegaan moet worden met video streams die te groot zijn voor het netwerk. Er is voor gekozen om dit niet te doen binnen dit afstudeerproject omdat de streams die door standaard netwerken kunnen voldoende waren. Echter kan het zijn dat de eisen van de video hoger wordt, en hier later wel behoefte aan is.

Tot slot wordt er aangeraden om bij het cloud framework een broadcast systeem te implementeren. Dit maakt het mogelijk om sneller grotere clouds te vormen zonder dat de gebruiker hierbij betrokken hoeft te zijn.

Evaluatie

De afstudeerperiode bij Memoria Media heb ik als erg positief ervaren. De opdracht was zeer interessant, vernieuwend en uitdagend. Daarnaast heb ik de kans gehad om met leuke mensen samen te werken, en heb ik ook van de begeleiding veel geleerd.

Het was erg leuk dat het project helemaal nieuw was. En dat ik het bovendien zelf helemaal mocht ontwerpen en realiseren. Zo ben ik bij elk aspect van het project betrokken geweest.

Van dit project heb ik geleerd om goed onderzoek te doen binnen een project. Zo was ik al begonnen met het onderzoeken wat de klant precies wenst, en vervolgens bestond een erg groot deel van het project uit het doen van onderzoek en het maken van keuzes. Ik heb hier van geleerd. De klant heeft ook aangegeven dat ze meer hebben gehad aan de manier waarop het project nu uitgevoerd is, dan hoe het verlopen zou zijn als er zonder onderzoek meteen aan de implementatie begonnen was. Ook het ontwerpen en implementeren ging dankzij het uitgebreide onderzoek ook een stuk vlotter.

Het was prettig samenwerken met de medewerkers van het bedrijf. Zo had ik er iedere week een bespreking met de technisch begeleider en de bedrijfsbegeleider waarin de voortgang besproken werd en de planning en requirements doorgenomen of eventueel aangepast werden. Daarnaast werd ik goed ondersteunt.

Met het behaalde resultaat ben ik erg tevreden.

Literatuurlijst

1. Medialooks library

- http://www.medialooks.com/
- http://www.medialooks.com/company/about_medialooks.html
- http://www.medialooks.com/products/software_development_kits/vision_mixer_sdk.html

2. Het oude prototype

- http://www.apple.com/nl/mac/
- http://www.archos.com/products/imt/archos_7/specs.html?country=us&lang=en
- http://www.apple.com/nl/iPhone/specs.html

3. Communicatie

3.1 Protocollen

- http://en.wikipedia.org/wiki/Transmission_Control_Protocol
- http://tools.ietf.org/html/rfc793
- http://tools.ietf.org/html/rfc3168
- http://en.wikipedia.org/wiki/Internet_Protocol_Suite
- http://tools.ietf.org/html/rfc1122
- http://web.inter.nl.net/users/Rohiet.Seosahai/TCP.htm
- http://www.freesoft.org/CIE/Course/Section4/8.htm
- http://en.wikipedia.org/wiki/User_Datagram_Protocol

3.2 Video formats

- http://msdn.microsoft.com/en-us/Library/dd407253(v=VS.85).aspx
- http://msdn.microsoft.com/en-us/Library/dd391027.aspx
- http://msdn.microsoft.com/en-us/library/ms867704.aspx
- http://msdn.microsoft.com/en-us/Library/dd317599(v=VS.85).aspx

3.3 Resolutie en framerate

- http://en.wikipedia.org/wiki/Computer_display_standard
- http://en.wikipedia.org/wiki/Display_resolution
- http://www.sourcesecurity.com/news/articles/co-3289-ga.2806.html
- http://en.wikipedia.org/wiki/Frame_rate

4. Alternatief Medialooks library

4.1 Videolan

- http://www.videolan.org/
- http://wiki.videolan.org/
- http://www.videolan.org/developers/
- http://wiki.videolan.org/Developers_Corner
- http://www.videolan.org/support/faq.html

4.2 Gstreamer

- http://gstreamer.freedesktop.org/
- http://gstreamer.freedesktop.org/dev/
- http://gstreamer.freedesktop.org/data/doc/gstreamer/head/manual/manual.pdf
- http://gstreamer.freedesktop.org/data/doc/gstreamer/head/pwg/pwg.pdf
- http://code.google.com/p/ossbuild/
- http://ossbuild.hoytsoft.org/blog/
- http://ossbuild.hoytsoft.org/forums/

4.3 Media Foundation

- http://msdn.microsoft.com/en-us/library/ms696274(v=vs.85).aspx
- http://en.wikipedia.org/wiki/Media_Foundation

4.4 Leadtools

- http://www.leadtools.com/sdk/multimedia.htm

4.5 VisioForge

- http://www.visioforge.com/
- http://www.visioforge.com/video-edit-sdk.html
- http://www.visioforge.com/video-capture-sdk.html
- http://www.visioforge.com/media-player-sdk.html
- http://www.visioforge.com/video-info-sdk.html

5. DirectShow

- http://msdn.microsoft.com/en-us/library/ms783354(v=vs.85).aspx
- http://msdn.microsoft.com/en-us/library/ms778825(v=vs.85).aspx
- http://msdn.microsoft.com/en-us/library/ms778903(v=vs.85).aspx
- http://social.msdn.microsoft.com/Forums/en/windowsdirectshowdevelopment/threads

6. Pipeline

- http://www.dossier-andreas.net/software_architecture/pipe_and_filter.html
- http://en.wikipedia.org/wiki/Pipes_and_filters
- http://www.roguewave.com/downloads/white-papers/pipelines-overview.pdf
- Freeman, Eric & Elisabeth, *Head First Design Patterns*, O'Reilly
- Landi, Claudio, *Philips presentatie*

7. Cloud computing

7.1 Onderzoek

- http://www.tech-faq.com/distributed-computing.html
- http://en.wikipedia.org/wiki/Distributed_computing
- http://searchcloudcomputing.techtarget.com/definition/cloud-computing
- http://en.wikipedia.org/wiki/Cloud_computing
- http://csrc.nist.gov/publications/drafts/800-146/Draft-NIST-SP800-146.pdf
- http://en.wikipedia.org/wiki/Peer-to-peer
- http://condorstorage.wordpress.com/2008/09/02/open-source-cloud-computing/
- http://www.cse.nd.edu/~ccl/research/pubs/abstractions-cloud-chapter.pdf

7.2 Producten

- http://mpapi.codeplex.com/
- http://www.appistry.com
- http://www.microsoft.com/windowsazure/
- http://www.utilify.com/
- http://sourceforge.net/projects/alchemi/
- http://aws.amazon.com/ec2/