Het ontwikkelen van een multimedia test-tool voor ChipSoft

Afstudeerverslag

Auteur: James Strickland

De Haagse Hogeschool
Informatica

20 maart 2017

Referaat

Dit verslag beschrijft het proces dat James Strickland gedurende 17 weken heeft doorlopen tijdens het uitvoeren van zijn afstudeeropdracht bij ChipSoft in Amsterdam. De afstudeerperiode is op 14 november 2016 gestart en op 20 maart 2017 is dit verslag ingeleverd.

Descriptoren:

- C#
- .NET
- UML
- Scrum
- TWAIN
- Scanner

Voorwoord

Graag wil ik hier mijn bedrijfsbegeleider Jurgen Deege bedanken voor de mogelijkheid om bij ChipSoft af te studeren en voor zijn begeleiding en betrokkenheid.

Verder wil ik graag de begeleiders vanuit de opleiding bedanken voor hun feedback.

James Strickland

Amsterdam, 16-03-2017

Inhoudsopgave

1 Inleiding

Dit verslag is geschreven in het kader van mijn afstudeeropdracht bij ChipSoft. ChipSoft is een bedrijf dat zich specialiseert in de combinatie zorg en ICT en heeft een eigen elektronisch patiëntendossier ontwikkeld, HiX (Healthcare Information eXchange) genaamd.

HiX bevat een multimediacomponent voor het beheren van multimediabestanden, waarmee naast andere functionaliteiten gebruikers documenten kunnen inscannen. Het testen van deze functionaliteit gebeurt met fysieke scanners. Binnen het kader van deze opdracht heb ik voor ChipSoft een virtuele scanner ontwikkeld, waarmee het mogelijk wordt om de multimediacomponent te testen zonder fysieke aangesloten scanners.

Als onderdeel van de opdracht heb ik onderzocht of er virtuele scanners beschikbaar zijn die gebruikt kunnen worden bij het ontwikkelen van het eindproduct. Tijdens dit onderzoek verdiepte ik me ook in scanner softwaretechnologieën. Het uiteindelijk gekozen softwarepakket is een class library, waarmee ik een virtuele scanner heb ontwikkeld.

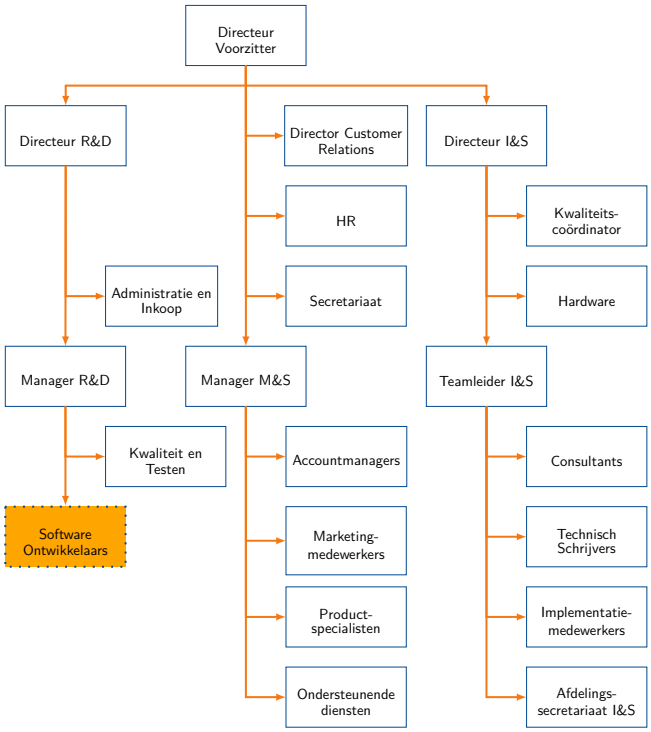
Van sprint één tot zeven beschrijf ik de ontwikkeling van het eindproduct. Per sprint is te zien welke taken zijn geplant en hoe ik de taken heb uitgevoerd. In sprint vier begint het eindproduct een bruikbare vorm aan te nemen als virtuele scanner. Gedurende het verloop van het project worden eerder gemaakte keuzes en ontwikkelingen aangepast, om tot steeds degelijker gebouwde software te komen. Het verslag wordt afgesloten met een evaluatie van het proces dat ik heb doorlopen en de daaruit resulterende producten.

2 Context en opdracht

Dit hoofdstuk plaatst de afstudeeropdracht in de context waarin het is uitgevoerd. Het doel is om het probleem en de aanleiding toe te lichten en het resultaat van de opdracht te beschrijven.

2.1 Bedrijf

Het hoofdkantoor van ChipSoft bevindt zich in Amsterdam en specialiseert zich in de combinatie zorg en ICT. ChipSoft heeft een eigen elektronisch patiëntendossier ontwikkeld, HiX genaamd, die door zorginstellingen in Nederland en België wordt gebruikt. Het is een hiërarchische organisatie met meerdere vestigingen. Ik ben werkzaam in een klein team van softwareontwikkelaars die zich specialiseren in multimedia. De teamleider hiervan neemt in deze opdracht zowel de rol van opdrachtgever als begeleider aan. In figuur 1 wordt de hiërarchie weergegeven.



Figuur 1: Organogram

Zoals in het organogram te zien is, ben ik werkzaam als softwareontwikkelaar op de afdeling Research & Development. Hieronder valt het team van multimedia, waar het zich bezighoudt met het ontwikkelen van de multimediacomponent van HiX. Met dit component beheren

zorginstellingen multimediabestanden in het systeem. Eén van de toepassingen is het inscannen van documenten met scanners.

2.2 Probleemstelling

De multimediacomponent van HiX maakt voor communicatie met scanners gebruik van de TWAIN of WIA API's. Dit zijn twee standaarden die softwareontwikkelaars gebruiken om vanuit een applicatie scanners aan te spreken. Scanner-leveranciers ontwikkelen op basis van deze standaarden drivers voor hun apparaten. Er is een grote diversiteit aan typen scanners die door verschillende leveranciers gefabriceerd worden. Een voorbeeld van een type scanner is er één die voorzien is van een (automatische) documentlader. Dit is ook te omschrijven als een scanner-configuratie. Scanners kunnen met verschillende instellingen documenten scannen, zoals een specifiek kader, papierformaat of verschillende kleurendieptes (de hoeveelheid bits die gebruikt worden om de kleur van een pixel te coderen). In HiX kunnen ICT beheerders bij zorginstellingen vaste scaninstellingen configureren voor bepaalde typen documenten, aan de hand van verschillende scanner-configuraties. Een gebruiker hoeft dan alleen op een knop in HiX te drukken om een document in te scannen. Een beheerder kan dan de opslagruimte beter benutten, omdat bepaalde typen documenten ingescand kunnen worden met minder data door bijvoorbeeld een lagere kleurendiepte te gebruiken. De weergave van het document is dan nog steeds voldoende om het te kunnen lezen.

Bij grote wijzigingen aan HiX moet de multimediacomponent getest worden. Het testen van de scanner functionaliteit gebeurt in de huidige situatie handmatig met fysieke scanners. Dit kost vanwege het beperkte aantal scanners en grote hoeveelheid testcases veel tijd. Een voorbeeld: bij het testen van vijf consecutieve scans door een scanner met documentlader, moet men bij het uitvoeren van de testcase handmatig vijf keer een document in de lader plaatsen. Een ander probleem is dat de oorzaak van storingen lastig te achterhalen kan zijn wanneer bij een zorginstelling bepaalde typen scanners of scaninstellingen niet juist functioneren met HiX. Soms escaleert dit tot een situatie waarbij een bezoek aan de zorginstelling genoodzaakt is, zodat het probleem verholpen kan worden. Tot slot is het zo dat na een overgang van een zorginstelling naar het gebruik van HiX toe, er enige tijd overheen kan gaan voordat scanners aan het systeem aangesloten en geconfigureerd kunnen worden. In die situaties is het wenselijk van tevoren te testen welke typen scanners en scaninstellingen binnen hun netwerk en hardware- en softwareconfiguraties passen.

Hierbij de kern van het probleem: De grote diversiteit aan scanner-configuraties, gecombineerd met alle mogelijke instellingen voor het scannen van een document, levert een grote hoeveelheid variaties op. ChipSoft heeft een beperkt aantal resources, zoals testmachines en scanners, waardoor het veel tijd en mankracht kost om alle variaties te testen.

2.3 Doelstelling

Het doel van de afstudeeropdracht is om een systeem te ontwikkelen waarmee met een virtuele scanner verschillende typen scanners en scaninstellingen getest kunnen worden. Hier voorafgaand wil de opdrachtgever dat ik een softwareonderzoek verricht waarbij ik kijk of er virtuele scanners beschikbaar zijn en of deze voorzien zijn van een API waar het gewenste systeem mee ontwikkeld kan worden. Om tot een goed passende oplossing to komen, verdiep ik me tijdens dit onderzoek ook in scanner softwaretechnologieën zodat ik hier voldoende kennis over heb.

2.4 Resultaat

Het systeem wordt ingezet bij het testen van de multimediacomponent en reduceert het handmatige werk dat komt kijken bij het aansluiten en configureren van scanners. De virtuele scanner is te configureren als elk type scanner die vanuit HiX wordt ondersteunt. Het is met het systeem mogelijk om verschillende scanner-configuraties en scaninstellingen op te slaan en later weer te herroepen. De virtuele scanner registreert TWAIN operaties die het verwerkt in een logbestand, wat helpt bij het diagnosticeren van problemen en verifiëren van testresultaten. Wanneer zorginstellingen storingen ondervinden met scanners, of nog geen scanners hebben geconfigureerd na een overgang naar HiX, baat het systeem bij zowel het op afstand diagnosticeren van storingen als het van tevoren testen van de compatibiliteit.

3 Plan van aanpak

Gedurende de eerste week van de opdracht heb ik het plan van aanpak opgesteld. Dit proces bestond onder andere uit het inventariseren van de door ChipSoft gehanteerde methoden en technieken. Om dit document op te stellen heb ik de opdracht afgebakend en nagedacht over hoe de kwaliteit gewaarborgd zou worden. Na het bepalen van project en ontwikkelmethodieken kon ik het project globaal plannen. Het plan van aanpak is na bespreking goedgekeurd door de opdrachtgever.

3.1 Projectmanagementmethodiek

Tijdens het maken van het afstudeerplan heb ik besloten het project op incrementele en iteratieve wijze uit te voeren. De waterval methode vind ik voor dit project minder geschikt om de volgende redenen:

- Het is een project wat 'from scratch' wordt gestart in een voor mij onbekende omgeving.
- Er zijn, voor mij als individu die het project uitvoert, teveel onbekende factoren zoals:
  - De mensen waar ik mee ga werken.
  - De technologieën waarin ik me moet gaan verdiepen en mee moet gaan werken.
  - Onduidelijke wensen en eisen voor het op te leveren product.

Door deze onbekende factoren is het lastig om van tevoren een passende fasering volgens het waterval-model in te delen. Ze verhogen het risico dat de wensen en eisen aan het op te leveren product onvoldoende worden geformuleerd. Hoe later dat ontdekt wordt, hoe lastiger het wordt om in te spelen op de benodigde wijzigingen. Het gevaar is dan groot dat het gewenste eindproduct niet wordt opgeleverd.

Bij ChipSoft wordt Scrum gebruikt. Hier ben ik naast UP het meest bekend mee wat betreft softwareontwikkelmethodieken. Scrum past goed in dit project vanwege de korte lijnen tussen mij en de opdrachtgever. Er wordt frequent feedback geleverd op verrichte werk en er kan snel worden ingespeeld op wijzigingen. Daarbij zijn niet alle wensen en eisen vooraf bekend en door het hanteren van deze methode kan daar meer rekening mee gehouden worden. Deze redenen hebben mij doen besluiten om het project met scrum aan te pakken.

3.2 Werkwijze Scrum

Het is niet gebruikelijk om scrum als individu uit te voeren. De volgende afspraken zijn gemaakt over de wijze van uitvoering van scrum binnen deze opdracht:

- De opdrachtgever neemt de rol van Product Owner aan. Hij is verantwoordelijk voor de Product Backlog.

- Ik neem de rol van Scrum Master aan. Mijn taak is het beheren van de projectuitvoering.
- Sprints hebben een vaste lengte van twee weken. Dit biedt voldoende tijd om functionaliteiten te ontwikkelen en levert frequent genoeg feedback op om fouten te herstellen en rekening te houden met eventuele wijzigingen. Omdat ChipSoft ook een lengte van twee weken aanhoudt, past dit goed bij het ritme van het bedrijf.
- Ik doe mee met de reguliere Daily Scrum. Al draaien andere multimedia-teamleden, op de Product Owner na, niet mee in dit project, levert deelname hieraan een reflectiemoment op waarin knelpunten geïdentificeerd worden. De Product Owner blijft dan ook dagelijks op de hoogte van de vorderingen.
- Wanneer gekozen Product Backlog items niet af zijn, wordt het resterende werk meegenomen naar de volgende sprint.
- Retrospectives worden in het afstudeerverslag verwerkt.
- Test-driven development wordt als softwareontwikkelmethodiek gebruikt.

3.3 Globale requirements

Om de requirements te achterhalen heb ik gesprekken gevoerd met de opdrachtgever. Het was lastig om hier concreet geformuleerde wensen uit te halen. Daarbij had ik een gebrek aan kennis over scanners. Om dit probleem op te lossen heb ik na het uitbreiden van mijn kennis over scanners, een verdeling gemaakt in typen requirements. Met beter gerichte vragen heb ik achterhaald wat de opdrachtgever wou doen met het eindproduct in combinatie met HiX. Vervolgens heb ik de vastgestelde criteria eenduidig, acceptabel en realistisch gemaakt. Dit heb ik geverifieerd bij de opdrachtgever.

Functionele requirements

- Scanner-configuraties en instellingen die vanuit HiX worden gebruikt bij het scannen kunnen verwerkt worden.
  - Dit gebeurt met een virtuele scanner.
  - Dit kan geautomatiseerd opgeslagen en herroepen worden.
- Gegevensuitwisseling in de koppeling wordt vastgelegd.
- De virtuele scanner kan zonder zijn eigen UI aangestuurd worden.
- De virtuele scanner levert een volgens de gevraagde instellingen correct weergegeven beeldbestand op.

Technische beperkingen

- Het systeem wordt ontwikkeld in C# binnen het .NET framework.
- Het systeem gebruikt de TWAIN en/of WIA standaarden.
- Het systeem beschikt over een API waarmee ontwikkeld kan worden.

Niet-functionele requirements

Uit de ISO 9126[1] standaard:

- Koppelbaarheid - Het systeem werkt volgens een standaard met andere applicaties.
- Wijzigbaarheid - Het is wenselijk later andere scanner-configuraties of scaninstellingen te kunnen implementeren.
- Testbaarheid - De correcte werking van scaninstellingen moet testbaar zijn.

3.4 Softwareonderzoek

Voordat ik begin met de eerste sprint, verdiep ik me in TWAIN en WIA. Met meer kennis over deze technologieën ben ik beter gekwalificeerd om een goed passende oplossing te ontwikkelen. Parallel aan dit proces onderzoek ik of er virtuele scanners beschikbaar zijn op de markt, die gebruik maken van deze standaarden. Aan de hand van de globale requirements in sectie 3.3 stel ik selectiecriteria samen, waarmee ik kan bepalen of de onderzochte scanners binnen het kader van de opdracht toegepast kunnen worden. De opdrachtgever heeft mij gevraagd om dit advies vast te leggen in een rapport.

3.5 Opdrachtscope

Binnen het project wordt er telkens per time-box (sprint) een stukje werkende functionaliteit opgeleverd. De Product Owner beslist per sprint in welke volgorde functionaliteiten worden ontwikkeld. Het is daarom niet van tevoren duidelijk over welke functionaliteiten het eindproduct zal beschikken wanneer de opdracht voltooid is. De beschikbare tijd bepaalt de functionele scope van het systeem, wat betekent dat het zo kan zijn dat niet aan alle vooraf gestelde requirements is voldaan. Ook is het zo dat de scope van de functionaliteiten die wel worden geïmplementeerd kan worden aangepast aan de hand van de beschikbare tijd en complexiteit van implementatie. Dit komt omdat het voor zowel de opdrachtgever als opdrachtnemer niet van tevoren bekend is hoe complex het is om bepaalde functionaliteiten te implementeren.

3.6 Projectorganisatie

Tabel 1: Projectrollen

Naam	Functie	Rol
James Strickland	Opdrachtnemer	Scrum Master, Developer
Jurgen Deege	Opdrachtgever, Begeleider	Product Owner

3.7 Op te leveren producten

De volgende artefacten worden opgeleverd:

- het plan van aanpak;
- het softwareadviesrapport;
- ontwerpdocumentatie;
- het te ontwikkelen systeem en de unit tests die hierbij horen.

3.8 Planning

Tabel 2: Globale projectplanning

Datums	Activiteit
14/11/2016 - 02/12/2016	Plan van aanpak opstellen Softwareonderzoek uitvoeren
05/12/2016 - 16/12/2016	Sprint 1
19/12/2016 - 30/12/2016	Sprint 2
02/01/2017 - 13/01/2017	Sprint 3
16/01/2017 - 27/01/2017	Sprint 4
30/01/2017 - 10/02/2017	Sprint 5
13/02/2017 - 24/02/2017	Sprint 6
27/02/2017 - 10/03/2017	Sprint 7
13/03/2017 - 17/03/2017	Afronden afstudeerverslag

3.9 Projectrisico's

Tabel 3: Projectrisico's

Risico	**Maatregelen**
Deadlines worden niet gehaald	▪ Planning baseren op basis van werkschattingen en sprints. ▪ Overgebleven werk meenemen naar volgende sprints. ▪ Functionaliteitenscope aanpassen.
Niet genoeg kennis van TWAIN en/of WIA	▪ Specificaties en documentatie lezen. ▪ Gericht kennis verdiepen op basis van uit te voeren taken.
Niet genoeg kennis van C#	▪ Collega's om hulp vragen. ▪ Zoeken op internet. ▪ Gericht inlezen over C# op basis van de huidige taak.
Te onervaren met test-driven development	▪ Literatuur raadplegen. ▪ Code simpel houden en alleen bouwen wat nodig is. ▪ Scope van de toepassingswijze van test-driven development aanpassen.
Third-party libraries niet goed gedocumenteerd	▪ API specificaties raadplegen. ▪ Het internet raadplegen.
Moeite met het begrijpen of toepassen van third-party libraries	▪ Simplificeren tot een niveau wat wel begrijpbaar is. ▪ Op te leveren functionaliteitenscope aanpassen. ▪ Beschikbare voorbeelden gebruiken om begrip te verbreden. ▪ Afhankelijkheden verminderen met bijvoorbeeld dependency injection.

3.10 Kwaliteit

In dit project worden een aantal technieken gebruikt om de kwaliteit van de producten te waarborgen:

- **Sprint reviews** - Elke sprint wordt samen met de Product Owner geëvalueerd. Dit levert feedback op waarmee de producten beter aansluiten op het gewenste eindresultaat van de opdrachtgever.
- **Codeconventies** - Ik schrijf code volgens dezelfde codeconventies die ChipSoft aanbeveelt. Dit bevordert de leesbaarheid en overdraagbaarheid van de code.
- **Test-driven development** - Dit levert software op die bestaat uit los van elkaar gekoppelde code, omdat je steeds aan eenduidige en specifieke functionaliteit werkt. Het idee is dat je dan klassen ontwikkelt die verantwoordelijk zijn voor hun eigen gedrag en in minder mate afhankelijk zijn van andere klassen. Bij oplevering komt daarbij dat de software voorzien is van uitgebreide unittests. Dit verhoogt de kwaliteit.
- **Loggen** - Het eindproduct legt gegevensuitwisseling vast in een logbestand. Hiermee kan geverifieerd worden of het de verwachte resultaten oplevert.

3.11 Technieken, tools en software

- **Ontwikkelomgeving** - Op mijn werkstation is Microsoft Visual Studio 2010 Premium met het .NET 4.0 framework geïnstalleerd.
- **Test framework** - Het test framework dat ik gebruik om test-driven development toe te passen is NUnit.
- **Versiebeheer** - Voor versiebeheer maak ik gebruik van de Team Foundation Server bij ChipSoft.
- **UML** - Bij het maken van ontwerpdiagrammen gebruik ik de UML 2.x standaard.
- **Leadtools** - Leadtools is een commercieel softwarepakket bestaande uit een verzameling imaging, oftewel beeldverwerking, SDK's. Twain schrijft het protocol voor waar binnen afspraken gemaakt kunnen worden tussen applicaties en scanners om data over te dragen. De beeldverwerkingsfunctionaliteit is nodig om ingevoerde beelden te kunnen omzetten naar de gevraagde scaninstellingen. Zo kan een scanner bijvoorbeeld een specifiek kader van een beeld scannen (het beeld ondergaat dan een herkadrering), of een grijstintenbeeld opleveren wanneer een kleurenbeeld wordt ingevoerd.
- **Twacker** - Twacker is een TWAIN applicatie waarmee TWAIN scanners getest kunnen worden. Met behulp van deze tool kan geverifieerd worden of een scanner en de features daarvan een correcte werking hebben. De tool vereist veel handmatig werk en heeft niet de meest gebruikersvriendelijke interface, maar biedt wel de mogelijkheid om op een laag niveau een scanner te testen.

- **Twirl TWAIN Probe** - Dit is ook een tool waarmee TWAIN scanners getest kunnen worden. Het is onmisbaar wanneer je aan het ontwikkelen bent met TWAIN. De status van het TWAIN protocol is hierin zichtbaar. Capabilities en hun waardes kunnen uitgelezen, veranderd en gereset worden. Informatie over overgedragen data wordt inzichtelijk. Tot slot geeft deze tool waardevolle informatie over excepties door wanneer de debugger in Visual Studio eraan wordt gekoppeld.
- **Twister** - Twister is ook een test programma voor TWAIN. Het voert een analyse uit op een scanner, door automatisch alle capabilities te doorlopen en een scan uit te voeren met bepaalde instellingen. Na de analyse levert het programma een rapport op, waarin te zien is in hoeverre de scanner en de capabilities voldoen aan de TWAIN specificatie.
- **DosadiLog** - DosadiLog levert een gedetailleerde log op over een lopende TWAIN sessie. Dit is voorwaardelijk aan het gebruik van Twirl TWAIN Probe of Twister, omdat deze producten door hetzelfde bedrijf zijn ontwikkeld. In de log zijn alle TWAIN operaties en de bijbehorende resultaten te zien van de TWAIN sessie.

4 Softwareonderzoek

Voorafgaand aan het ontwikkelproces heb ik voor de opdrachtgever een softwareonderzoek verricht. Tijdens dit onderzoek heb ik me verdiept in scanner softwaretechnologieën. Omdat dit voor mij onbekende en complexe technologieën zijn, wil ik over voldoende kennis beschikken om een goed passende oplossing te bieden. Daarnaast heb ik onderzocht of er virtuele scanners beschikbaar zijn die ik bij de ontwikkeling van het eindproduct kan toepassen. Aan de hand van selectiecriteria heb ik een lijst opgesteld met kandidaten. Uiteindelijk heb ik een advies uitgebracht en vastgelegd in een rapport om het eindproduct met de gekozen kandidaat te ontwikkelen.

4.1 Selectiecriteria

Met behulp van de globale requirements uit het plan van aanpak heb ik de selectiecriteria bepaald. Daarbij heb ik ze uitgebreid na meer kennis te hebben over TWAIN en gespecificeerd wat wenselijk is in de bijbehorende API. De selectiecriteria zijn als volgt gedefinieerd:

- De software moet met C# werken.
- De software maakt gebruik van of is een virtuele scanner.
  - De capabilities (functionaliteiten waar een scanner over beschikt) hiervan kunnen naar gelang uitgebreid of aangepast worden.
- De software gebruikt TWAIN en/of WIA.
- De software beschikt over een API.
- Scans kunnen uitgevoerd worden zonder de UI te tonen.
- De API is goed gedocumenteerd.
- Met de API kunnen berichten naar de scanner vastgelegd worden.
- Met de API is het mogelijk meerdere variaties van scan configuraties en instelling op te slaan en geautomatiseerd te laden.

De selectiecriteria zijn met de opdrachtgever besproken en door hem goedgekeurd.

4.2 Afbakening protocol

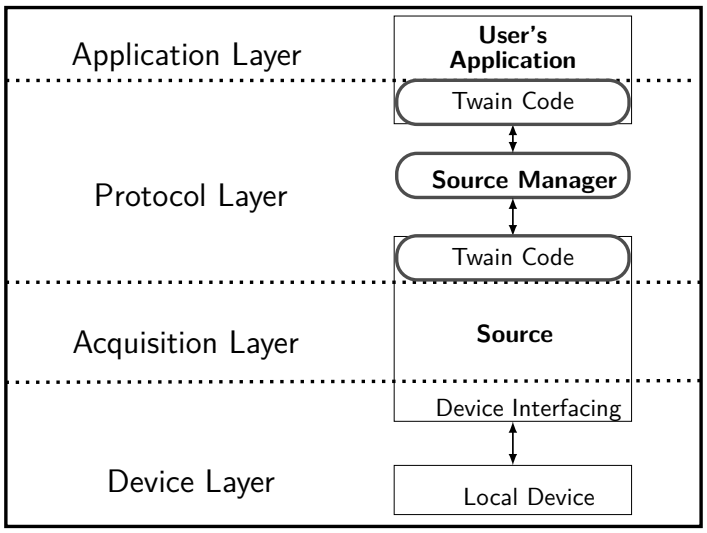
Het werd tijdens het onderzoek duidelijk dat er geen virtuele scanners op basis van WIA te vinden zijn die bij deze opdracht past. Ik vond slechts een enkele, van de TWAIN groep zelf. Dit was een virtueel TWAIN device die compatibiliteit heeft met WIA. Ik kon deze software niet operationeel maken op mijn werkstation. Voor TWAIN kon ik wel een aantal geschikte softwareproducten vinden. Naar aanleiding hiervan heb ik samen met de opdrachtgever besloten om het onderzoek en project voorts te richten op TWAIN.

4.3 Werking van TWAIN

Na het afbakenen van het protocol heb ik de TWAIN specificatie [2] bestudeerd, omdat het voor mij een onbekende technologie is. Dit heb ik gedaan om een beeld te kunnen vormen van de werking, zodat ik beter kon bepalen wat het beste past bij de opdracht. TWAIN definieert een standaard softwareprotocol en API voor communicatie tussen applicaties en beeldacquisitieapparaten (de bron(Engels: source) van de data). Binnen TWAIN zijn er drie software elementen die samenwerken om het overdragen van data mogelijk te maken. Dit zijn de applicatie, de Source Manager, en de Source [2].

- **Applicatie software** - Een applicatie moet aangepast worden om van TWAIN gebruik te maken. In de context van dit project vertegenwoordigt de multimediacomponent in HiX hier de applicatie. Deze is met behulp van een SDK aangepast om met TWAIN scanners te communiceren.
- **Source Manager software** - Beheert de interactie tussen applicatie en de source. De broncode hiervan is beschikbaar in de TWAIN Developer's Toolkit en deze software wordt gratis bij elke TWAIN applicatie en source geleverd.
- **Source software** - Bestuurt het beeldacquisitieapparaat. Ontwikkelaars dienen ervoor te zorgen dat het voldoet aan de TWAIN specificatie.

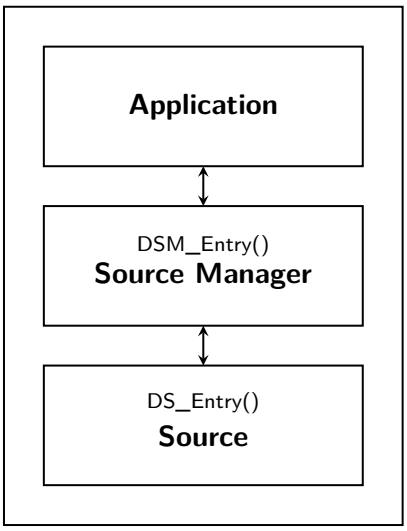
Deze software elementen gebruiken de architectuur van TWAIN om te communiceren. De architectuur van TWAIN is verdeeld in vier lagen [2, p.2-1]. In figuur 2 is de architectuur en de software elementen met hun plaats hierin te zien.



Figuur 2: TWAIN software elementen [2, p.2-2]

De protocollaag is in de TWAIN specificatie het meest grondig en strikt gedefinieerd om de communicatie tussen applicaties en sources nauwkeurig te maken [2, p.2-3].

Communicatie tussen de software elementen gebeurt in TWAIN door middel van twee entry points (de plaats in een computerprogramma waar de controle door het besturingssysteem wordt overgedragen aan het programma en de processor met uitvoering begint). Dit wordt in figuur 3 weergegeven.



Figuur 3: Entry points voor communicatie tussen de software elementen [2, p.2-5]

Op Windows systemen zijn deze entry points te bereiken door middel van DLL bestanden [2, p.2-7]. Applicaties gebruiken de entry point van de Source Manager om te communiceren met de Source en kunnen niet direct de entry point van de Source gebruiken. De entry point van de Source is alleen aan te spreken door de Source Manager. De entry points zijn functies die door de DLL worden uitgevoerd. Aan de hand van de parameters, de richting en oorsprong van de berichten en de Return Codes verloopt het communicatieproces correct, wanneer code die van de entry points gebruik maakt voldoet aan de TWAIN specificatie.

4.4 Plaats van het te ontwikkelen systeem

In de protocollaag van TWAIN, te zien in figuur 2, zijn drie elementen aanwezig[2, p.2-2 t/m 2-3]:

1. Een deel van de applicatiesoftware met de interface tussen applicatie en TWAIN.
2. De Source Manager.
3. Het deel van de software van de Source waarmee instructies vanuit de Source Manager worden ontvangen en data en Return Codes[2, Hs.11] worden teruggeleverd.

Het gewenste eindproduct wordt mijns inziens in de protocollaag geplaatst waar het één van deze elementen vervangt of daar via een interface mee communiceert. Om het systeem in de protocollaag tussen applicatie en Source Manager te kunnen plaatsen, zonder aanpassing

van de applicatie, moeten TWAIN operaties vanuit de applicatie (HiX in dit geval) door het te ontwikkelen systeem opgevangen en doorgestuurd worden naar de Source Manager. Deze implementatie brengt een hoge complexiteit met zich mee in verband met geavanceerde Windows programmeertechnieken die hiervoor nodig zijn. Dat valt buiten wat realistisch gezien haalbaar is voor deze opdracht. Dit heb ik besproken en geverifieerd met de opdrachtgever. Het aanpassen van HiX voor deze toepassing, wat de volgende mogelijkheid is met deze implementatie, is voor de opdrachtgever geen optie en daarmee valt deze implementatie af.

Broncode voor de Source Manager wordt publiekelijk beschikbaar gesteld door de 'TWAIN Working Group', de organisatie die verantwoordelijk is voor de TWAIN specificatie. Dit zou gebruikt kunnen worden om het systeem in de protocollaag te plaatsen als Source Manager. Omdat deze software is geschreven in C++ past het niet bij deze opdracht. Dit heeft twee redenen:

- De source manager wordt gecompileerd naar een DLL bestand. Communicatie bewerkstelligen tussen een DLL bestand ontwikkeld in C++, en een applicatie ontwikkeld in C#, is te complex voor mij als afstudeerder in het kader van deze opdracht.
- C++ code is voor ChipSoft niet overdraagbaar.

Naar aanleiding van deze informatie heb ik samen met de opdrachtgever besloten om het eindproduct te ontwikkelen in de vorm van element 3, beschreven in de genummerde lijst in sectie 4.4. Dat houdt in dat het te ontwikkelen systeem als logische TWAIN source ontwikkeld wordt met een dergelijk element als abstractielaag, of gebruik maakt van een bestaande TWAIN source met interfaces naar het te ontwikkelen systeem.

4.5 Capabilities

Capabilities zijn functionaliteiten waar een beeldacquisitieapparaat over beschikt en kan leveren. Functionaliteiten zoals verschillende resoluties, het scannen van beelden met kleur en automatische documentladers. TWAIN definieert in de specificatie meer dan 150 capabilities [2, p.4-6]. Apparaten hoeven niet alle capabilities te ondersteunen en sommige capabilities komen vaker voor dan anderen.

Er is een grote diversiteit aan acquisitieapparaten die beelddata kunnen leveren. Met TWAIN is het voor applicatieontwikkelaars eenvoudig om met deze grote diversiteit aan apparaten om te gaan in hun applicaties. Door middel van capabilities kunnen applicatieontwikkelaars achterhalen over welke capabilities een source beschikt, om vervolgens hiermee te onderhandelen welke capabilities gebruikt gaan worden. Deze kunnen dan aan de gebruiker van de applicatie aangeboden worden. In sectie 7.7 wordt dit verder toegelicht.

Bij het achterhalen van de globale requirements heb ik samen met de opdrachtgever afgesproken wat praktisch gangbare capabilities zijn van veelvoorkomende scanners. Dit zodat er voor ons beiden een uitgangspunt over welke functionaliteiten wenselijk zijn in scanners. Later in het project worden de capabilities die de multimediacomponent aangeboden wilt hebben vanuit het eindproduct nader gespecificeerd.

4.6 Softwarekandidaten

Na te hebben besloten hoe ik het eindproduct zou ontwikkelen, heb ik onderzocht of er virtuele scanners, of sources, beschikbaar zijn op de markt. De software die ik heb onderzocht onderscheid ik in drie categorieën:

- Kant en klare commerciële TWAIN sources.
- Publiekelijk beschikbare TWAIN sources die naar gelang uitgebreid en gecompileerd kunnen worden.
- Een C# class library, namelijk Saraff.Twain.DS, die de TWAIN specificatie volledig implementeert.

Saraff.Twain.DS is een volledige implementatie van de TWAIN specificatie in C# .NET. Het is een verzameling van klassen en code waarmee ontwikkelaars TWAIN sources kunnen ontwikkelen die voldoen aan de TWAIN specificatie.

In tabel 4 heb ik in de kolom 'TWAIN source' de kant-en-klare commerciële sources onderscheidden van andere categorieën door aan deze een enkele plus toe te kennen. Het laatste item in de tabel is de gevonden class library. Dit waren alle beschikbare software pakketten die ik kon vinden.

Tabel 4: Kandidaten

Softwareproduct	**C#**	**Source**	**Capabilities**	**API**	**Scan zonder UI**
UniTwain	-	+	-	-	-
Scanpoint Virtual TWAIN	-	+	-	-	-
TwainImporter	-	+	-	+	+
Dosadi GenDS	-	++	+	+	+
TWAIN Sample Data Source	-	++	+	+	+
Saraff.Twain.DS	++	++	+	++	+

4.7 Selectie

Om te bepalen in hoeverre de producten voldoen aan de criteria, heb ik ze op mijn werkstation geïnstalleerd en uitgeprobeerd. Ook heb ik, waar beschikbaar, bijbehorende documentatie doorgenomen. Helaas kon ik dit niet doen voor Scanpoint Virtual TWAIN. Er was zowel geen demoversie als documentatie te vinden. Dat betekent dat dit pakket afvalt, omdat ik niet kan vaststellen in hoeverre het aan de criteria voldoet.

Zowel TwainImporter als UniTwain bieden niet de mogelijkheid om capabilities naar wens aan te passen of uit te breiden. Dit is een belangrijk criterium, omdat het wenselijk is in de Source een grote hoeveelheid typen scanner-configuraties en scaninstellingen te bieden. Wanneer de capabilities niet zijn aan te passen, blijft de virtuele scanner beperkt tot de capability en

hun mogelijke waardes die erin zijn gebouwd door de ontwikkelaar. Daarmee vallen deze kandidaten af.

De mogelijkheid om te scannen zonder UI is een belangrijk criterium, omdat het gewenst is bij het testen van de multimediacomponent te scannen op grote schaal en met batchverwerking. Dat maakt het lastig om geautomatiseerd verschillende scanner-configuraties en scaninstellingen op te roepen en uit te voeren, omdat een tester dan steeds tussen elke scan door in de UI de instellingen moet wijzigen. Daarmee valt UniTwain, naast dat het geen interface biedt in de vorm van een API, als kandidaat af.

De producten met een enkele plus bij de API zijn niet geïmplementeerd in C#. Dat betekent dat ondanks met deze producten een systeem te ontwikkelen is dat voldoet aan de requirements, het implementeren hiervan te complex is binnen het kader van deze opdracht. Daarnaast is het niet goed uit te breiden of overdraagbaar voor ChipSoft, omdat hier niet met C++ ontwikkeld wordt.

De reden dat het te complex is, is te wijten aan het feit dat TWAIN implementaties op besturingssystemen de basis hebben in unmanaged code, specifiek C. Er moet gebruik worden gemaakt van een TWAIN.H header bestand [2, p.3-3] en C code om een applicatie of source te laten voldoen aan de TWAIN standaard. Unmanaged code is code die rechtstreeks naar machinetaal wordt gecompileerd en direct door het besturingssysteem gedraaid wordt. C# is een taal waarmee managed code gegenereerd wordt. Managed code draait in een virtuele machine en is voorzien van functionaliteiten zoals memory management. Door het gebruik van gecompileerde managed code hoeft een ontwikkelaar niet bang te zijn om onveilige of onstabiele software te ontwikkelen, waardoor ze zich kunnen richten op bedrijfslogica.

Het mengen van beide vormen van code brengt een niveau van complexiteit met zich mee dat buiten de scope van deze opdracht valt. Daarvoor is veel kennis van beide talen en Microsoft-centrische interoperabiliteit nodig. In het kader van deze opdracht heb ik samen met de opdrachtgever besloten dat dit te complex is en niet realistisch is om te realiseren binnen de beschikbare tijd.

4.8 Conclusie

Vanwege het beperkte aantal keuzes in softwareproducten, en de redenen die net zijn benoemd, hebben de opdrachtgever en ik besloten om verder te gaan met de class library Saraff.Twain.DS. In dit product class library is alle complexiteit die komt kijken bij interoperabiliteit tussen managed en unmanaged code al verzorgd door de ontwikkelaar van de library. Het voldoet volledig aan de TWAIN specificatie en is ontwikkeld in C#. Alle gewenste functionaliteiten kunnen hierdoor volledig geïmplementeerd worden. Dat betekent dat dit product het beste past bij deze opdracht.

5 Sprint 1 - Initiële opzet

Vlak voor aanvang van de eerste sprint werd ik helaas ziek. Aan deze sprint zijn daarom minder uren voor het budget toegekend. In deze sprint heb ik samen met de Product Owner de Product Backlog opgesteld en mijn ontwikkelomgeving opgezet. De allereerste opzet voor het te ontwikkelen systeem heb ik in deze sprint opgeleverd.

5.1 Product Backlog

Op basis van de globale requirements uit het plan van aanpak en gesprekken met de Product Owner is de Product Backlog gemaakt. In de Product Backlog zijn, naast andere items, user stories te vinden. Deze zijn afgeleid van de globale requirements en gericht op het opleveren van specifieke functionaliteiten. Er zijn voor dit project niet veel user stories en slechts een enkele rol. Dit komt omdat het systeem dat ik ontwikkel binnen het bedrijf door medewerkers wordt gebruikt, zonder onderling verschil in rechten of toepassing. In tabel 5 wordt de Product Backlog zoals hij in het begin is opgezet getoond.

Tabel 5: Product Backlog Sprint 1

ID	Item	Prioriteit	Points
US1	Als ChipSoft medewerker wil ik een log kunnen zien van het berichtenverkeer, zodat ik storingen kan diagnosticeren	Middel	13
US2	Als ChipSoft medewerker wil ik scanner-configuraties opslaan, zodat ik vaste scanner-configuraties kan definiëren	Middel	8
US3	Als ChipSoft medewerker wil ik scanner-configuraties kunnen laden, zodat ik vooraf gedefinieerde scanner-configuraties kan kiezen bij het scannen	Middel	8
US4	Als ChipSoft medewerker wil ik scaninstellingen kunnen opslaan, zodat ik vaste scaninstellingen kan definiëren	Middel	8
US5	Als ChipSoft medewerker wil ik scaninstellingen kunnen laden, zodat ik vooraf gedefinieerde scaninstellingen kan kiezen bij het scannen	Middel	8
US6	Als ChipSoft medewerker wil ik beeldbestanden kunnen definiëren om in te laden, zodat deze virtueel ingescand kunnen worden	Laag	5
US7	Als ChipSoft medewerker wil ik beeldbestanden kunnen opslaan, waarmee de correcte werking van de capabilities gevalideerd worden.	Laag	5
PBI1	Initiële opzet Saraff.Twain.DS als ChipSoft TWAIN source	Hoog	8
PBI2	Ontwikkelomgeving opzetten	Hoog	5
PBI3	Solutions in Visual Studio opzetten	Hoog	3

5.2 Prioriteren en toekennen van story points

De prioriteiten zijn bepaald op basis van gesprekken met de Product Owner. De Product Owner weet wat voor de business het belangrijkst is om in het gewenste systeem te implementeren. Items met de hoogste prioriteit worden als eerst uitgevoerd. De getallen waaruit geselecteerd kan worden om story points toe te kennen zijn gebaseerd op de Fibonacci reeks[3] van 1 t/m 21. Dit is een schatting van de hoeveelheid werk het zal kosten om de item op te leveren. Op basis hiervan kan worden bepaald welke items in de komende sprint opgepakt zullen worden.

5.3 Sprint Backlog

Elke sprint worden items uit de Product Backlog geselecteerd en verdeeld in taken. Deze komen in de Sprint Backlog terecht. Voor elke sprint wordt een urenbudget berekend. Om dit budget te bepalen, kijk ik aan het begin van de sprint naar het aantal werkdagen in de betreffende periode. Daar worden een aantal uren van afgetrokken voor zaken zoals pauzes, sprint planningen, vergaderingen, afspraken en feestdagen. Aan deze sprint is een budget van 32 uur toegekend.

Tabel 6: Sprint 1 Backlog

Item	**Taken**	**ETC**
PBI1	Documentatie lezen	8
(Saraff.Twain.DS)	Minimale implementatie uit documentatie	4
	Install package maken	4
PBI2	Source control implementeren	4
(Ontwikkelomgeving)	Test framework implementeren	6
PBI3	Test projecten maken in solutions	2
(Solutions)	Leadtools licentie unlocken	2
	TWAIN sessie starten	2

ie

De kolom 'ETC' in tabel 6 geeft aan hoeveel uren ik heb geschat dat de taak duurt. Dit heb ik bepaald op basis van het sprint budget in combinatie met de toegekende story points in de Product Backlog. Aan de hand van deze schatting kan ik zien wanneer ik te lang bezig ben met een taak. Wanneer dit gebeurt kan het betekenen dat de scope of planning in overleg met de Product Owner aangepast moet worden om de progressie en het resultaat van de sprint te ondersteunen.

5.4 Solutions

Er wordt gesproken over solutions in meervoud, omdat de Product Owner aangaf dat naast de TWAIN source ook een eigen scanapplicatie onderdeel zou zijn van het eindproduct. Deze applicatie incorporeert Leadtools en werkt samen met de TWAIN source die ik ga ontwikkelen. Leadtools bevat ook een SDK voor communicatie met scanners, die ook door de multimediacomponent wordt gebruikt. Met deze applicatie wil de Product Owner scanner-configuraties en scaninstellingen geautomatiseerd laden in de TWAIN source. Na deze sprint wordt er niet meer aan deze applicatie gewerkt, omdat het later niet nodig blijkt te zijn. Toch omschrijf ik hierin de opzet van deze applicatie, omdat het heeft geholpen met het leren omgaan met voor mij nieuwe scanner en softwaretechnieken.

5.5 Opzetten ontwikkelomgeving

Na het aanmaken van de solutions in Visual Studio, heb ik deze met hulp van de opdrachtgever onder versiebeheer gezet in de Team Foundation Server van ChipSoft. Daarna heb ik me verdiept in het werken met NUnit als test framework. Ik vond het belangrijk om de grafische test runner te kunnen gebruiken, omdat van daaruit makkelijker individuele testen uitgevoerd kunnen worden. Dit voorkomt het draaien van een enorme reeks testen na het toevoegen van enkele functionaliteiten. In verband met de wat oudere versie van .NET die ChipSoft gebruikt, kostte dit iets meer moeite dan verwacht. Toen ik het eenmaal werkend kreeg, heb ik in beide solutions testprojecten aangemaakt. Op die manier kan ik het ontwikkelen gescheiden houden van het testen.

5.6 Eerste opzet scanapplicatie

Om de scanapplicatie te ontwikkelen heb ik me eerst verdiept in de Leadtools.Twain API. Het blijkt dat de licentie om Leadtools vrijelijk te gebruiken eerst in de code met een key moet worden vrijgegeven. Om dit in de test-driven development ontwikkelmethodiek toe te passen, besloot ik eerst te testen of een specifieke boolean in Leadtools 'false' is. Voor de tweede test besloot ik om te kijken of ik een TWAIN sessie kon initialiseren met de scanapplicatie. Een TWAIN sessie is de periode gedurende een verbinding tussen applicatie en source via de source manager [2, p.2-11]. Het probleem waar ik hier tegen aanliep, was dat ik niet vanuit Leadtools kon bepalen of de sessie succesvol tot stand is gekomen. Om die reden heb ik besloten om te testen of er geen exceptie was bij het initialiseren van de sessie.

Het tweede probleem waar ik tegen aanliep bij het testen van het succesvol initialiseren van een TWAIN sessie had te maken met een threading probleem. In de scanapplicatie maak ik gebruik van WPF voor de UI. Bij ChipSoft wordt ook WPF gebruikt voor het ontwikkelen van UI's. De reden dat een UI nodig is bij het starten van een TWAIN sessie, is omdat TWAIN onder Windows gebruik maakt van 'window handles' om zowel applicatie als source te identificeren. Een window handle is in dit geval een verwijzing naar een UI scherm. De

TWAIN source manager houdt deze bij en gebruikt het om berichten de juiste kant op te sturen. In WPF is het zo dat UI updates moeten gebeuren op de UI thread. Dit maakt het lastig om te unit testen. De juiste window handle kan niet worden meegegeven aan de method waarmee de TWAIN sessie wordt geïnitialiseerd, omdat die niet bestaat. Je geeft namelijk de huidige window handle mee aan deze method, en aangezien een unit test geen window handle heeft omdat het geen applicatie is met een UI, is dit onmogelijk. Er zijn manieren om deze problemen tegen te gaan, maar die zijn complex en vergen veel tijd om te implementeren. Omdat ik vanuit test-driven development het minimale ontwikkel om een test te laten slagen, heb ik dit heel simpel opgelost.

```
1  // If there is no Application.Current.MainWindow, for example during unit testing, set _owner up
   ↪  with a generic window handle.
2  try
3  {
4      _owner = new WindowInteropHelper(Application.Current.MainWindow).Handle;
5  }
6  catch (NullReferenceException)
7  {
8      _owner = new WindowInteropHelper(new Window()).Handle;
9  }
```

Broncode vermelding 1: Generieke windows handle voor unit tests

In vermelding 1 is deze oplossing te zien. Wanneer de huidge window handle niet kan worden meegegeven, maak ik een generieke window aan die aan de method wordt meegegeven. Nu kan er getest worden of de TWAIN sessie succesvol is geïnitialiseerd.

```
1  namespace ChipSoft_Scan_Test_Tool.Test
2  {
3      [TestFixture, RequiresSTA]
4      [Category("Integration")]
5      public class LeadtoolsInitializationTest
6      {
7          [Test]
8          public void IsLockedShouldReturnFalse()
9          {
10             LeadtoolsInitilization LeadtoolsInit = new LeadtoolsInitilization();
11             Assert.False(LeadtoolsInit._isLocked);
12         }
13
14         [Test]
15         public void SessionStartupDoesNotThrowException()
16         {
17             LeadtoolsInitilization LeadtoolsInit = new LeadtoolsInitilization();
18             Assert.DoesNotThrow(() => { LeadtoolsInit.Startup(); });
19         }
20     }
21 }
```

Broncode vermelding 2: Eerste tests

In vermelding 2 worden de eerste unit tests die ik heb geschreven weergegeven. In regel 4 worden deze tests in een eigen categorie gezet, omdat hier getest wordt of mijn code werkt

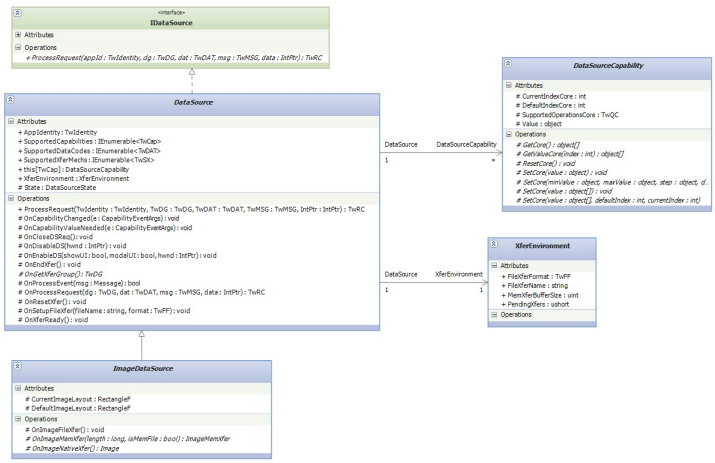
met een third-party API. Dit is puur voor het overzicht gedaan bij het draaien van testen. Door testen te categoriseren kan ik ze makkelijk apart houden en draaien. Na het refactoren en wederom slagen van de tests, is het tijd om verder te gaan met de overige taken.

5.7 Saraff.Twain.DS

TWAIN specificeert het protocol en definieert data structuren waarmee applicaties en scanners kunnen communiceren. Functionaliteiten die scanners bieden, zoals het scannen van documenten, moeten door ontwikkelaars van scanners zelf geïmplementeerd worden. Saraff.Twain.DS is het softwareproduct dat ik gekozen heb om een TWAIN source, oftewel virtuele scanner, te ontwikkelen. Het is een class library die de TWAIN specificatie[2] implementeert in C#. Dat betekent dat ikzelf alle gewenste functionaliteiten in de virtuele scanner ontwikkel.

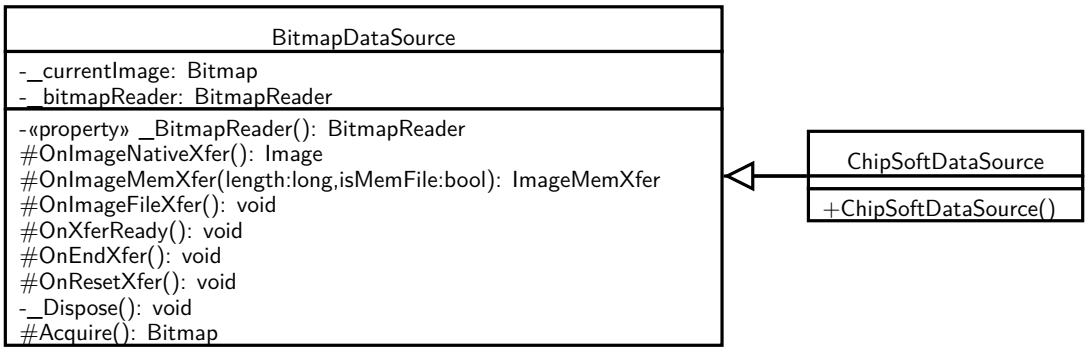
5.8 Minimale implementatie

In de documentatie van Saraff.Twain.DS is een voorbeeld te vinden van een minimale implementatie van een TWAIN source, die niets anders doet dan een beeldbestand uitvoeren in de vorm van een zwarte vierkant van 100 x 100 pixels[4]. Deze implementatie erft van een klasse die de ontwikkelaar heeft meegeleverd, waar ook functionaliteiten in zijn verwerkt waarmee de source data kan overdragen in geheugenbuffers. Dit is een overdrachtsmechanisme die elke TWAIN source moet ondersteunen[2, p.3-38]. Aangezien het programmeren met bitmaps complexiteit met zich kan meebrengen, zeker wanneer ze worden overgedragen in buffers, heb ik ervoor gekozen om van deze klasse gebruik te maken. Dit scheelt veel werk en zorgt ervoor dat de source in ieder geval drie van de vier TWAIN overdrachtsmechanismes ondersteunt. Deze drie mechanismes zijn binnen het kader van de opdracht voldoende voor de Product Owner.



Figuur 4: Saraff.Twain.DS klassendiagram[5]

Voordat ik begon met het implementeren van dit minimale voorbeeld, heb ik de documentatie bestudeerd en bedacht hoe ik zelf aansluiting zou vinden op de class library. In figuur 4 is het klassendiagram die door de ontwikkelaar van de class library is ontworpen te zien. In het minimale voorbeeld erft de ontwikkelaar met de class 'BitmapDataSource' van de class 'ImageDataSource'. Daarom besloot ik zelf met een eigen class 'ChipSoftDataSource' te erven van de class 'BitmapDataSource'. Dit wordt in figuur 5 weergegeven. Hiermee is de basis van de TWAIN source die de komende sprints verder ontwikkeld wordt voltooid.



Figuur 5: Eerste implementatie

5.9 Install package maken

Op Windows besturingssystemen worden TWAIN sources geïmplementeerd als DLL bestanden met een .ds extensie. Een TWAIN source ontvangt via dit DLL bestand, via een enkele entry point, TWAIN operaties van de source manager. De source verwerkt deze operaties en laat het resultaat weten aan de source manager door middel van Return Codes[2, p.2-7]. Saraff.Twain.DS werkt met een DLL bestand, geschreven in unmanaged code, die operaties verwerkt en doorstuurt naar eigen DLL's die met managed code zijn ontwikkeld. Een TWAIN data source kan nooit als standalone programma werken omdat het een DLL bestand is. Dat houdt in dat de DLL bestanden in Windows geregistreerd en geïnstalleerd moeten worden, zodat het besturingssysteem weet van het bestaan en de locatie van deze bestanden. In de documentatie van Saraff.Twain.DS zijn instructies te vinden voor het maken van de Windows install package en de directorystructuur van de DLL's[6], [7]. Na de eerste installatie heb ik een workflow ontwikkeld waarbij ik het project compileer, om vervolgens de DLL's uit de uitvoermap te verplaatsen naar de map in het besturingssysteem waar de DLL's zijn geïnstalleerd. Op die manier kan ik de code compileren en de source gebruiken vanuit TWAIN applicaties, zonder steeds het pakket te hoeven installeren.

5.10 Retrospective

Wat ging goed?

- Communicatie met de Product Owner.
- Een goede basis waarmee verder ontwikkeld kan worden opgeleverd.

Wat ging minder goed?

- Kleiner urenbudget vanwege ziekte.

6 Sprint 2 - Logbestand genereren

Het doel van deze sprint is het vastleggen van TWAIN operaties die de source verwerkt in een bestand. Dit wil de Product Owner gebruiken voor diagnostische doeleinden. Het is de taak van ontwikkelaars van TWAIN sources om fouten bij het scannen op te vangen. Daarnaast levert het bij het uitvoeren van testcases met de source in de toekomst een logbestand op, waarmee geverifieerd kan worden of de test goed is verlopen.

6.1 Sprint planning

Het Product Backlog item waar deze sprint aan wordt gewerkt is in tabel 7 te zien:

Tabel 7: Product Backlog Item voor Sprint 2

ID	Item	Prioriteit	Points
US1	Als ChipSoft medewerker wil ik een log kunnen zien van het berichtenverkeer, zodat ik storingen kan diagnosticeren	Hoog	13

Dit item heb ik samen met de Product Owner verdeeld in taken, waarmee de Sprint Backlog, te zien in tabel 8, is opgesteld. Aan deze sprint is een budget van 56 uren toegekend.

Tabel 8: Sprint 2 Backlog

Item	Taken	ETC
US1 (Logging)	Eerste opzet UI (WinForms MVP)	16
	TWAIN operaties -> XML	12
	TWAIN operaties -> UI	12
	Logbestand genereren	8
	Logbestand opslaan naar bestand	8

6.2 Eerste opzet UI

Elke TWAIN source wordt voorzien zijn van een UI om gebruikers te assisteren bij het ophalen van data uit de scanner. De Product Owner gaf aan de gegenereerde log ook in de UI terug te willen zien. De reden dat ik ervoor heb gekozen om WinForms te gebruiken, in plaats van WPF, is omdat het opvragen van de in sectie 5.6 genoemde window handle minder complexiteit met zich meebrengt dan in WPF. De Product Owner is hiermee akkoord gegaan. Omdat ik met test-driven development ontwikkel, wil ik functionaliteiten in de UI kunnen testen. Echter dit levert in WinForms soortgelijke threading problemen op, zoals besproken in sectie 5.6.

Exploratief onderzoek op internet leerde mij dat dit goed is op te vangen door bij het ontwikkelen van de UI het MVP, of Model-View-Presenter, softwarearchitectuur pattern toe te passen. In deze pattern is het doel om presentatie te scheiden van logica, zodat deze losjes gekoppeld zijn in de code. Het model is een interface waarin data wordt gedefinieerd. Dit is de data die getoond wordt, of waarop gereageerd wordt vanuit de UI. De presenter reageert op het model en de view. Data wordt erdoor opgehaald uit het model, en naar de gewenste weergave geformatteerd voor de view. De view is een interface die data uit het model weergeeft, en gebruikerscommando's stuurt naar de presenter in de vorm van events. De presenter reageert vervolgens op deze commando's.

Na deze keuzes gemaakt te hebben, heb ik een eerste opzet gemaakt van de UI , oftewel de 'view'. Voor de logging functionaliteit heb ik bedacht om een apart tabblad in de UI te maken met de label 'Diagnostics'. In het scan-tabblad heb ik wat code overgenomen uit het voorbeeld van Saraff.Twain.DS. In verband met het sprintbudget, laat ik het UI gedeelte nu hierbij. Het heeft alle aan UI gerelateerde taken toegekende tijd gekost om tot deze keuze te komen en een eerste implementatie te maken. De uiteindelijk UI is in bijlage B te zien.

6.3 TWAIN operaties formatteren naar XML elementen

Ik ben enige tijd bezig geweest met het proberen te achterhalen hoe ik op een laag niveau elke TWAIN operatie kon vastleggen. Wat ik voor ogen zag, was letterlijk elke TWAIN operatie, met de resulterende Return Codes, vastleggen bij de entry point van de DLL. Het bleek na het bestuderen van de software, dat het implementeren van gegevensvastlegging op die wijze zou vereisen dat de code van Saraff.Twain.DS aangepast moet worden. Dat vind ik een risico, omdat ik er vanuit ga dat de huidige correcte werking en het ontwerp van de software door de ontwikkelaar getest is. Wanneer ik zelf aanpassingen maak, is het mogelijk dat ik de correcte werking en stabiliteit van de software op ongewenste wijze beïnvloed.

Na het bespreken hiervan met de Product Owner, zagen we dat er in de class library methods beschikbaar zijn die worden afgevuurd bij bepaalde TWAIN operaties. Deze methods zijn voldoende om te gebruiken voor het vastleggen van gegevens. Aangezien dit virtual methods zijn, kan ik hierin code plaatsen in de context van het TWAIN protocol. Een virtual method is in C# namelijk overerfbaar en te overschrijven.

Na besloten te hebben hoe de klasse ChipSoftataSource TWAIN operaties zou vastleggen, richtte ik mijn focus op de formattering hiervan voor het logbestand. De Product Owner en ik hebben besloten om gebruik te maken van XML. De eerste stap was een logische verdeling maken in de beschikbare methods, op basis van naamwoorden in de benaming. Hierin zag ik, met een enkele uitzondering, dat er hoofdzakelijk drie events zijn waar de methods betrekking op hebben. Dit zijn:

- source events;
- capability events;
- xfer (overdracht) events.

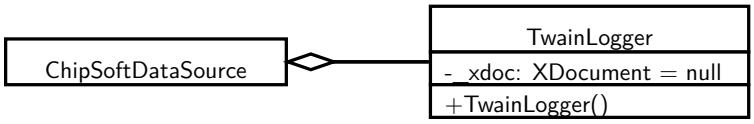
Deze woorden declareer ik als root elementen. Op basis van andere woorden uit de method declaratie, zoals werkwoorden, heb ik child elementen gedeclareerd. Method parameters leveren de content van het element. Per method stelde ik op deze wijze een XML element string samen.

6.4 XML valideren

Nu ik voor elke method een XML element had, heb ik hier strings van gemaakt met nep method argumenten erin verwerkt. Om de strings te valideren, heb ik unit tests geschreven die de strings parsen. Als de strings succesvol worden geparset, retourneert de method valide XML elementen. Door dit proces uit te voeren tot de unit tests slagen, waarborg ik de validiteit van XML in het logbestand. Dit helpt later fouten voorkomen bij het genereren, parsen en lezen van de XML.

6.5 Voorontwerp

Voordat ik begon met het ontwikkelen van de functionaliteit, heb ik het klassendiagram uitgebreid. Zoals in figuur 6 te zien is, heeft ChipSoftDataSource een TwainLogger klasse tot zijn beschikking. TwainLogger heeft een document om de gegenereerde XML op te slaan. Het type daarvan wordt bepaald door de gekozen C# API voor XML verwerking, namelijk LINQ to XML.



Figuur 6: TwainLogger uitbreiding

6.6 Tests waarmee TwainLogger is ontwikkeld

Hier volgen de tests die ik heb geschreven om volgens test-driven development de TwainLogger klasse te ontwikkelen:

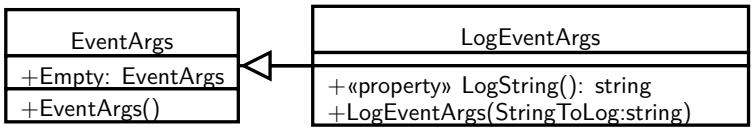
1. Wordt er een document aangemaakt bij de initialisatie?
2. Kan een XML element worden toegevoegd aan het document?
3. Lukt het om de tijdelijke map van de huidige systeemgebruiker te vinden?
4. Bestaat het logbestand nadat het document erin wordt opgeslagen?

Uit deze tests ontstond code waarmee een door de source gegenereerde logbestand in de tijdelijke map van de gebruiker opgeslagen werd. Na een aantal keer exploratief de werking

hiervan te testen, kwam ik er achter dat het gegenereerde bestand telkens niet meer dan een enkel XML element bevatte. Naar aanleiding hiervan heb ik aan de klasse een root element attribuut toegevoegd, waarbij ik opmerkte dat ik dit van tevoren ook had willen testen. Daarna heb ik de code aangepast om elementen aan dit root element toe te voegen. Na deze aanpassing genereerde de code logbestanden met meerdere genestelde XML elementen in het root element.

6.7 LogEventArgs

De method in TwainLogger waarmee XML elementen worden toegevoegd aan het logbestand wordt uitgevoerd door de methods benoemd in sectie 6.3. Die methods leveren elk een eigen hard gecodeerde string als parameter, bestaande uit de eerder gevalideerde XML elementen. Bij het ontwikkelen van deze functionaliteit merkte ik dat dit toch niet de beste constructie was, omdat het bij bepaalde methods voorkwam dat ik niet bij de TwainLogger klasse kon. Dit had onder andere te maken met de volgorde van interactie en scope tussen bepaalde objecten, methods en events. Daarbij merkte ik op dat er nu een sterke koppeling was tussen de methods in ChipSoftDataSource en de logging method in de TwainLogger klasse. Om de problemen met de logica op te lossen en de code te refactoren, heb ik voor TwainLogger een event ontwikkeld.



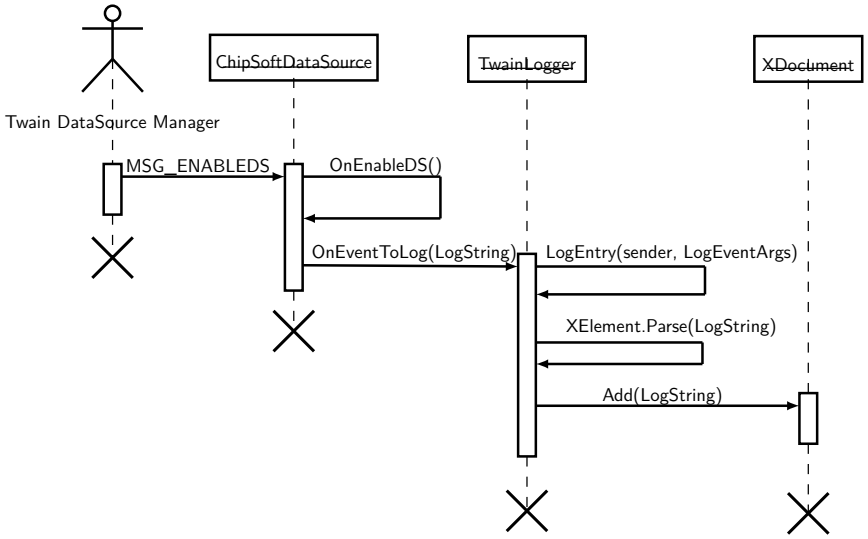
Figuur 7: LogEventArgs uitbreiding

In plaats van een hard gecodeerde method call door andere methods, laten de methods in ChipSoftDataSource nu een event afgaan, met de XML string als argument. .NET biedt een ingebouwde manier om deze functionaliteit te implementeren. Door het erven van de standaard EventArgs klasse kun je hier gebruik van maken en indien gewenst argumenten aan het event meegeven. Dit is een variant van de 'Observer' design pattern. Het is een door Microsoft aangeraden conventie om de naamgeving in de eigen klasse te eindigen met 'EventArgs'. In figuur 7 is deze uitbreiding op het klassendiagram te zien.

6.8 Ontwerp tot nu toe

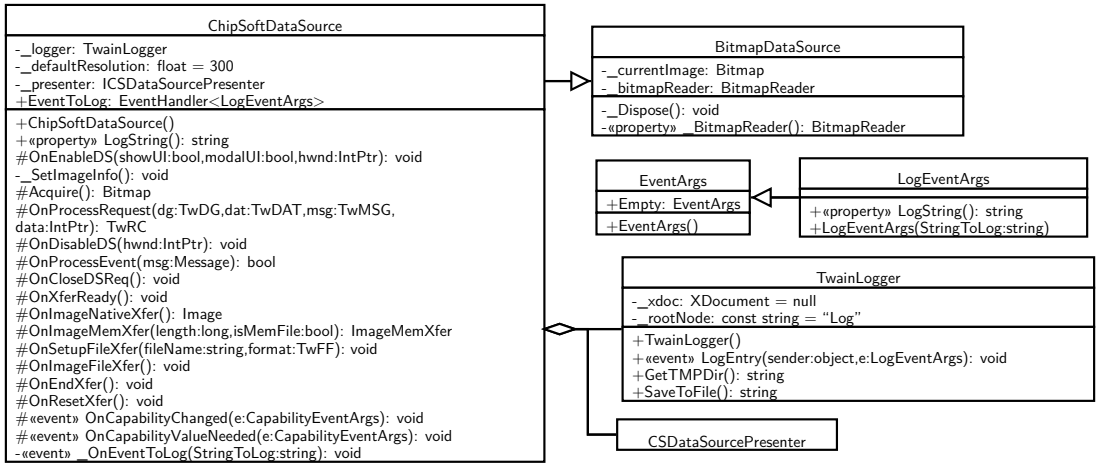
In figuur 8 illustreer ik met behulp van een sequence diagram verder de implementatie van het 'Log' event. Na het refactoren is TwainLogger niet meer afhankelijk van specifieke methods in ChipSoftDataSource om 'LogEntry' uit te voeren. ChipSoftData source is nu uit te breiden met eigen methods om het event af te laten gaan. Met dit ontwerp kun je ook andere

objecten ontwikkelen die luisteren naar het 'Log' event. Voorheen zou je dat hard moeten koppelen in de betreffende methods in ChipSoftDataSource.



Figuur 8: LogEvent sequence diagram

In figuur 9 is het klassendiagram van de virtuele scanner zoals hij tot nu toe is ontwikkeld te zien. Het UI gedeelte heb ik, op de presenter na, achterwege gelaten. Dit is namelijk nog niet uitgewerkt.



Figuur 9: Klassendiagram sprint 2

6.9 Retrospective

Wat ging goed?

- Communicatie met de Product Owner.
- 'Observer pattern' geïmplementeerd om objecten losjes te koppelen.
- Goede ontwerpkeuze gemaakt met betrekking tot de UI.
- Goede tests bedacht voor het ontwikkelen van de 'TwainLogger' klasse.

Wat ging minder goed?

- Tijd verloren aan het uitzoeken en implementeren van UI gerelateerde ontwerpbeslissing en code.
- Tijd verloren door te moeilijk te denken over het implementeren van de logging functionaliteit.

7 Sprint 3 - UI en capabilities

In deze sprint is de UI verder ontworpen en ontwikkeld. Het wegschrijven van log-regels naar de UI is ook opgeleverd. De multimediamodule van HiX is door mij onderzocht om te inventariseren welke capabilities de virtuele scanner moet ondersteunen. Deze zijn samen met de Product Owner geprioriteerd. Om het implementeren van capabilities goed te kunnen uitvoeren is zowel de TWAIN specificatie, als de documentatie van Saraff.Twain.DS geraadpleegd. Tot slot zijn enkele capabilities geïmplementeerd, met het doel om te leren hoe het ontwikkelen daarvan in zijn werk gaat.

7.1 Sprint planning

In tabel 9 worden de items van de Product Backlog getoond waar deze sprint aan gewerkt worden. De Product Owner en ik gaan er al vanuit dat het opleveren van PBI6 meerdere sprints zal beslaan. Vandaar het hoge aantal story points en lagere prioritering voor dit item.

Tabel 9: Product Backlog Items voor Sprint 3

ID	Item	Prioriteit	Points
PBI4	UI verder uitwerken	Hoog	13
PBI5	Log-regels wegschrijven naar UI (Restant sprint 2)	Hoog	8
PBI6	Capabilities waarmee vanuit HiX gescand kan worden implementeren in de virtuele scanner	Middel	21

Deze items zijn opgesplitst in uit te voeren taken om het item op te leveren. Het sprintbudget is bepaald en uren zijn toegekend aan de taken. Deze sprint heeft een budget van 56 uren.

Tabel 10: Sprint 3 Backlog

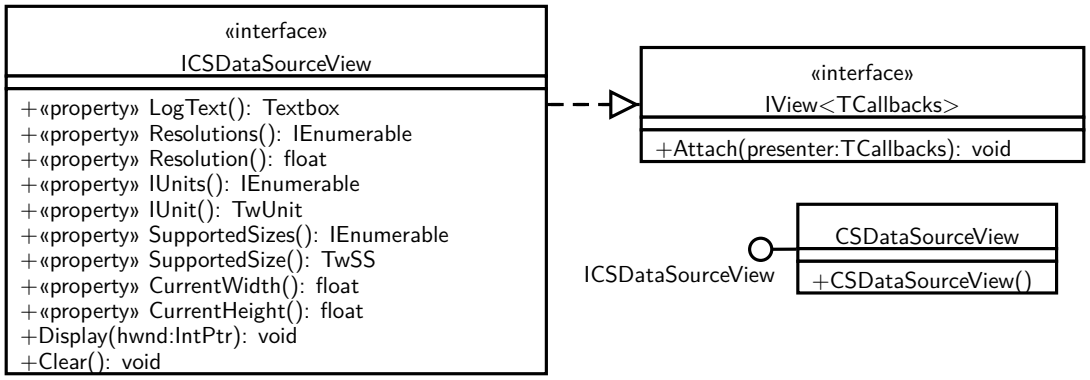
Item	Taken	ETC
PBI4	MVP Passive view implementeren	12
(UI uitwerken)	UI voor scannen ontwerpen	8
PBI5	UI component voor log-regels bepalen	8
(Log -> UI)	Log-regels -> gekozen component	4
	Save knop toevoegen om bestand apart op te slaan	8
PBI6	Capabilities HiX inventariseren	8
(Capabilities matchen)	Capabilities één voor één implementeren	8

7.2 Passive view

Er zijn meerdere manieren om de MVP design pattern, omschreven in sectie 6.2, te implementeren. Ik heb ervoor gekozen om 'Passive View' te gebruiken, zoals omschreven door Martin Fowler [8]. Het idee achter deze implementatie van MVP, is dat de view geheel passief is. De presenter is verantwoordelijk voor het updaten van de view. Dit betekent dat de presenter getest kan worden, zonder afhankelijk te zijn van een bepaalde implementatie van een daaraan gekoppelde view.

Bij andere implementaties is er meer koppeling tussen de view en de presenter. Daarbij is het bij andere implementaties nodig meer rekening te houden met synchronisatie tussen model en view. Daarom vond ik 'Passive View' het meest geschikt om te gebruiken met mijn ontwikkelmethode. De presenter kan getest worden zonder daadwerkelijke UI. Deze redenen hebben mij doen besluiten om voor 'Passive View' te gaan.

7.3 Klassendiagrammen UI

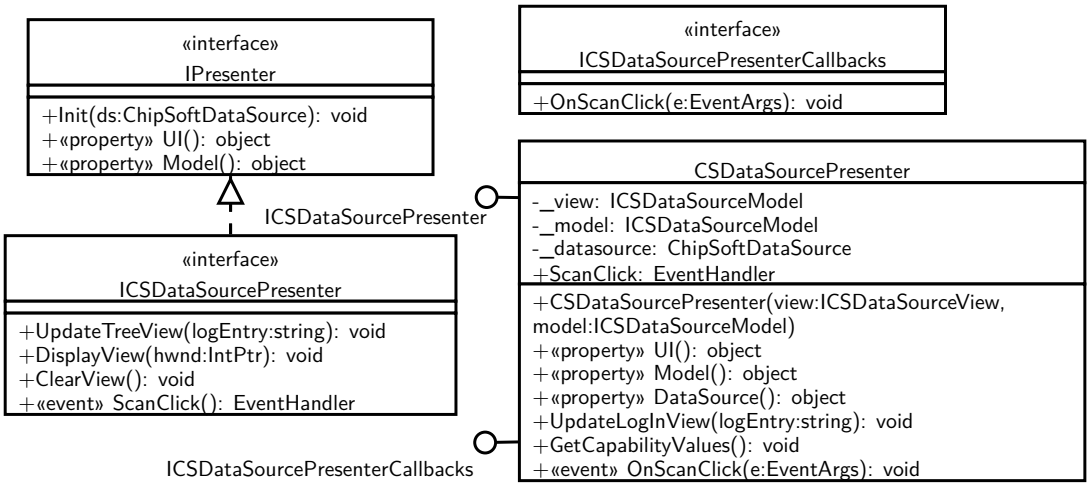


Figuur 10: View klassendiagram

In figuur 10 is het klassendiagram voor het view gedeelte van de UI te zien. Ik heb interfaces gedefinieerd zodat het mogelijk wordt om met nep view objecten de presenter te unit testen met behulp van dependency injection. In de constructor van de presenter, te zien in figuur 11, worden interfaces in de argumenten meegegeven in plaats van objecten. Op die manier kan elk object als view meegegeven worden aan de presenter, als het maar de interface implementeert. Methods die in de interface gedefinieerd worden kunnen binnen elke object die de interface implementeert een eigen implementatie hebben. Bijkomend voordeel is dat nu ook meerdere views ontwikkeld kunnen worden, die allemaal hetzelfde 'contract' gebruiken voor onderlinge communicatie. De model heeft dezelfde opzet en klassendiagram.

In figuur 10 is in de 'IView' interface een 'Attach' method te zien. Aan deze method wordt, ook gebruik makend van dependency injection, een callback interface meegegeven als argument.

Deze interface is te zien in figuur 11. Bij het initialiseren van de presenter wordt in de view de 'Attach' method uitgevoerd, waarbij de presenter als argument meegegeven. In de 'Attach' method van de view is het dan mogelijk om events vanuit de view, zoals het klikken op een knop, te koppelen aan methods in de presenter.



Figuur 11: Presenter en callback klassendiagram

7.4 Log-regels wegschrijven naar de UI

Bij het ontwikkelen van deze functionaliteit heb ik eerst geprobeerd, gebruik makend van het eerder door mij ontwikkelde 'Log' event, om log-regels weg te schrijven naar een 'treeview' component. Een treeview component is een UI component waarin data in een boomstructuur wordt weergegeven. Dit leent zich uitstekend voor XML, omdat een XML document ook is opgebouwd volgens een boomstructuur. Je kunt dan gericht bepaalde informatie opzoeken aan de hand van de boomstructuur.

Na teveel tijd hieraan gespendeerd te hebben, wat ik kon afleiden uit het sprintbudget en de planning, heb ik samen met de Product Owner besloten dat het voldoende is om het logbestand te spiegelen in een tekstbox component. Ik heb niet gebruik gemaakt van het 'Log' event. In plaats daarvan heb ik ervoor gekozen om na het updaten van het logbestand, de tekstbox leeg te maken en het log-bestand als string in de tekstbox te zetten. Op die manier geeft de tekstbox het logbestand weer.

7.5 Save knop voor log-bestand toevoegen

Tijdens het gebruik van de virtuele scanner, wordt er een logbestand gegenereerd. Dit wordt naderhand automatisch opgeslagen op het werkstation in de tijdelijke map van de gebruiker.

Zo lukt het altijd om het bestand automatisch op te slaan, omdat een gebruiker altijd in zijn eigen tijdelijke map schrijfrechten heeft. De Product Owner wenst een knop in de UI waarmee het logbestand apart bewaard kan worden met een standaard 'Save file' dialoog.

Dit bleek lastiger te bouwen dan ik aanvankelijk had ingeschat. Weer liep ik tegen threading problemen aan, dit keer omdat ik te maken had met de 'Save file' dialoog. Communicatie tussen verschillende componenten dient in dit geval in een single thread te gebeuren. Na overleg met de Product Owner hebben we besloten om dit item op te leveren in een volgende sprint.

7.6 Capabilities inventariseren

De virtuele scanner die ik ontwikkel wordt gebruikt om de scanfunctionaliteit in HiX te testen. Daarom moet het alle capabilities ondersteunen die in HiX zijn in te stellen. Aan de hand van gesprekken wist ik wat de Product Owner wenste te doen met de virtuele scanner. Deze wensen kon ik koppelen aan de scaninstelling in HiX en de broncode hiervan. Volgens de TWAIN specificatie bleek ook dat bepaalde capabilities afhankelijk zijn van andere capabilities. Op basis van deze informatie heb ik de capability inventaris samengesteld, te zien in bijlage A.

Samen met de Product Owner is een MoSCoW prioritering toegekend aan de capabilities. In MoSCoW prioritering zijn er vier prioriteiten waar uit gekozen kan worden. Deze zijn:

- Must have
- Should have
- Could have
- Would be nice to have

'Must have' heeft de hoogste prioriteit, en 'Would be nice to have' heeft een prioriteit die zo laag is dat het naar alle waarschijnlijkheid niet geïmplementeerd wordt. In die volgorde wordt aan de capabilities gewerkt.

7.7 Capabilities implementeren

Om de capabilities te implementeren heb ik me verdiept in de documentatie van Saraff.Twain.DS en de TWAIN specificatie. Een aantal capabilities uit de capability inventaris, te zien in bijlage A, zijn volgens de TWAIN specificatie verplicht [2, p.5-14]. Deze worden standaard in Saraff.Twain.DS ondersteunt [9]. Ter verduidelijking herhaal ik hier nogmaals dat TWAIN alleen het protocol en de data structuren definieert. Dit betekent dat wanneer de virtuele scanner bepaalde capabilities ondersteunt, het hier met een TWAIN applicatie afspraken over kan maken. Het is aan de ontwikkelaar van de scanner om de functionaliteit van de capability te implementeren.

Tabel 11: Capability containers

Capability container	Type inhoud
TW_ONEVALUE	Een enkele waarde.
TW_ARRAY	Een array van waardes die de huidge capability omschrijven.
TW_RANGE	Een reeks gelijkmatig verdeelde waardes met een minimum en maximum.
TW_ENUMERATION	Definieert een lijst van waardes van waaruit de 'Current' waarde gekozen kan worden.

TWAIN maakt gebruik van een viertal datastructuren, capability containers genaamd, waarmee applicaties kunnen onderhandelen met een TWAIN source over de capabilities waar de source over beschikt [2, p.2-15]. Aan de hand van deze containers wordt bepaald welke waardes mogelijk zijn en kunnen ze worden ingesteld voor de capability. Alle capabilities hebben een 'Default' en 'Current' waarde. In tabel 11 worden deze vier datastructuren weergegeven.

Saraff.Twain.DS beschikt over klassen die gebruikt kunnen worden om capabilities te implementeren. In het klassendiagram in figuur 4 in sectie 5.8 is te zien dat de 'DataSource' klasse beschikt over 'DataSourceCapability' klassen. ChipSoftDataSource erft ook alles van de klasse 'DataSoure'en beschikt hier ook over. In Saraff.Twain.DS zijn klassen die erven van 'DataSourceCapability', die direct te matchen zijn aan de containers omschreven in tabel 11.

In de documentatie wordt omschreven hoe nieuwe capabilities ontwikkeld kunnen worden aan de hand van enkele voorbeelden [10]. Het is nog wel nodig om de TWAIN specificatie te raadplegen, zodat de capabilities op de juiste manier worden geïmplementeerd en werken volgens de verwachting van het TWAIN protocol. Deze sprint heb ik gebruikt om mijn kennis hierover te verbreden en de voorbeelden uit de documentatie uit te proberen. Daarbij heb ik geleerd hoe ik vanuit de door mij ontwikkelde UI met de capabilities kon werken. Aan het einde van de sprint had ik op deze manier voldoende kennis om in de volgende sprint de capabilities te implementeren. Wat ik aan het einde heb opgeleverd is de mogelijkheid om de scanresolutie aan te passen. Nogmaals, het enige wat nu hiermee gebeurd is dat de applicatie en scanner een resolutie hebben afgesproken. De functionaliteit waarbij de resolutie van het beeldbestand aan de hand van deze afspraak wordt aangepast moet ik nog ontwikkelen. Wel is in de metadata van het beeldbestand na uitvoer de afgesproken resolutie te zien. Zo heb ik gevalideerd dat de capability werkt.

7.8 Retrospective

Wat ging goed?

- Goede implementatie van het 'MVP' pattern gekozen, namelijk 'Passive View'.

- Passive View goed geïmplementeerd, te merken aan de losse koppeling en testbaarheid van de presenter.
- Verduidelijkt welke specifieke capabilities de virtuele scanner moet ondersteunen.
- Geleerd hoe capabilities te implementeren.

Wat ging minder goed?

- Hoeveelheid werk bij het ontwikkelen van een save knop voor het logbestand onderschat.
- Tijd verloren bij het proberen log-regels weg te schrijven naar een 'treeview' component.

8 Sprint 4 - Capabilities en Leadtools

In deze sprint heb ik het ontwerp van de virtuele scanner gedocumenteerd met UML diagrammen. Een groot deel van de capabilities worden na deze sprint ondersteunt door de scanner. Daarnaast heb ik Leadtools verwerkt in de scanner, zodat ik beeldbestanden kon opleveren die voldoen aan de afspraken die door middel van de capabilities gemaakt zijn. De scanner begint nu echt bruikbaar te worden.

8.1 Sprint planning

In tabel 12 ziet u de Product Backlog items die voor deze sprint zijn geselecteerd. PBI6 heeft nu minder story points, omdat ik na het sprint 3 het idee heb gekregen dat het wat minder werk zal zijn dan aanvankelijk gedacht. PBI7 is het werk wat is overgebleven uit PBI5 van sprint 5, namelijk de save knop. Ook heeft de Product Owner aangegeven wat meer informatie in de log te willen. Dit is in de Sprint Backlog in tabel 13 te zien.

Tabel 12: Product Backlog Items voor Sprint 4

ID	Item	Prioriteit	Points
PBI6	Capabilities waarmee vanuit HiX gescand kan worden implementeren in de virtuele scanner	Hoog	13
PBI7	Log functionaliteiten afronden	Middel	8
PBI8	Ontwerp documenteren	Hoog	5

Tabel 13 geeft de Sprint Backlog van deze sprint weer. Aan deze sprint is een budget van 52 uur toegekend.

Tabel 13: Sprint 4 Backlog

Item	Taken	ETC
PBI6 (Capabilities)	Eén voor één de capabilities uit de inventaris ontwikkelen	16
PBI7 (Log functionaliteiten afronden)	Save knop maken	16
	Logbestand voorzien van timestamp in de benaming	4
	Gebruiker en werkstation toevoegen aan log	4
	Informatie over datasource toevoegen aan log	4
PBI8 (Ontwerp documenteren)	ChipSoftDataSource klassendiagrammen	4
	Passive View klassendiagrammen	2
	LogEventArgs klassendiagram en sequence diagram	2

8.2 Ontwerpdocumentatie

De Product Owner gaf aan dat hij graag het ontwerp wou zien van wat er tot nu toe ontwikkeld is. Vooraf aan het ontwikkelingsproces had ik wel informeel bedacht hoe het ontwerp eruit zou komen te zien. De documentatie van Saraff.Twain.DS is al voorzien van enige ontwerpdiagrammen, zoals het klassendiagram in figuur 4 in sectie 5.8. Dit heb ik uitgebreid door 'ChipSoftDataSource' te laten erven van 'BitmapDataSource'. Dit was letterlijk het eerste ontwerpbesluit.

Bij het ontwikkelen van de logger had ik al van te voren bedacht dat ik in 'ChipSoftDataSource' een aparte klasse zou gebruiken waarin gelogd werd. Dit omdat ik graag goed object georiënteerd wil ontwikkelen, waarbij klassen zoveel mogelijk een enkele verantwoordelijkheid hebben. Bij het ontwikkelen van het 'Log' event kwam ik erachter dat het lastig is om dit met alleen klassendiagrammen te illustreren. Daarom besloot ik om er een sequence diagram bij te maken, zodat de context van het event en de objecten die er gebruik van maken duidelijker wordt.

Bij het ontwikkelen van de virtuele scanner heb ik met behulp van test-driven development eerst tests geschreven, gevolgd door de code om de tests te laten slagen. Daar waar dit niet zo goed mogelijk was, heb ik vanwege het veelvuldige gebruik van third-party libraries en code me moeten beperken tot wat binnen deze kaders mogelijk was. Deze twee factoren hebben ertoe geleid dat ik niet van tevoren veel heb ontworpen. Wel heb ik tijdens het ontwikkelen zoveel mogelijk principes van goed object georiënteerd programmeren toegepast, en zodoende kon ik binnen deze sprint het ontwerp tot nu toe vastleggen. Dit omdat er daadwerkelijk wel is nagedacht over het ontwerp tijdens het ontwikkelen. De diagrammen die ik hiervoor heb gemaakt heb ik aan de opdrachtgever opgeleverd en besproken. Ze zijn in dit verslag verwerkt met de aan het onderwerp relevante delen in de respectievelijke secties, zoals in secties 5.8, 6.5, 6.7, 6.8 en 7.3.

8.3 Leadtools

Naarmate ik meer van de gewenste capabilities aan het ontwikkelen was, werd het duidelijk dat de scanner die ik aan het ontwikkelen was prima afspraken kon maken met applicaties aan de hand van de capabilities, maar niet de afgesproken uitvoer gaf. ChipSoft maakt in HiX gebruik van Leadtools voor diverse toepassingen. Alle toepassingen van Leadtools binnen HiX omschrijven valt buiten de scope van dit project.

Leadtools is een verzameling digital imaging SDK's met toepassingen voor onder andere rasterbeelden, document imaging (bijvoorbeeld het scannen van documenten) en medical imaging. Hiermee kunnen applicatieontwikkelaars digital imaging technieken in hun applicaties verwerken. HiX gebruikt bijvoorbeeld een deel van Leadtools voor het scannen van documenten via het TWAIN en WIA protocol. Een ander onderdeel van Leadtools is een SDK waarmee digitale beelden onder andere bewerkt, opgeslagen en getoond kunnen worden. Omdat er veel komt kijken bij het programmatisch bewerken van digitale beelden en ChipSoft al beschikt

over een SDK daarvoor, maak ik in de virtuele scanner hier gebruik van. Dit scheelt veel tijd en de scanner kan met behulp hiervan beeldbestanden opleveren zoals via het TWAIN protocol afgesproken wordt.

8.4 Beeldbestand geselecteerd

Tot nu toe voert de scanner een zwarte vierkant uit van 100 x 100 pixels. Om beter te kunnen valideren dat aan de afgesproken capability waardes voldaan wordt na het uitvoeren van een scan, heb ik gezocht naar een ander beeldbestand voor de scanner. Op internet zag ik op diverse TWAIN en scanner fora dat een gangbare native resolutie voor veel scanners 300 dpi is.

Dpi staat voor dots per inch. In dit geval kunnen dots ook als pixel worden gezien. Het beeldbestand bestaande uit 100 x 100 pixels is dan een vierkant blaadje van 1 x 1 inch, bij een scaninstelling van 100 dpi. Ik heb me gericht op het A4 papierformaat, omdat dit in Nederland een veel gebruikte standaard is voor fysieke documenten. Daarbij heb ik met de Product Owner besproken dat de scanner in A4-formaat moet kunnen scannen.

TWAIN gebruikt inches als standaard maateenheid, alhoewel het ook mogelijk is om met andere maateenheden te werken. Een vel papier op A4 formaat heeft een grootte van 8,27 x 11,69 inches. Aangezien ik 300 dpi als standaard native resolutie heb gekozen in de virtuele scanner, ben ik op zoek gegaan naar een beeldbestand dat qua aantal pixels overeenkomt met het A4 formaat met 300 dpi. Dit is een beeldbestand geworden van 2481 x 3507 pixels, want 8,27 x 300 = 2481 en 11,69 x 300 = 3507.

8.5 Resolutie capabilities

Nu dat ik een beeldbestand had die overeenkwam met specifieke waardes, kon ik me verdiepen in het bouwen van de functionaliteit die hoort bij het kiezen van een andere dpi waarde bij het scannen. Als een scanner met een andere dpi waarde scant, zie je dit terug in het aantal pixels van het beeldbestand. Een hogere dpi betekent dat er een grotere pixeldichtheid is, en een lagere dpi vertaalt zich naar een kleinere pixeldichtheid.

Dit betekent dat het beeldbestand wat nu 2481 x 3507 pixels is bij het scannen van 300 dpi omhoog of omlaag wordt geschaald bij een andere dpi waarde. Als een dpi van 150 wordt ingesteld bijvoorbeeld, wordt het beeldbestand omlaag geschaald en bestaat het uit 1240 x 1753 pixels. Op die manier kon ik valideren of alles juist werkte na het ontwikkelen van de functionaliteit.

8.6 Capabilities functioneel maken

Het heeft enige tijd geduurd om de documentatie van Leadtools te lezen en in de scanner te implementeren. Ik besloot me eerst te richten op de functionaliteiten met betrekking tot de

layout van het beeld. Scanners hebben namelijk de mogelijkheid om het scannen te beperken tot een bepaald gebied van het document, zodat niet altijd het hele document gescand hoeft te worden.

Het eerste wat ik heb geïmplementeerd was het omlaag en omhoog schalen van het beeldbestand op basis van de gekozen dpi. Toen dat werkte heb ik ervoor gezorgd dat het mogelijk was om met andere dimensies te scannen, zoals A5 of een zelf in te vullen breedte en hoogte. Daarna heb ik het mogelijk gemaakt om te scannen met een andere 'Kleurendiepte'. Kleurendiepte is een maat voor de hoeveelheid bits die gebruikt worden om de kleur van een pixel te coderen. Dit kon ik allemaal met Leadtools doen. Met een enkel beeldbestand kan de scanner nu aan de hand van capability waardes diverse soorten beeldbestanden opleveren in verschillende formaten.

8.7 Supported Sizes capability

Eén van de capabilities die ik heb geïmplementeerd is ICAP_SUPPORTEDSIZES [2, p.10-202], te zien in tabel 19 in bijlage A. Hiermee kunnen standaard papierformaten gebruikt worden bij het scannen. Wederom bleek dat ik deze formaten zelf moest definiëren en implementeren. Met de Product Owner had ik afgesproken dat de scanner in A4, A5, en een zelf gedefinieerde formaat moest kunnen scannen. Met het oog op uitbreiding in de toekomst om meer formaten te ondersteunen heb ik hier een object voor gemaakt. Dit object bevat een simpele data structuur met width en height attributen, en een dictionary waarbij de door TWAIN gedefinieerde formaten in ICAP_SUPPORTEDSIZES als key gelden en de eigen data structuur als value. Uiteraard heb ik in de dictionary op deze wijze de A4 en A5 formaten gedefinieerd.

Helaas kwam ik tijd tekort in de sprint om aan Product Backlog item PBI7 te werken. Het koste veel tijd om Leadtools te bestuderen voordat ik ermee kon ontwikkelen. Daarbij zijn nog niet alle capabilities geïmplementeerd. Dit wordt meegenomen naar de volgende sprint.

8.8 Retrospective

Wat ging goed?

- Product Owner zeer tevreden met de resultaten uit deze sprint.
- Veel gangbare scanner functionaliteiten geïmplementeerd.

Wat ging minder goed?

- Teveel geplant, waardoor ik een item achterwege moest laten.

9 Sprint 5 - Logging refactoren en capabilities opslaan

In deze sprint heb ik de log functionaliteiten uit voorgaande sprints afgerond, en hierin het 'Builder' design pattern toegepast. Daarbij heb ik in deze sprint functionaliteit in de scanner opgeleverd waarmee de huidge capability waardes in een XML bestand opgeslagen kunnen worden en later weer opgevraagd kunnen worden. Wij hebben aan deze functionaliteit voor deze sprint een hogere prioriteit toegekend, dan aan het implementeren van de overige capabilities.

9.1 Sprint planning

Tabel 14: Product Backlog Items voor Sprint 5

ID	Item	Prioriteit	Points
US2 t/m US5	Scanner-configuraties en scaninstelling opslaan en laden	Hoog	13
PBI7	Log functionaliteiten afronden	Hoog	8

In tabel 14 worden de voor deze sprint gekozen Product Backlog items getoond. US2 t/m US5, te zien in tabel 5 in sectie 5.1, worden allemaal opgeleverd wanneer ik de functionaliteit implementeer om de huidige capability waardes op te slaan. Er is namelijk overlap tussen scanner-configuraties en scaninstelling. Een (automatische) document feeder is bijvoorbeeld een configuratie die een scanner kan hebben, en bij het scannen met deze configuratie kan je dan instellen of je gebruik maakt van de automatische feeder.

Tabel 15: Sprint 5 Backlog

Item	Taken	ETC
US2 t/m US5	XML elementen formattering bedenken	4
	UI uitbreiden met selectiebox en save en load knoppen	4
	XML bestand genereren & opslaan	8
	XML bestanden tonen in selectiebox	8
PBI7	Save knop maken	14
(Log functionaliteiten	Logbestand voorzien van timestamp in de benaming	2
afronden)	Gebruiker en werkstation toevoegen aan log	2
	Informatie over scanner toevoegen aan log	2
	Code refactoren	12

In tabel 15 zijn de taken te zien die uitgevoerd moeten worden om de Product Backlog items op te leveren. Aan deze sprint is een budget van 56 uren toegekend.

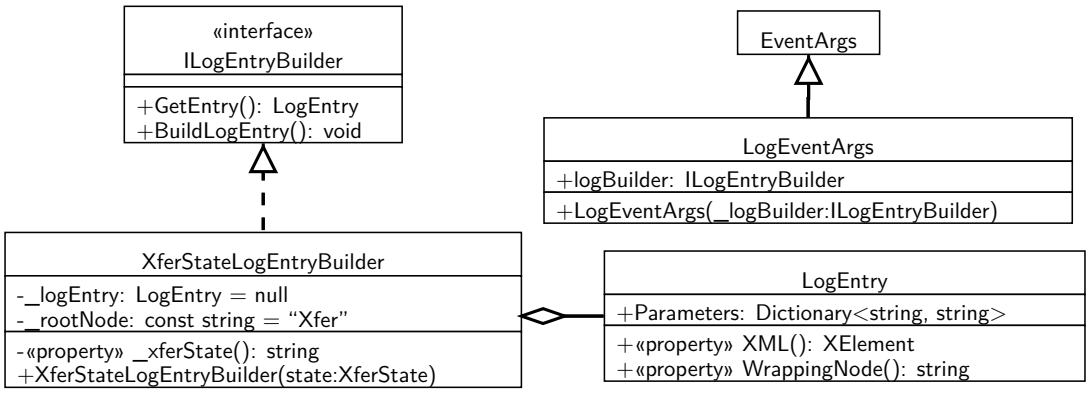
9.2 Logging code refactoren

De huidige code waarmee het logbestand gegenereerd wil ik refactoren. In het log event wordt nu een hard gecodeerde string meegegeven. Dit is niet gunstig, omdat je nu voor elke log entry een aparte string moet samenstellen. In de code zie je op diverse plekken string variabelen die er bijna identiek uitzien. Zo is er ook een sterke koppeling, omdat je wanneer je het wilt wijzigen de verantwoordelijke method moet opzoeken om de string daarin aan te passen.

Aangezien het hier gaat om steeds hetzelfde soort object op te leveren, namelijk een XML element in de vorm van een string, die verschilt aan de hand van de context en parameters, zag ik een oplossing in de vorm van de 'Builder' pattern. Het doel van de builder pattern is om de constructie van een complex object te scheiden van zijn representatie. Wanneer dit gescheiden wordt, kan door hetzelfde constructieproces verschillende representaties van hetzelfde object gebouwd worden.

Denk bijvoorbeeld aan het samenstellen van een kindermaaltijd bij een fastfood restaurant. Dit bevat altijd een hoofdgerecht, een bijgerecht, een drankje en een speelgoedje. De inhoud kan variëren, echter wordt de maaltijd altijd op dezelfde manier samengesteld door de medewerkers.

9.3 Klassendiagram na refactoren



Figuur 12: Builder pattern implementatie klassendiagram

In figuur 12 is een klassendiagram te zien van de builder pattern zoals ik hem in de scanner heb geïmplementeerd. In de builder pattern heb je voor elke vertegenwoordiging van een object dat je wil maken een eigen concrete implementatie. In dit klassendiagram laat ik dat met een enkele voorbeeld zien met de klasse 'XferStateLogBuilder'. Alle concrete implementaties van builder objecten implementeren de 'ILogEntryBuilder' interface en bevatten een 'LogEntry'

class. 'LogEventArgs' is aangepast om in plaats van een string een builder object mee te leveren als argument. Dit gebeurt weer met dependency injection.

Door deze constructie is er nu een lossere koppeling tussen de eerder in dit verslag beschreven ChipSoftDataSource en TwainLogger klassen. In plaats van een hard gecodeerde string wordt het 'Log' event nu afgevuurd door een concrete implementatie van een builder object mee te geven. In de argumenten van de constructor van deze objecten kunnen dan argumenten vanuit de bovenliggende methods worden meegegeven. In de concrete implementatie van het builder object is de code waarmee het gewenste XML element gegenereerd wordt ingekapseld. Dit zorgt voor een lossere koppeling. Het is nu ook makkelijk om nieuwe log XML elementen toe te voegen. Het enige wat je dan hoeft te doen is een nieuwe concrete implementatie van het builder object te maken. Dit heb ik dan ook gebruikt om de gewenste uitbreidingen in het logbestand toe te voegen, namelijk de gebruiker en werkstation en informatie over de virtuele scanner.

9.4 Log save knop

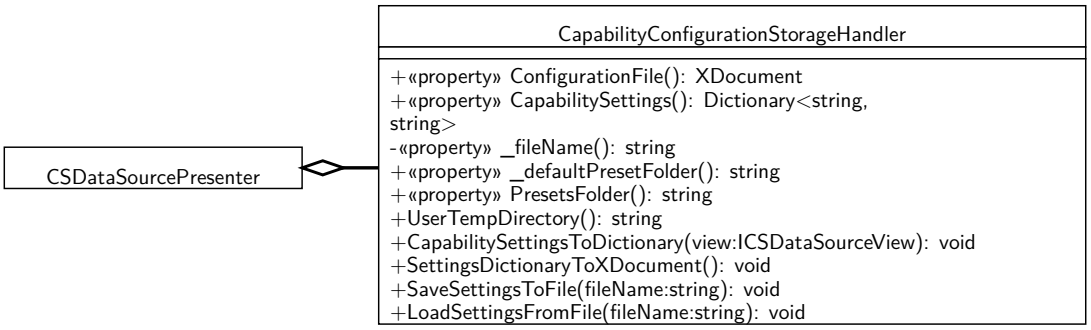
Na het raadplegen van de Microsoft developer network en diverse fora op internet bleek het mee te vallen om de save knop voor het log venster te ontwikkelen. Het dialoog wat ik wou gebruiken om dit te doen is een standaard 'Save file' dialoog van Microsoft. Het is een apart venster, maar wil toch graag in een enkele thread interactie hebben met het hoofdvenster van waaruit het word opgeroepen.

Om dit op te lossen heb ik een standaard 'Save file' dialoog toegevoegd aan de UI. Voorheen probeerde ik vanuit de code een nieuwe te creëren Daarna heb ik een method toegevoegd die het dialoog toont en wanneer de gebruiker op 'OK' klikt de tekst in het log tekstveld met LINQ to XML parset en opslaat met de in het bestandsnaamveld ingevulde bestandsnaam in de 'Save file' dialoog. Dit werkt omdat ik het tekstveld zo heb ontwikkeld dat het XML bestand wat automatisch wordt gegenereerd en opslaat spiegelt, zoals omschreven in sectie 7.4.

Dit werkt pas wanneer het in een single thread gebeurt. Het threading probleem zorgt ervoor dat het 'Save file' niet getoond of gecreëerd kan worden wanneer je dit vanuit het hoofdvenster opvraagt. OM dit probleem op te lossen heb ik nog een method toegevoegd die reageert op het klikken van de save knop. Binnen deze method wordt een nieuwe thread aangemaakt voor de method die ik in de vorige alinea omschreef. Vervolgens gebruik een method van Microsoft om de thread in een 'Single-Thread Apartment' te zetten en start ik daarna de thread. .NET zorgt er dan voor dat de nieuwe thread in hetzelfde thread terechtkomt en kan dan het venster openen. Het hoofdvenster is dan niet meer te gebruiken totdat het 'Save file' dialoog is afgesloten. Het moet wachten op het nieuwe venster omdat het nu in dezelfde thread zit.

9.5 Capability-configuraties opslaan

Het volgende item waar ik aan heb gewerkt is het opslaan van de huidige capability waardes en deze later weer herroepen. Hiervoor heb ik vier nieuwe elementen aan de UI toegevoegd, namelijk een selectiebox en drie knoppen. Met de drie knoppen hebben de labels 'Save', 'Load' en 'Delete'. Wanneer op de 'Save' knop wordt gedrukt, komt er een dialoog tevoorschijn met een tekstveld en een 'OK' knop. Wanneer op de ok knop wordt geklikt wordt de naam in het tekstveld meegegeven aan de scanner UI. Vervolgens gaat er in de view een event af die in de presenter een callback method uitvoert. Vanuit de callback worden de huidig ingestelde capability waardes opgeslagen in een XML bestand.



Figuur 13: CapabilityConfigurationStorageHandler klassendiagram uitbreiding

In figuur 13 is deze uitbreiding op het klassendiagram te zien. In de 'CapabilitySettings' dictionary hebben de keys dezelfde naamgeving als de properties in de view. In C# zijn properties attributen met getters en setters. Daarmee hoef je niet voor elk attribuut in je code aparte get en set methods te programmeren. Wel kun je de getters en setters zelf een eigen invulling geven indien dat gewenst of nodig is.

Doordat ik dezelfde naamgeving als de properties in de view hanteer in de keys van de dictionary, is het mogelijk om door middel van reflectie de waarde van de property aan te passen. Reflectie is het proces waarbij een computerprogramma op runtime niveau zijn eigen structuur en gedrag kan observeren en aanpassen. In de presenter heb ik een method ontwikkeld met een foreach loop erin. In de loop worden alle dictionary entries doorlopen en op basis van de naam van de keys kan ik dan de property in de view aanpassen. Op die manier hoef je dit niet hard to coderen en kun je het makkelijk uitbreiden, zolang de namen van de properties in de view en de keys in de dictionary matchen.

9.6 Capability-configuraties herroepen

De configuratie bestanden worden opgeslagen in de tijdelijke map van de gebruiker. Wanneer de scanner opstart wordt in deze map gekeken naar alle aanwezige configuratie XML bestanden.

De naam van elk bestand wordt toegevoegd aan de lijst in de selectiebox, benoemd in sectie 9.5. Wanneer hier een configuratie is geselecteerd en op de 'Load' knop wordt gedrukt, worden de configuraties en bijbehorende instellingen herladen. Dit is direct te zien in de UI van de scanner. Ook hier wordt gebruik gemaakt van de dictionary benoemd in sectie 9.5.

9.7 Retrospective

Wat ging goed?

- Tevreden Product Owner.
- Lossere koppeling vanwege combinatie van Builder en Observer design patterns in de logging functionaliteit.
- Capability configuratie functionaliteiten los gekoppeld en goed uitbreidbaar door het toepassen van reflectie.

10 Sprint 6 - Puntjes op de i's

In deze sprint heb ik nog een aantal capabilities geïmplementeerd. Het configuratiebestand opslag en herroep systeem wat ik in de vorige sprint heb ontwikkeld is de aanleiding tot het refactoren van de wijze waarop capability waardes in de scanner wordt uitgewisseld tussen objecten. De view is voorzien van een andere variant van de observer pattern, zodat het zelf meer verantwoordelijkheid over de eigen events.

10.1 Sprint planning

Tabel 16: Product Backlog Items voor Sprint 6

ID	Item	Prioriteit	Points
PBI6	Laatste capabilities uit inventaris implementeren	Hoog	8
PBI9	Wijze van capability waardes uitwisselen refactoren	Hoog	8
PBI10	View refactoren	Hoog	5

In tabel 16 worden de voor deze sprint gekozen Product Backlog items getoond. PBI6 is een restant uit voorgaande werk. Wanneer het autofeeden correct werkt, kan duplex scannen ook correct werken. Ik heb bij Product Owner aangegeven dat ik bepaalde delen van de code wil refactoren. Daar is hij mee akkoord gegaan.

Tabel 17: Sprint 6 Backlog

Item	Taken	ETC
PBI6	Autofeed capability implementeren	8
(Capabilities)	Duplex capability implementeren	12
PBI9	Data class toevoegen	4
(Wijze data	Methods vervangen om met de data class te werken	8
uitwisseling refactoren)	Code die met oude properties in View werkte aanpassen	8
PBI10 (View)	INotifyPropertyChanged gebruiken	8

In tabel 17 is de Sprint Backlog voor deze sprint te zien. Aan deze sprint is een budget van 48 uren toegekend.

10.2 Laatste capabilities

De laatste capabilities die ik implementeer voordat de scanner wordt opgeleverd, zijn de mogelijkheid tot het gebruiken van de automatische papierlader en duplex scannen. Duplex scannen is het scannen van beide kanten van een document. Dit kan op twee manieren, namelijk in één keer, of in twee keer. Duplex scannen in twee keer is simpelweg het document

of stapeltje documenten na het scannen er andersom in doen, en een tweede keer scannen voor de andere kant. Wanneer het in een enkele gebeurt, beschikt een scanner over twee camera's waarmee het beide kanten in een enkele keer kan scannen. De laatste manier is de manier waarop ik het in de virtuele scanner implementeer.

Duplex scannen werkt pas correct wanneer een scanner een werkende automatische papierlader heeft. Dit komt omdat een scanner in duplex mode, twee keer zoveel document scant dan zonder duplex. Dat betekent dus dat de tweede scan meteen automatisch achter de eerste scan komt, ook bij scanners die met twee camera's in een enkele keer scannen. Dit komt omdat het scannen nog steeds volgens het protocol één voor één scant.

Het implementeren van de autofeeder en daarbij behorende capabilities was niet heel lastig. Ik heb in de UI een 'NumericUpDown'component toegevoegd. Hiermee kan een getal met pijltjes verhoogd of verlaagd worden. Met dit component kan een gebruiker aangeven hoeveel documenten er gescand worden. Om dit te bewerkstelligen heb ik ook een eigen capability geïmplementeerd waarmee de scanner de tel kan bijhouden.

10.3 Duplex scannen

Duplex scannen was een iets lastiger vraagstuk. Aanvankelijk heb ik geprobeerd deze capability eerst te implementeren, tot ik erachter kwam dat een scanner hier een automatische papierlader voor nodig heeft. Nadat ik die capability had geïmplementeerd was ik er echter niet meteen. Het heeft enig onderzoek op fora en verdieping in de TWAIN specificatie gevergd voordat ik dit werkend kreeg.

Ik heb het ontwikkeld door ten eerste logica in de code te bouwen die altijd het aantal scans verdubbelt wanneer de scanner in duplex-modus scant. Nadat dit functioneel was heb ik in de method waarin het beeldbestand wordt opgehaald en bewerkt wordt voor uitvoer met een modulus operatie gekeken of het aantal resterende scans een even getal is. Wanneer dit niet het geval is, hebben we met de achterkant te maken en wordt het beeldbestand verticaal omgedraaid. Op die manier is het voor gebruikers te valideren welke van de opgeleverde beeldbestanden de achterkanten zijn.

10.4 Dataklasse

De functionaliteiten om capability configuraties op te slaan en te laden werken op dit moment alleen als de UI zichtbaar is. Er is daarnaast op meerdere plekken in de code een sterke koppeling tussen properties in de view en andere objecten die interactie hebben met de view. De Product Owner wilt ook zonder zichtbare UI gebruik kunnen maken van de functionaliteit om configuraties op te slaan en te laden. Naar aanleiding hiervan heb ik besloten om de harde koppeling die hierdoor ontstaat met de view te refactoren.

Dit heb ik gedaan door alle properties uit de view te verplaatsen naar een eigen object, namelijk de ScanSettings klasse. Vervolgens heb ik op alle plaatsen in de code waar direct in

de view properties werden aangepast, de code gewijzigd om gebruik te maken van het nieuwe object. Er waren een aantal methods en events die ook direct gekoppeld waren aan view properties die ik ook allemaal heb aangepast om te werken met het nieuwe object.

Nadat ik had geverifieerd dat de aangepaste code werkte met het nieuwe object, heb ik 'CapabilityConfigurationStorageHandler' aangepast om gebruik te maken van het nieuwe object. In plaats van het meegeven van de view aan de method waarmee de waardes vanuit de view in de dictionary worden geplaatst, wordt er nu een event afgevuurd waarin een ScanSettings object wordt meegegeven als argument. Overigens gebruik ik op de andere plekken waar ik aanpassingen heb gemaakt om met het nieuwe object te werken soortgelijke constructies met events.

10.5 View refactoren

Bij het werken met bepaalde capabilities, zoals het kiezen van voorgedefinieerde papierformaten, werden bepaalde waardes in de view aangepast om dit te reflecteren. Denk hierbij bijvoorbeeld aan het wijzigen van de lengte en breedte van het te scannen papier. Dit werd tot nu toe door de presenter geregeld. Bij een wijzigingen gaat er in de view een event af die een callback method in de presenter uitvoert, die dan vervolgens in de view weer bepaalde waardes aanpast.

Na het verwijderen van de properties in de view moest ik dit ook refactoren. Ik kon ze niet meer vanuit de presenter aanpassen. Microsoft heeft in .NET een interface beschikbaar waarmee clients genotificeerd kunnen worden dat een property waarde is gewijzigd. Deze interface, 'INotifyPropertyChanged' genaamd, implementeer ik in de view. Vervolgens kun je de property van een gewenste component koppelen, of 'binden', aan een andere waarde. Je kunt bijvoorbeeld een checkbox component binden aan een boolean. Wanneer die binding er is zal de checkbox in de view aan of uit gevinkt worden wanneer de boolean respectievelijk true of false is.

Hier licht ik met een voorbeeld toe hoe ik dit heb geïmplementeerd in de scanner UI. Wanneer de gebruiker kiest voor 'None' in plaats van een standaard papierformaat, vertegenwoordigt door de capability ICAP_SUPPORTEDSIZES, is het de bedoeling dat de tekstvelden met de lengte en breedte van het te scannen oppervlak bewerkbaar worden zodat een gebruiker daar iets kan invullen. Bij standaardpapierwaarden moeten deze velden niet bewerkbaar zijn en staan de voorgedefinieerde waardes erin. Tekstvelden hebben een property 'ReadOnly' waarmee bepaald wordt of een een gebruiker een tekstveld kan bewerken of alleen kan lezen.

De 'ReadOnly' property van de tekstvelden waar lengte en breedte worden ingevuld heb ik gebonden aan een boolean in de view, namelijk 'CustomSizeEnabled'. Wanneer een gebruiker een ander papierformaat selecteert, gaat er in de view een event af die een callback method in de presenter uitvoert. Wanneer een gebruiker hier 'None' kiest, stelt de presenter de boolean 'CustomSizeEnabled' in op false en gaat er een event af, namelijk PropertyChanged. Aan de hand van het event weet de view dat een property is veranderd en wordt de waarde van de gebonden component veranderd aan de hand van de waarde waar het aan is gebonden. In dit

geval verandert de property 'ReadOnly' van true naar false, wat betekent dat de gebruiker nu zelf in de tekstvelden een lengte en breedte kan invullen.

10.6 Retrospective

Wat ging goed?

- Product Owner tevreden.
- Klassen losser gekoppeld.
- Verantwoordingen view meer ingekapseld.

Wat ging minder goed?

- Teveel tijd besteed aan het het implementeren van Duplex scannen.

11 Sprint 7 - Oplevering

Deze sprint is gewijd aan het opleveren van de virtuele scanner die ik heb ontwikkeld. Ik gebruik de scanner vanuit HiX en test op exploratieve wijze of het hier goed in werkt met behulp van de debugger en enkele TWAIN gerelateerde programma's. Het bleek in eerste instantie niet meteen te werken met HiX, maar dat probleem heb ik verholpen en het eindproduct aan de Product Owner opgeleverd.

11.1 Sprint planning

Deze sprint heeft het doel het eindproduct op te leveren in een form waarmee het in HiX werkt. De user stories US6 en US7 met betrekking tot het laden en opslaan van beeldbestanden, te zien in tabel 5 in sectie 5.1, worden niet meer opgeleverd omdat daar geen tijd meer voor is. Wel is US6 deels opgeleverd, omdat je het plaatje wat door de scanner wordt geladen kunt veranderen door deze te kopiëren in de map waar de scanner is geïnstalleerd en dezelfde bestandsnaam te geven als het daar aanwezige beeldbestand. Aan deze sprint is een budget van 32 uur toegekend. In tabel 18 is de Sprint Backlog te zien.

Tabel 18: Sprint 7 Backlog

Item	Taken	ETC
Opleveren	Werking in HiX verifiëren	12
	Aanpassingen maken in code om werkend te maken	16
	Product op netwerk beschikbaar stellen	4

11.2 Werking in HiX verifiëren

Tijdens een vorige sprint bleek het dat de virtuele scanner niet correct werkte in HiX. Dit vond ik vreemd, omdat de scanner vanuit elke applicatie die ik er tot nu toe mee heb gebruikt een correcte en verwachte werking hadden. Wanneer in HiX een voorbeeld werd opgevraagd, werkte het prima. Zodra we probeerden 'echt' te scannen crashte HiX.

Om te achterhalen wat hier de oorzaak van is heb ik de debugger gebruikt. Met een breakpoint op de method van waaruit de scan wordt geïnitialiseerd heb ik de code stap voor stap laten uitvoeren tot ik de crash tegenkwam. Hierdoor kon ik achterhalen waar het fout liep en was ik een stapje dichterbij de oorzaak.

Er was wel een probleem, HiX bleef crashen en daardoor kon ik niet verder met behulp van de debugger achterhalen waar het dan precies misging. Ik was namelijk aan het debuggen door de debugger te koppelen aan HiX als draaiende applicatie. Hiervoor gebruikte ik alleen de code die betrekking had op multimedia. Het zou namelijk vrij onwerkbaar zijn voor dit

probleem om met de gehele codebase te runnen en te debuggen. HiX is heel groot en dat zou te veel tijd kosten.

Aanvankelijk dacht ik dat het probleem zat in een bepaalde capability die ik niet in de scanner werkend kreeg, namelijk ICAP_FRAMES. Leadtools maakt gebruik van twee capabilities om vanuit de applicatie het gewenste formaat af te spreken met de source. De andere capability waar het hier om gaat is ICAP_SUPPORTEDSIZES. Dit zijn beide niet standaard capabilities waarmee in TWAIN het gewenste scanformaat afgesproken wordt.

Na bespreking met de Product Owner hebben we een aanpassing gemaakt in HiX waarmee het gebruik van deze capability omzeild wordt. Dit leverde een DLL op voor de multimedia-component die ik in de map van HiX kon kopiëren. Hierna werkte het scannen en leek het probleem even opgelost. Het zat me echter niet lekker. Een fysieke scanner hier op kantoor, die ook ICAP_FRAMES niet ondersteunt, werkt wel in HiX. Dit heb ik na deze aanpassing gecontroleerd om de constatering dat deze capability het probleem was te ondersteunen. Het lijkt er dan op dat ICAP_FRAMES niet verantwoordelijk is voor het crashen.

11.3 Verdere aanpassingen

Naar aanleiding van voorgaande besloot ik te proberen om ICAP_SUPPORTEDSIZES beter te implementeren. Deze capability had ik al geïmplementeerd, en was functioneel, alleen paste de scanner niet intern het gewenste scanformaat aan na het kiezen van een standaard papierformaat. Wel gebeurde dit in de view, en bij het klikken op de scan knop werden de waardes hieruit opgehaald. Dat werkte goed en levert het gewenste formaat op.

Om dit probleem op te lossen heb ik de scanner aangepast om ook intern meteen na het kiezen van een papierformaat het gewenste scanformaat aan te passen in plaats van alleen in de view. Na dit getest te hebben besloot ik weer te proberen vanuit HiX te scannen. Ik gebruikte hiervoor de originele multimediacomponent DLL zonder de hiervoor beschreven aanpassing. Weer crashte HiX.

11.4 TWAIN DSM logging functionaliteit

De virtuele scanner beschikt ook over logging functionaliteit. Alleen is het op dit moment zo dat dit verloren gaat wanneer de applicatie die er gebruik van maakt crasht. De UI verdwijnt dan, waardoor ik het daar niet kan inzien, en het logbestand wordt pas gegenereerd als de scanner 'netjes' wordt afgesloten. Hier had ik helaas geen rekening mee gehouden bij het ontwikkelen van deze functionaliteit.

De 'Data Source Manager' van TWAIN, standaard aanwezig op elk Windows station waar een scanner op is geïnstalleerd, is voorzien van functionaliteit waarin een log van TWAIN activiteit te zien is. Om deze log in te zien, moet je in de Windows omgeving een environment variabel aanmaken waarin je een map en bestandsnaam kan specificeren. Daar wordt dan bij

elk gebruik van de data source manager een log aangemaakt. Van deze functionaliteit heb ik overigens gedurende het hele project veelvuldig gebruik gemaakt.

De log kapte steeds na dezelfde TWAIN operatie af, vermoedelijk het punt waar HiX telkens crasht. De triplet waar het ophield was een instructie om een bestandsoverdracht te configureren. Met dit type overdrachtsmechanisme kunnen een applicatie en TWAIN source een bestandsnaam en bestandstype afspreken om de data in op te slaan. In deze overdrachtsmodus wordt het bestand eerst aangemaakt op de pc, om deze vervolgens te vullen met data na overdracht.

11.5 Laatste aanpassingen

Op basis van deze informatie besloot ik om in de scanner de ondersteuning voor dit type overdrachtsmechanisme uit te schakelen zodat de scanner het niet meer ondersteunt. Na deze aanpassing in de scanner werkte de scanner in HiX. Dit was kennelijk de oorzaak van het crashen. Het vreemde hiervan is dat er bij geen enkele instelling in de scanner of HiX gevraagd werd om gebruik te maken van het bestandsoverdrachtsmechanisme.

Toen ik dit had bevestigd heb ik de bestandsoverdrachtsmodus weer ingeschakeld en de configuratie triplet hiervoor getest in Twacker. Nadat ik hierin probeerde een bestandsnaam in te vullen werkte het niet. Twacker gaf aan dat dit niet werd ondersteunt. Hierna heb ik de scanner aangepast om dit wel functioneel te maken. Dit heb ik gedaan in de method 'OnSetpFileXfer', één van de methods beschreven in sectie 6.3 die op basis van TWAIN operaties worden uitgevoerd. Ik heb de correcte werking hiervan gevalideerd met behulp van Twacker.

Na het opnieuw proberen te scannen in HiX crashte het nog steeds. Volgens de log ook nog steeds op dezelfde plaats. Dit probleem heb ik bij de Product Owner aangekaart en dit heeft ons doen besluiten om het bestandsoverdrachtsmechanisme achterwege te laten. Dit overdrachtsmechanisme stamt nog uit de tijd dat PC's weinig geheugen tot hun beschikking hadden en wordt niet meer in de praktijk gebruikt. Zodoende was het niet nodig om dit in de scanner te ondersteunen, zeker als dat betekent dat de scanner werkt in HiX.

11.6 Oplevering

Ik heb de install package op een netwerk schijf geplaatst waar het voor iedereen toegankelijk is. De Product Owner vroeg of ik er eventueel instructies bij wou doen, met betrekking tot het installeren van de scanner en de locaties van bijbehorende bestanden, zoals het beeldbestand. Dit heb ik geregeld vanuit de install package, dus de instructies waren niet nodig.

Het bleek dat de install package niet goed werkte op andere machines. De installer klaagt over een ontbrekende DLL en kapt af. Later ontdekte ik dat het wel werkt wanneer eerst een ander pakket van Saraff.Twain.DS geïnstalleerd wordt. Dit heeft te maken met een bepaalde DLL die systeem-breed geïnstalleerd moet worden. Ik kon niet ontdekken waarom het nu niet

precies werkte, maar de sprint en beschikbare tijd is voorbij. ChipSoft heeft me in dienst genomen, dus ik zal in de toekomst hier verder aan kunnen werken.

11.7 Retrospective

Wat ging goed?

- ICAP_SUPPORTEDSIZES capability nu correct geïmplementeerd.
- Tevreden opdrachtgever.

Wat ging minder goed?

- Install package bleek niet helemaal juist te werken. Dat had eerder ontdekt kunnen worden.

12 Evaluatie

Dit hoofdstuk is een evaluatie van de producten en uitvoering van de afstudeeropdracht. Eerste evalueer ik de producten. Vervolgens wordt het proces wat ik heb doorlopen geëvalueerd. Tot slot evalueer ik in tot hoeverre voldaan is aan de gekozen beroepscompetenties.

12.1 Producten

Adviesrapport
Er is een beperkt aanbod van software met virtuele scanners te vinden. De keuzes vielen tussen commerciële virtuele scanners, open source scanners die niet voldoen aan de eis om te ontwikkelen in C# en een C# .NET class library waarmee naar wens een eigen TWAIN source ontwikkeld kan worden. Bij het kiezen van één van de commerciële pakketten word je in de scanner functionaliteit beperkt door wat de ontwikkelaar ervan heeft geïmplementeerd en moet het gewenste eindproduct eromheen ontwikkeld worden. Daar is een goede API met interfaces wenselijk in. Ik zie het verschil in kiezen tussen de commerciële pakketten en de class library een beetje als de keuze tussen het uitbreiden van een goedkope magnetronmaaltijd (de commerciële pakketten) en het zelf koken van een maaltijd met kwalitatieve ingrediënten (de class library). De open source scanners zijn voor ChipSoft niet overdraagbaar. Het was zodoende voor duidelijk dat er een enkele keuze uitsprong voor het ontwikkelen van het eindproduct. Hier heb ik de opdrachtgever ook van kunnen overtuigen met mijn bevindingen. Dit is voor mij een reden om dit product als goed te beoordelen.

Ontwerpdiagrammen
Het opstellen van ontwerpdiagrammen heeft geholpen bij het op een abstract niveau vaststellen of er goede ontwerpkeuzes gemaakt zijn. Het proces wat hoort bij op een abstract niveau van tevoren nadenken over het ontwerp van software baat bij de ontwikkeling hiervan. Ook kan er later naar gerefereerd worden wanneer je in de code iets vergeten bent of niet meer goed begrijpt. Daarbij is het ontwerp van het systeem duidelijker over te brengen aan derden, zoals de Product Owner of collega ontwikkelaars. Zonder in de code te kijken wordt het duidelijk welke klassen er zijn en welke samenhang ze hebben. Ik heb me gehouden aan de UML standaard en heb aan de hand van de diagrammen die ik heb opgesteld kunnen verduidelijken aan derden hoe ik het systeem heb ontworpen. Daarbij levert het een stukje documentatie op wat bij het eindproduct hoort. Dit is gunstig voor toekomstige derde partijen die ermee te maken krijgen.

Virtuele scanner
De scanner die ik heb ontwikkeld voldoet aan alle globale eisen zoals in het begin opgesteld, te lezen in sectie 3.3. Aan alle functionele eisen zijn voldoen. Het overschrijdt niet de technische beperkingen. Het eindproduct voldoet ook aan de gewenste kwaliteitsnormen, vastgelegd in de niet-functionele requirements in sectie 3.3. Ik heb wel twee user stories met een lage

prioriteit achterwege gelaten. Dit zijn US6 e US7, te zien in tabel 5 in sectie 5.1. Het beeldbestand die door de scanner wordt geleverd voldoet voor testdoeleinden en met behulp van applicaties is het mogelijk beelden op te slaan voor verificatiedoeleinden. Het achterwege laten van deze user stories leidt daarom niet tot een ontevreden opdrachtgever.

De opdrachtgever heeft aangegeven tevreden te zijn met het eindproduct. De capabilities vanuit HiX, waarvan de inventaris te zien is in sectie 7.6, worden allemaal in de scanner ondersteunt. De opdrachtgever geeft aan dat het opgeleverde eindproduct bruikbaar is en in kan worden gezet bij het testen van HiX. Er moeten nog wel dingen aan gebeuren. Aangezien de afstudeeropdracht heeft geresulteerd in een baan voor mij bij ChipSoft, kan gesteld worden dat de opdrachtgever tevreden is over het opgeleverde eindproduct.

12.2 Proces

Over het proces ben ik tevreden. Door het gebruik van scrum vond er veel communicatie plaats met de opdrachtgever. Ik kreeg snel feedback en er was steeds voldoende ruimte en mogelijkheid om wijzigingen in de planning aan te brengen. Het is fijn om in vaste time boxes te werken en hierin concrete werkzaamheden of taken te hebben. Met behulp van het urenbudget kon ik in de gaten houden of taken niet teveel tijd in beslag nemen. Ik zal in het vervolg zeker weer gebruik maken van scrum. Verbeterpunten zijn het beter schatten van werkzaamheden en hierop inspelen in de planning en tijdens de uitvoering. Het was wel heel lastig om werkschattingen te maken, aangezien ik deze werkzaamheden in combinatie met deze technologieën niet eerder verricht heb. Ook zou ik verbeteringen kunnen aanbrengen in het vooraf in grote lijnen bepalen wat er gedurende het project opgeleverd moet worden. Scrum biedt voldoende mogelijkheid en ruimte om hier op in te spelen en afspraken over te maken.

Het werken met test-driven development vind ik ook een succes en een fijne ervaring, alhoewel ik dit niet altijd kon uitvoeren zoals het officieel hoort. Dit komt omdat ik veel te maken had met third-party code en API's. Dit is een beperkende factor voor een correcte uitvoering van test-driven development, omdat er gewerkt moet worden met al bestaande code. Hierdoor is het lastig om unit tests te schrijven voor test cases gebaseerd op user stories, omdat de onderliggende third-party code al bepaalde code voorschrijft of vereist voor een correcte werking. Je bent dan meer bezig met integratie tests en het belemmert het ontstaan van een fris ontwerp. Test-driven development is een ontwikkelmethode die ik in de toekomst vaker wil gebruiken. Een verbeterpunt is het duidelijker koppelen van testcases aan specifieke requirements die gevalideerd kunnen worden.

Het onderzoeken van de TWAIN specificatie en het selecteren van software om binnen de opdracht toe te passen heeft het geholpen bij het definiëren van het gewenste eindproduct. Tijdens dit onderzoek had ik veelvuldig contact met de opdrachtgever om mijn bevindingen te bespreken. We kwamen hierdoor steeds meer op dezelfde lijn terecht met onze visies. Aan het begin van het project kwam ik wat moeilijker op gang, omdat er voor mij nog veel onduidelijk en onbekend was. Niet alleen op het gebied van de eisen aan het eindproduct, maar ook

de technologieën waar ik mee aan de slag moest. Ik vond het lastig om de probleemstelling te formuleren en om concrete wensen en eisen vast te stellen. Nadat ik me had verdiept in de techniek kon ik beter draagvlak creëren met de opdrachtgever. Dit proces heeft veel bijgedragen aan het succesvol opleveren van een eindproduct waar de opdrachtgever tevreden over is.

Bij het ontwikkelen van de scanner had ik weleens de neiging om te complex denken over oplossingen. Wat me hierbij heeft geholpen is proberen te denken aan principes die vaak gepaard gaan met test-driven development, zoals 'Keep it simple, stupid!', 'You ain't gonna need it' en 'Don't repeat yourself'. Het is me wel telkens gelukt om hier overheen te komen en toch een mooi resultaat neer te zetten. Ik ben blij met de gemaakte ontwerpkeuzes en toegepaste design patterns. Tijdens de opleiding heb ik veel geleerd over degelijk objectgeoriënteerde software en het ontwikkelen daarvan. Hier kon ik veel van toepassing tijdens het ontwikkelproces en het was leuk om bepaalde concepten succesvol in de praktijk te brengen. Het geeft een goed gevoel om een zelf ontwikkeld product op te leveren dat aan bepaalde kwaliteiten en principes voldoet en een nuttige bijdrage gaat leveren in een bedrijfsproces.

12.3 Beroepstaken

Hier evalueer ik in hoe en waar voldaan is aan de beroepstaken. De tekst uit de beroepstaken zijn cursief weergegeven, met daaronder mijn verantwoording.

1.3 Selecteren van standaardsoftware (niveau 3)

Adviseren over en/of selecteren van software, zoals een applicatie (commerciële software of open source), framework (al dan niet gebaseerd op open standaarden), databasemanagementsysteem of besturingssysteem.

Het uitgebrachte advies over de door mij gebruikte class library is vastgelegd in het adviesrapport. Het proces wat heeft geleid tot het advies, samen met de verantwoording hiervan, is beschreven in sectie 4. Het advies is onderbouwd door me te verdiepen in de technische aspecten, zodat ik geschikte software kandidaten kon toetsen aan de selectiecriteria.

3.2 Ontwerpen systeemdeel (niveau 3)

Beschrijven van systeemdelen (subsystemen, componenten, modules), hun onderliggende structuur en het gedrag in detail, zodanig dat bouwen van het systeemdeel mogelijk is.

Ontwerpen (grafische) user interface.

De structuur en het gedrag is beschreven met behulp van klassendiagrammen en sequentiediagrammen. De onderliggende relaties zijn duidelijk, en de klassen bevat genoeg detailinformatie zodat nabouwen mogelijk is. Ik heb een ontwerpmethodiek toegepast en heb rekening

gehouden met toekomstige wijzigingen, testbaarheid en hergebruik. Er is gebruik gemaakt van design patterns.

Onderdeel van het eindproduct is een grafische UI. Deze heb ik zelf ontworpen, rekening houdend met de wensen van gebruikers om alle scaninstellingen toe te kunnen passen.

3.3 Bouwen applicatie (niveau 3)

Verfijnen en/of transformeren van het (detail)ontwerp van de systeemdelen (subsystemen, componenten, modules) van een applicatie.

Bouwen en documenteren van de systeemdelen.

Systeemdelen samenstellen tot een werkende applicatie.

Ik heb mijn eigen ontwerpen aangesloten op een al bestaand ontwerp in de gekozen class library en op basis hiervan het eindproduct gebouwd. Hierbij heb ik geavanceerde concepten, zoals reflectie, van de gebruikte programmeertaal toegepast. Er is rekening gehouden met toekomstige wijzigingen, testbaarheid en hergebruik. Het eindproduct is gebouwd volgens een complexe protocol en ik heb er een externe SDK in verwerkt. De ontwikkelomgeving waarin dit plaatsvond was voorzien van een versiebeheertool. Ik heb een testframework gebruikt bij de ontwikkeling om test-driven development te kunnen toepassen.

3.5 Uitvoeren van en rapporteren over het testproces (niveau 3)

Opstellen testscript, waaronder het specificeren van de testgegevens, de benodigde startsituatie en de uitvoerverwachtingen per testgebied. Uitvoeren testplan en opstellen rapportage.

Het bedenken en uitvoeren van unit- en integratietesten heeft de kwaliteit van de code het eindproduct gewaarborgd. Ik heb testcases bedacht waarvan de startsituaties en uitvoerverwachtingen het ontwikkelproces heeft gestuurd. Waar het mogelijk was heb ik ze gebaseerd op verifieerbare requirements of user stories. Hier heb ik testscripts voor opgesteld in de vorm van unit- en integratietests. Ik heb om dit te rapporteren het afstudeerverslag voorzien van enkele voorbeelden hiervan. Ook zijn ze bij de software opgeleverd.

Referenties

[1] *ISO/IEC 9126-1:2001*. adres: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=22749.

[2] *TWAIN Specification*, Version 2.4, TWAIN Working Group, dec 2015. adres: http://www.twain.org/wp-content/uploads/2017/03/TWAIN-2.4-Specification.pdf.

[3] *Fibonacci Sequence - History*. adres: http://science.jrank.org/pages/2705/Fibonacci-Sequence-History.html.

[4] Saraff, *Minimal Implementation with a Saraff.Twain.DS.BitmapSource*, Version 3, sep 2016. adres: https://twainds.codeplex.com/wikipage?title=Minimal%20Implementation%20with%20a%20Saraff.Twain.DS.BitmapSource.

[5] ——, *The framework of a Source*, Version 3, jul 2016. adres: https://twainds.codeplex.com/wikipage?title=The%20framework%20of%20a%20Source.

[6] ——, *Tree of directories*, Version 2, sep 2016. adres: https://twainds.codeplex.com/wikipage?title=Tree%20of%20directories.

[7] ——, *How create a Windows Installer package (msi)*, Version 5, sep 2016. adres: https://twainds.codeplex.com/wikipage?title=How%20create%20a%20Windows%20Installer%20package%20%28msi%29.

[8] M. Fowler, *Passive View*, jul 2006. adres: https://martinfowler.com/eaaDev/PassiveScreen.html.

[9] Saraff, *The list of implemented capabilities*, Version 4, aug 2016. adres: https://twainds.codeplex.com/wikipage?title=The%20list%20of%20implemented%20capabilities.

[10] ——, *Adding new capabilities*, Version 6, aug 2016. adres: https://twainds.codeplex.com/wikipage?title=Adding%20new%20capabilities.

Woordenlijst

API

Een application programming interface is een verzameling definities op basis waarvan een computerprogramma kan communiceren met een ander programma of onderdeel (meestal in de vorm van bibliotheken).. 3, 4, 7, 9, 12, 13, 16, 17, 20, 22, 27, 54, 55

capability

Een TWAIN capability is een optie of instelling waar een acquisitieapparaat over beschikt. Met andere woorden, capabilities vormen het vermogen van wat een acquisitieapparaat allemaal kan doen.. iv–vi, 11, 12, 15, 16, 18, 31, 34–41, 44–48, 51, 53, 55, 63

class library

Een verzameling klassen en code die door een ontwikkelaar gespecificeerd en gebruikt kunnen worden bij het ontwikkelen van een applicatie.. 1, 16, 17, 22, 23, 26, 54, 56, 57, 62

Daily Scrum

Dagelijkse vergadering van maximaal 15 minuten waarbij de voortgang van de sprint wordt besproken. Hoofdvragen: Wat is er gedaan? Wat wordt gedaan? welke knelpunten zijn er?. 6

dependency injection

Een geavanceerd ontwerppatroon uit de informatica dat het mogelijk maakt klassen losjes te koppelen. Dit wil zeggen dat ze data kunnen uitwisselen zonder dat deze relatie hard (in de broncode) vastgelegd is; althans niet door de programmeurs van die (beide) klassen.. 9, 32, 43

design pattern

Een design pattern is een generiek opgezette softwarestructuur, die een bepaald veelvoorkomend type software-ontwerpprobleem oplost. Het patroon geeft geen concrete oplossing, maar biedt een soort sjabloon, waarmee het ontwerpprobleem kan worden aangepakt.. 28, 61

DLL

Een dynamic-link library, ook wel bekend als DLL, is een bibliotheek met functies, die door meerdere applicaties gebruikt kunnen worden. Het is het tegenovergestelde van een statisch gekoppelde bibliotheek, waarbij de bibliotheek in elk programma dat de bibliotheek gebruikt moet worden ingebouwd.. 14, 15, 23, 26, 51, 52

elektronisch patiëntendossier

Een softwaretoepassing waarbij medische patiëntengegevens in digitale vorm bewaard en beschikbaar gemaakt worden.. 1, 2

entry point

Een entry point is de plaats in een computerprogramma waar de controle door het besturingssysteem wordt overgedragen aan het programma en de processor met uitvoering begint.. 14, 23, 26

event

Een event is een actie of gebeurtenis in software, waarop gereageerd kan worden door de software. Ze kunnen gegenereerd worden door het systeem of een gebruiker.. 26, 28, 29, 33, 38, 42–44, 46, 48

kleurendiepte

Kleurendiepte is een maat voor de hoeveelheid bits die gebruikt worden om de kleur van een pixel te coderen.. 3

method

In het objectgeoriënteerd programmeren is een method een synoniem voor functie. Het is dus een functie die behoort tot een klasse.. 26–29, 32, 33, 42–44, 46, 48, 50, 52

MVP

Model-View-Presenter is een gebruikersinterface softwarearchitectuur pattern dat ontworpen is om geautomatiseerde unit tests te faciliteren en een verbeterde 'seperation of concerns' in de presentatielogica te waarborgen.. 25, 26

Product Backlog

Een overzicht van de dingen die nog gedaan moeten worden. Dit wordt in andere software ontwikkelmethoden wel de requirements genoemd, de eisen en wensen. Het bestaat uit alles wat van belang is om mee te nemen in de sprint zoals eigenschappen, fouten uit vorige releases, niet functionele eisen zoals navigatie, kleur, "look en feel", snelheidseisen etc. De product owner is eigenaar van deze lijst en bepaalt de volgorde.. iv, 5, 6, 18, 19, 25, 31, 37, 40, 41, 46

Product Owner

De Product Owner (producteigenaar) is de opdrachtgever of klant. Hij beheert ook de product backlog, hij bepaalt wat er moet gebeuren en in welke volgorde.. 5–8, 10, 18–20, 22, 24–26, 30, 31, 33, 34, 37–40, 45–47, 49–52, 54

refactoren

Refactoren (Engels: refactoring) is het herstructureren van de broncode van een computerprogramma met als doel de leesbaarheid en onderhoudbaarheid te verbeteren

of het stuk code te vereenvoudigen. Het refactoren van broncode verandert de werking van de software niet: elke refactorstap is een kleine, ongedaan te maken stap die de leesbaarheid verhoogt zonder de werking aan te passen.. v, vi, 22, 28, 41, 42, 46–48

retrospective

Een evaluatie waarbij het doel is vast te stellen wat goed en fout ging bij het uitvoeren van een sprint.. v, vi, 6, 24, 30, 35, 40, 45, 49, 53

Scrum

Een flexibele manier om (software)producten te maken. Er wordt gewerkt in multidisciplinaire teams die in korte sprints, met een vaste lengte van 1 tot 4 weken, werkende (software) producten opleveren. Samenwerking is heel belangrijk en men moet snel kunnen inspelen op veranderende omstandigheden.. 5

Scrum Master

De Scrum Master begeleidt en helpt het team door ervoor te zorgen dat het juiste scrum-proces gevolgd wordt.. 6, 8

SDK

Een software development kit (vaak afgekort tot SDK) is een verzameling hulpmiddelen die handig zijn bij het ontwikkelen van computerprogramma's voor een bepaald besturingssysteem, type hardware, desktopomgeving of voor het maken van software die een speciale techniek gebruikt.. 10, 13, 20, 38, 39, 57

softwarearchitectuur pattern

Een generieke en herbruikbare oplossing voor een veelvoorkomend probleem in softwarearchitectuur, in een bepaalde context. Ze hebben een bredere scope dan design patterns.. 26, 60

solution

Een solution in Visual Studio is een groepering van één of meerdere projecten die samen een applicatie vormen.. iv, 18–20

sprint

Een periode van 1 tot 4 weken waarin men een aantal zaken van de productbacklog oppakt. Dit is een time-box.. iv–vi, 6, 7, 9, 10, 18, 19, 23, 25, 31, 37, 41, 46, 50

Sprint Backlog

In de sprint backlog staat wat er deze sprint wordt gedaan. De sprint backlog wordt samengesteld uit de items met de hoogste prioriteit van de product backlog. . iv, 19, 25, 37, 46, 50

story point

Worden gebruikt in planning om de omvang van een klus (story) aan te geven. Door een bekende korte klus 1 of 2 punten te geven kunnen grote klussen ingeschat worden als bijvoorbeeld driemaal zo lang of tienmaal zo lang.. iv, 19, 31, 37

test-driven development

Een ontwikkelmethode voor software waarbij eerst tests worden geschreven en daarna pas de code.. 6, 9, 10, 20, 21, 25, 27, 38, 55–57

TWAIN

De bekendste industrie-standaard om data via een scanner in een computer in te lezen. TWAIN is een API die grafische applicaties de mogelijkheid biedt om de scanner aan te sturen en de gescande beelden schijnbaar automatisch te importeren.. iv–vi, 3, 4, 7, 9–23, 25, 26, 31, 34, 35, 38–40, 47, 50–52, 54, 55, 59

UI

De gebruikersinterface, van het Engelse user interface (UI), is de interface (intermediair) tussen een computer (of andere machine) en de mens die de computer gebruikt. De gebruikersinterface maakt interactie tussen mens en machine mogelijk.. v, 12, 16, 17, 20, 21, 25, 26, 29–35, 41, 43–45, 47, 48, 51, 57

user story

Een user story is een korte beschrijving (story) van wat een gebruiker (user) wil.. 18, 50, 54, 55, 57

WIA

Windows Image Acquisition is een Microsoft driver-model en API voor Microsoft Windows-besturingssystemen die het grafische software mogelijk maakt om met beeldverwerkingshardware te communiceren, zoals scanners, digitale camera's en digitale video-apparatuur.. 3, 7, 9, 12, 38

WinForms

Windows Forms (WinForms) is een grafische class library dat onderdeel is van het Microsoft .NET framework.. 25

WPF

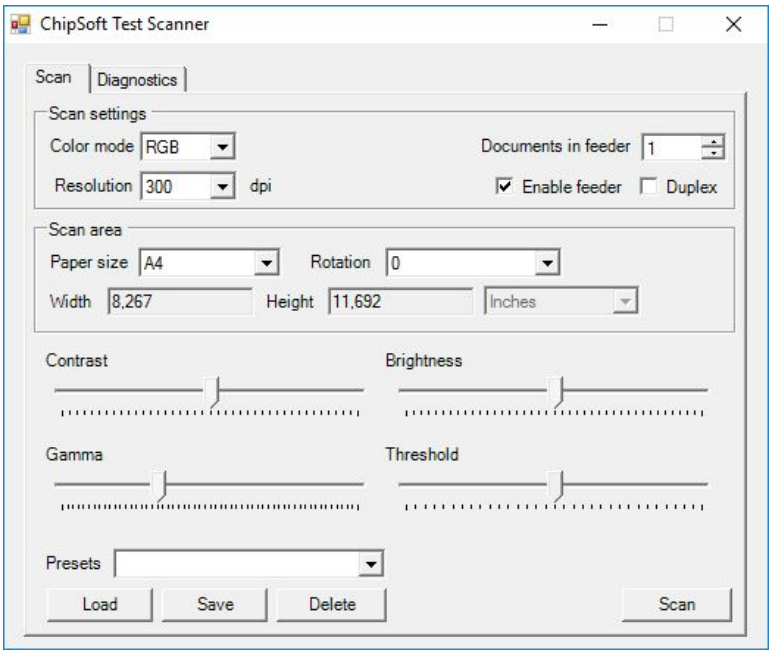
De Windows Presentation Foundation (of WPF) is het grafische subsysteem dat een onderdeel is van het .Net Framework versie 3 van Microsoft.. 20, 21, 25

A Capability inventaris

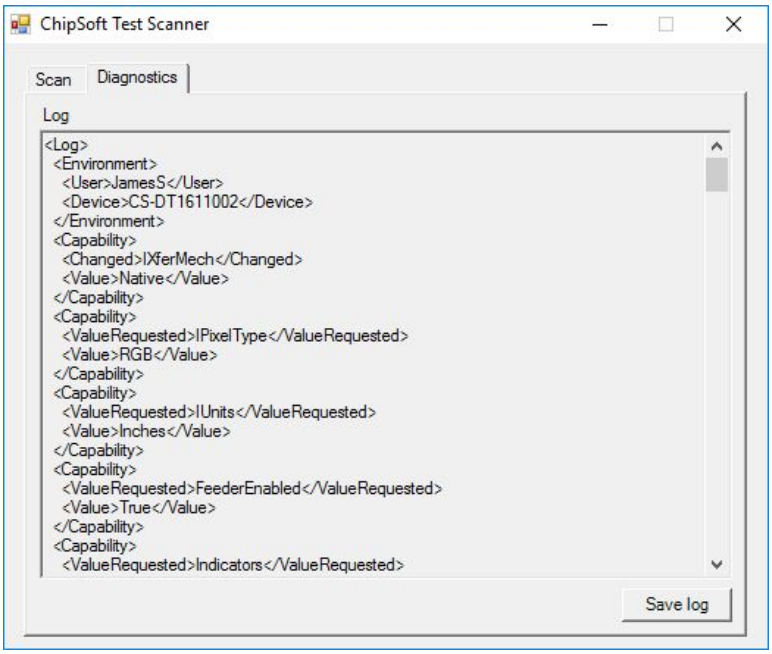
Tabel 19: Capability inventaris

Capability	Prioriteit
CAP_AUTOFEED	S
CAP_FEEDERENABLED	S
CAP_PAPERDETECTABLE	S
CAP_FEEDERLOADED	S
CAP_DUPLEX	S
CAP_DUPLEXENABLED	S
CAP_UICONTROLLABLE	M
ICAP_XFERMECH	M
ICAP_IMAGEFILEFORMAT	M
ICAP_UNITS	M
ICAP_XRESOLUTION	M
ICAP_YRESOLUTION	M
ICAP_PIXELTYPE	M
ICAP_BITDEPTH	M
ICAP_BITDEPTHREDUCTION	M
ICAP_THRESHOLD	S
ICAP_BRIGHTNESS	C
ICAP_CONTRAST	C
ICAP_GAMMA	C
ICAP_PHYSICALWIDTH	M
ICAP_PHYSICALHEIGHT	M
ICAP_ROTATION	M
ICAP_SUPPORTEDSIZES	S

B UI



Figuur 14: ChipSoftDataSource scan UI



Figuur 15: ChipSoftDataSource logging UI

C Afstudeerplan

Afstudeerplan

Informatie afstudeerder en gastbedrijf

Afstudeerblok: 2016-2.2 (start uiterlijk 14 november 2016)
Startdatum uitvoering afstudeeropdracht: 1 november 2016
Inleverdatum afstudeerdossier volgens jaarrooster: 20 maart 2017

Studentnummer: 20068551
Achternaam: dhr
Voorletters: J S
Roepnaam: James
Adres: Fregatsingel 123
Postcode: 2496 ZV Den Haag
Woonplaats: Den Haag
Telefoonnummer:
Mobiel nummer: 062 483 6412
Privé emailadres: jamess.strickland@gmail.com

Opleiding: Informatica
Locatie: Den Haag
Variant: voltijd

Naam studieloopbaanbegeleider: G.C.M. Heijne
Naam begeleidend examinator: E.M. van Doorn
Naam tweede examinator: H.G.J. Bechet

Naam bedrijf: ChipSoft
Afdeling bedrijf:
Bezoekadres bedrijf: Orlyplein 10
Postcode bezoekadres: 1043 DP
Postbusnummer: Postbus 37039
Postcode postbusnummer: 1030 AA
Plaats: Amsterdam
Telefoon bedrijf: +31 (0)20 4939 000
Telefax bedrijf: +31 (0)20 633 1975
Internetsite bedrijf: https://www.chipsoft.nl/

Achternaam opdrachtgever: dhr Deege
Voorletters opdrachtgever: J
Titulatuur opdrachtgever: Ing.
Functie opdrachtgever: Software ontwikkelaar
Doorkiesnummer opdrachtgever: nvt
Email opdrachtgever: jdeege@chipsoft.nl

Achternaam bedrijfsmentor: dhr Deege
Voorletters bedrijfsmentor: J
Titulatuur bedrijfsmentor: Ing.
Functie bedrijfsmentor: Software ontwikkelaar
Doorkiesnummer bedrijfsmentor: nvt
Email bedrijfsmentor: jdeege@chipsoft.nl

Doorkiesnummer afstudeerder: nvt
Functie afstudeerder (deeltijd/duaal): Software ontwikkelaar

Titel afstudeeropdracht:

Het ontwikkellen van een multimedia-simulatietool voor ChipSoft

Opdrachtomschrijving

1. Bedrijf

ChipSoft specialiseert zich in de zorgautomatisering. Als innovatieve partner stelt ChipSoft de gehele zorgketen in staat om op efficiënte wijze zorg te leveren. Het team bestaat uit ruim 500 mensen met een hart voor zorg en één gemeenschappelijk doel: het ontwikkelen van de allerbeste software voor de zorgsector.

HiX, hun eigen elektronisch patientendossier of *epd*, is een complete, innovatieve en gebruiksvriendelijke totaaloplossing op de Europese markt. Zorginstellingen beschikken hiermee over één oplossing voor het gehele multidisciplinaire zorgtraject. Van uitgebreide dossiervoering, zorgadministratie en –logistiek tot planning en eHealth. HiX ondersteunt elke gebruiker optimaal tijdens zijn werkzaamheden.

2. Probleemstelling

HiX is een applicatie met een 3-tierarchitectuur die specifieke functionaliteiten in de vorm van modules implementeert. Eén van deze modules is een multimediamodule waarmee multimediabestanden in HiX toegevoegd kunnen worden. Een veel voorkomend type dergelijk multimediabestand zijn documenten die door zorgverleners worden in-gescand. Bij een zorginstelling kan een beheerder in de multimediamodule configureren met welke instellingen documenten gescand worden. Denk hierbij bijvoorbeeld kleur, beeldcompressie, dubbelzijdig scannen, etc. Op die manier blijft de scan-actie voor gebruikers eenvoudig en wordt de dataopslag op een correcte wijze benut.

Bij major releases van HiX worden op dit moment, om het systeem te testen, handmatig digitale scanners gekoppeld aan de multimediamodule. Vanwege het beperkte aantal fysieke scanners bij ChipSoft en de diversiteit aan scannerfeatures en mogelijke instellingen bij het inscannen van documenten is dit een tijdrovend proces.

3. Doelstelling van de afstudeeropdracht

ChipSoft is op zoek naar een tool waarmee koppelingen tussen de multimediamodule en fysieke multimedia devices getest kunnen worden zonder deze apparaten fysiek te hoeven aansluiten. Hierbij worden de koppelingen getest met gesimuleerde devices die als echte devices aangesproken worden. De focus ligt initieel op een device (scanner/webcam) die via *twain* en/of *wia* aanspreekbaar is. *Twain* en *wia* bieden beide een *API* om op gestandaardiseerde wijze met onder andere scanners te communiceren. Er zal worden onderzocht wat beschikbaar is op het gebied van virtuele scanners en toe te passen is binnen de oplossing. De te ontwikkelen tool dient waar mogelijk zo opgezet te worden dat deze uitgebreid kan worden om andere devices (MRI\CT\Echo apparaten) te simuleren en andere onderdelen van de multimediamodule te testen.

Het doel van de opdracht is: een multimediasimulatie-tool ontwikkelen die fysiek aangesloten multimedia devices simuleert in communicatie en uitvoer waarmee de multimediamodule van HiX getest kan worden.

4. Resultaat

Met de simulatietool kan ChipSoft bij het uitvoeren van koppelingentests fysiek gekoppelde multimedia devices simuleren waardoor fysieke koppelingen niet langer nodig zijn. Dit bespaart de opdrachtgever tijd en biedt de mogelijkheid om sneller veranderingen te kunnen testen en doorvoeren in HiX en de

multimediamodule. De tool maakt het mogelijk om het testen van koppelingen tussen multimedia devices en de multimediamodule van HiX te automatiseren. De tool kan ook gebruikt worden om de inrichting bij zorginstellingen eenvoudiger te testen, zodat de kwaliteit van de inrichting verbeterd kan worden en met behulp van logging fouten in de koppelingen geanalyseerd kunnen worden.

5. Uit te voeren werkzaamheden, inclusief een globale fasering, mijlpalen en bijbehorende activiteiten

Activiteit	Tijdsduur
Plan van aanpak schrijven	5 dagen (lineair)
Pakketselectie uitvoeren	10 dagen (lineair)
Ontwerpen en ontwikkelen van de simulatietool	45 dagen (incrementeel en iteratief)
Testen van de simulatietool	45 dagen (incrementeel en iteratief)
Afstudeerdossier opbouwen	15 dagen (gedurende de hele opdracht)

Tijdens het uitvoeren van het project wordt van de volgende technieken gebruik gemaakt:

C#
Visual Studio
.Net
Twain/WIA
UML

Voor het uitvoeren van de pakketselectie wordt een methode gebruikt die gebaseerd is op de pakketselectiemethode die KPMG hanteert. Hierbij is er een initiële analyse naar de eisen en wensen voor het te selecteren pakket, en wordt met behulp van een longlist -> shortlist de keuze gemaakt en een aanbeveling uitgebracht.

Voor het ontwerpen en ontwikkelen van de tool wordt *test driven development* gebruikt, in combinatie met een voor individuele uitvoering aangepaste versie van scrum.

Indien nodig worden virtuele drivers ontwikkeld en getest in een virtuele Hyper-V omgeving om te voorkomen dat het besturingssysteem van de werkstation onstabiel wordt.

In de planning zijn 10 dagen gereserveerd voor eventuele uitloop en extra flexibiliteit in de planning. De activiteiten in de met lichtblauw gearceerde rijen worden parallel uitgevoerd.

6. Op te leveren (tussen)producten

- Plan van aanpak
- Advies op basis van de pakketselectie

De volgende producten worden op incrementele en iteratieve wijze ontwikkeld:

- Ontwerpdocumentatie
- Simulatietool

Aan de hand van de tests en ontwikkelde code wordt het ontwerp vormgegeven.

Door het gebruik van test driven development wordt er gedurende het traject test cases (scripts) opgesteld die samen het testdraaiboek vormen.

- Testscripts, testdraaiboek en testrapportages

7. Te demonstreren competenties en wijze waarop

- Selecteren van standaardsoftware
  Eindniveau: 3

Tijdens het uitvoeren van de pakketselectie worden er selectiecriteria opgesteld. Deze vormen de basis voor een longlist die verder verfijnd wordt tot een shortlist. Op basis hiervan wordt een advies uitgebracht.

- Ontwerpen systeemdeel
  Eindniveau: 3

Tijdens de ontwikkeling van de tool kan er steeds na het ontwikkelen van geteste en functionele systeemdelen het ontwerp beschreven worden. De verwachting is dat tijdens het ontwikkelen design patterns die toegepast kunnen worden naar voren komen. Vanwege de wijze van ontwikkeling zal er naast een goed beeld over de werking en structuur van het systeem ook een systeem ontwikkeld worden die low coupling heeft naast andere aspecten van degelijke softwareontwikkeling principes.

- Bouwen applicatie
  Eindniveau: 3

Door de gehanteerde ontwikkelmethode vind de ontwikkeling in een testomgeving plaats. Hierdoor zijn de mogelijkheden tot toekomstige wijzigingen en hergebruik geoptimaliseerd en is de applicatie al uitvoerig getest bij oplevering. Het ontwikkelen gebeurt ook in een geavanceerde framework met een programmeertool. Deze zijn respectievelijk .NET en Visual Studio. Versiebeheer gebeurt met behulp van *Team Foundation Server*.

- Uitvoeren van en rapporteren over het testproces
  Eindniveau: 3

Vanwege de gehanteerde ontwikkelmethode wordt er veelvuldig gebruik gemaakt van testscripts die opgesteld worden aan de hand van functionele en niet-functionele kwaliteitsattributen. Dit gebeurt binnen een testframework en zal de ontwikkeling en het ontwerp leiden. Deze testscripts vormen de testdraaiboek. Aan het einde van het ontwikkeltraject worden er tests opgesteld voor het gehele systeem die de invoer vormen voor testrapportages.

D Plan van aanpak

1 Achtergronden

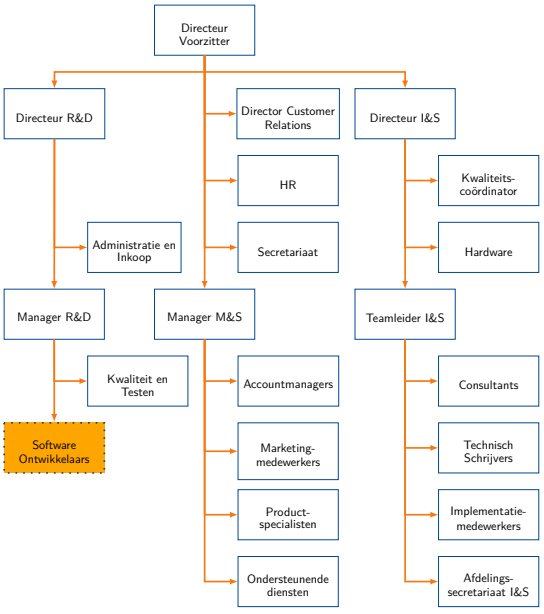
In dit hoofdstuk wordt eerst de organisatie waar de opdracht wordt uitgevoerd omschreven. Een organogram geeft de hiërarchie van de organisatie weer. De aanleiding voor de opdracht wordt beschreven. Tot slot volgt een overzicht van de betrokken personen en hun rol binnen de opdracht.

1.1 Organisatie

ChipSoft specialiseert zich in software voor de zorgsector en is marktleider in de Nederlandse ziekenhuissector met hun eigen zorginformatiesysteem. Het voornaamste product is een eigen ontwikkelde elektronisch patiëntendossier die door ziekenhuizen wordt afgenomen. Dit pakket heet HiX en staat voor *Healthcare Information eXchange*. Daarnaast automatiseert ChipSoft ook Zelfstandige Behandel Centra (klinieken) en GGZ-instellingen.

ChipSoft heeft twee vestigingen in Amsterdam waarvan één het hoofdkantoor is en de ander een opleidingscentrum. Ook in Heerenveen en Hoogeveen heeft ChipSoft vestigingen waar voornamelijk ontwikkeling plaatsvindt. De opdracht wordt uitgevoerd op het hoofdkantoor in Amsterdam.

1.2 Organogram



Figuur 1: Organogram

ChipSoft is een lijnorganisatie met drie afdelingen, namelijk Research & Development, Marketing & Sales en Implementatie & Support. De opdracht wordt uitgevoerd op de afdeling Research & Development bij de software ontwikkelaars. In figuur 1 is een organogram van de organisatie te zien en is met oranje gearceerd waar de opdracht wordt uitgevoerd. De subafdeling waar de opdracht wordt uitgevoerd bestaat uit een groep ontwikkelaars die zich met de multimediamodule van HiX bezighouden. De teamleider hiervan is Jurgen Deege.

1.3 Aanleiding

Aan de multimediamodule in HiX kunnen multimedia-apparaten, zoals MRI-scanners, echo apparaten of digitale scanners, gekoppeld worden. Bij major releases van HiX worden in het huidige proces de koppelingen met digitale scanners handmatig fysiek getest. Dit proces kost ChipSoft veel tijd. Ook is het zo dat bij problemen met de multimediamodule bij zorginstellingen het moeilijk kan zijn om te achterhalen wat de oorzaak is. Deze problemen zijn de aanleiding om een diagnostische tool te ontwikkelen waarmee de koppelingen getest kunnen worden. Daarnaast is er ook een aanleiding ontstaan omdat de opdrachtnemer het doel heeft om af te studeren. Met het uitvoeren van de opdracht toont de opdrachtnemer zijn tijdens de opleiding opgedane competenties.

1.4 Rollen

Binnen de opdracht zijn voornamelijk twee betrokkenen, deze zijn:

Opdrachtgever ChipSoft is de opdrachtgever en wordt vertegenwoordigt door de teamleider van R&D Multimedia\PACS, namelijk Jurgen Deege. Jurgen Deege is de key stakeholder van het project.

Opdrachtnemer De opdrachtnemer is James Strickland, student Informatica aan de Haagse Hogeschool. Aan de hand van het vooraf opgestelde afstudeerplan wordt de opdracht door hem ingevuld.

2 Projectopdracht

Allereerst wordt hier de titel van het project gedeclareerd. Vervolgens worden de doelstelling en het gewenste eindproduct van het project toegelicht.

2.1 Projecttitel

Het Project heeft de volgende titel: *Het ontwikkelen van een multimedia test-tool voor ChipSoft*.

2.2 Doelstelling

Het doel van het project is het ontwikkelen van een test-tool voor de multimediamodule van HiX met een initiële focus op TWAIN en/of WIA devices. Met de te ontwikkelen tool kan de koppeling met digitale scanners getest worden zonder een fysieke scanner. Daarbij wenst de opdrachtgever over diagnostische functionaliteiten te beschikken om te helpen bij het vaststellen van oorzaken van problemen in de koppelingen.

De behoefte voor de tool is bij de opdrachtgever ontstaan vanwege de onhandige werkwijze bij zowel het testen, als het troubleshooten van problemen met scanners. Om de koppelingen te testen moeten fysieke apparaten gekoppeld worden. Deze zijn niet overal geïnstalleerd bij de testfarm servers van HiX. Daar komt bij dat er meer onderhoud en beheer vereist is bij het inrichten van servers met fysieke scanners.

De opdrachtgever wil geautomatiseerd kunnen testen met virtuele scanners. Daarbij is het nu voor de opdrachtgever moeilijk om te zien wat er precies gebeurd bij gegevensuitwisseling tussen applicatie en scanner omdat dit nergens word gelogd. Ook komt het voor dat bij problemen met scanners bij een klant de opdrachtgever erheen moet rijden om te troubleshooten. De test-tool zou in dergelijke situaties veel tijd en geld kunnen besparen.

Een andere doelstelling van de opdracht is voor de opdrachtnemer het behalen van de studie. Met het uitvoeren van de opdracht toont de opdrachtnemer zijn competenties aan de onderwijsinstelling, die hem vervolgens voorziet van een diploma.

2.3 Eindproduct

Het eindproduct is een diagnostische test-tool waarmee koppelingen die van bepaalde protocollen gebruik maken getest kunnen worden. Zoals eerder aangegeven worden eisen en wensen later specifieker vastgesteld, wel zijn een aantal eisen al bekend.

2.3.1 Functionele requirements

- Scanner-configuraties en instellingen die vanuit HiX worden gebruikt bij het scannen kunnen verwerkt worden.
  - Dit gebeurt met een virtuele scanner.
  - Dit kan geautomatiseerd opgeslagen en herroepen worden.
- Gegevensuitwisseling in de koppeling wordt vastgelegd.
- De virtuele scanner kan zonder zijn eigen UI aangestuurd worden.
- De virtuele scanner levert een volgens de gevraagde instellingen correct weergegeven beeldbestand op.

2.3.2 Technische beperkingen

- Het systeem wordt ontwikkeld in C# binnen het .NET framework.
- Het systeem gebruikt de TWAIN en/of WIA standaarden.
- Het systeem beschikt over een API waarmee ontwikkeld kan worden.

2.3.3 Niet-functionele requirements

Uit de ISO 9126 standaard:

- Koppelbaarheid - Het systeem werkt volgens een standaard met andere applicaties.
- Wijzigbaarheid - Het is wenselijk later andere scanner-configuraties of scaninstellingen te kunnen implementeren.
- Testbaarheid - De correcte werking van scaninstellingen moet testbaar zijn.

3 Projectactiviteiten

Tijdens het project worden een aantal activiteiten uitgevoerd die tot producten leiden. Deze worden hier omschreven.

3.1 Opsomming activiteiten

De volgende activiteiten zullen lineair uitgevoerd worden aan het begin van het project, met uitzondering van het eerste punt. Het zal gedurende het hele project nodig zijn om specificaties van protocollen te raadplegen. Deze activiteiten bieden een basis waarmee het project wordt vormgegeven en biedt input op wensen en eisen voor de te ontwikkelen tool.

- Verdiepen in protocol(len) waarmee gewerkt wordt.
- Plan van aanpak maken.
- Selecteren van standaardsoftware.

Het eindproduct wordt incrementeel en iteratief ontwikkeld. De volgende activiteiten ondersteunen dit proces. Een iteratie wordt hier aangeduid met *sprint*.

- Samenstellen *Product Backlog*.
- *Sprint Planning*.
- *Sprint Reviews*.
- *Sprint Retrospective*.
- *Daily logs*.

Tijdens het uitvoeren van de sprints worden de volgende activiteiten uitgevoerd tijdens het ontwerp- en ontwikkelproces:

- Tests schrijven.
- Code schrijven/refactoren.
- Ontwerp vastleggen.

4 Projectgrenzen

In dit hoofdstuk wordt het project afgebakend met betrekking tot de tijd en verwachtingen.

4.1 Duur van het project

Het project begint op 14 november 2016 en heeft een minimale doorlooptijd van 17 werkweken van 40 uur. Hierna worden de producten opgeleverd en aan de hand van verslaglegging van de opdrachtnemer beoordeeld.

4.2 Projectafbakening

Tot het project hoort niet:

- HiX of de multimediamodule aanpassen.
- Nieuwe functionaliteiten in HiX of de multimediamodule implementeren.
- Medewerkers instrueren over gebruikte protocollen.

4.3 Randvoorwaarden

Voor een succes verloop van het project moeten er aan de volgende voorwaarden worden voldaan:

- Bereikbaarheid en betrokkenheid opdrachtgever.
- Medewerking van de *Product Owner* bij het vaststellen van wensen en eisen.
- Feedback bij sprint reviews van collega's en de *Product Owner*.
- Een werkstation voorzien van de juiste middelen om de test-tool te ontwikkelen.
- Proactieve houding van de opdrachtnemer.
- Goede communicatie tussen de betrokken partijen.

Naast bovengenoemde voorwaarden moet de opdrachtnemer ook voldoen aan de eisen die vanuit de onderwijsinstelling worden gesteld aan de afstudeeropdracht.

5 Producten

Het project leidt tot een aantal (deel-)producten. De eerste fase van het project wordt lineair uitgevoerd. Daarna wordt het project op iteratieve en incrementele wijze uitgevoerd.

5.1 Lineair ontwikkelde producten

Na de eerste drie tot vier weken van het project worden de volgende producten opgeleverd:

- Plan van aanpak. Projectdefinitie en afbakening.
- Adviesrapport gebaseerd op softwareselectie. Hierbij wordt er onderzoek gedaan naar beschikbare API's en andere toe te passen software binnen het project.

5.2 Iteratief ontwikkelde producten

Tijdens het ontwikkelproces zullen de volgende producten worden opgeleverd:

- Product Backlog. Dit zijn de eisen en wensen voor de tool.
- Sprint Planningen. Elke sprint wordt geplant door het kiezen van items uit de product backlog om in de sprint backlog te verwerken Per item wordt geschat hoeveel tijd het kost om te ontwikkelen.
- Sprint Backlogs. Dit zijn de items uit de product backlog die in de betreffende sprint ontwikkeld worden.
- Testscripts. Deze bestaan uit:
  - Unit tests.
  - Integration tests.
- Ontwerpdocumentatie volgens UML gedreven door test-driven development. Dit bestaat uit::
  - Toepasselijke diagrammen zoals klassen, sequentie, state, activity.
- Test-tool. Het eindproduct.

Voor beide vormen van de backlogs geldt dat deze continu aan het evolueren zijn op basis van sprint uitvoeringen en daaruit voortvloeiende resultaten.

6 Kwaliteit

6.1 Kwaliteit

In dit project worden een aantal technieken gebruikt om de kwaliteit van de producten te waarborgen:

- **Sprint reviews** - Elke sprint wordt samen met de Product Owner geëvalueerd. Dit levert feedback op waarmee de producten beter aansluiten op het gewenste eindresultaat van de opdrachtgever.
- **Codeconventies** - Ik schrijf code volgens dezelfde codeconventies die ChipSoft aanbeveelt. Dit bevordert de leesbaarheid en overdraagbaarheid van de code.
- **Test-driven development** - Dit levert software op die bestaat uit los van elkaar gekoppelde code, omdat je steeds aan eenduidige en specifieke functionaliteit werkt. Het idee is dat je dan klassen ontwikkelt die verantwoordelijk zijn voor hun eigen gedrag en in minder mate afhankelijk zijn van andere klassen. Bij oplevering komt daarbij dat de software voorzien is van uitgebreide unittests. Dit verhoogt de kwaliteit.
- **Loggen** - Het eindproduct legt gegevensuitwisseling vast in een logbestand. Hiermee kan geverifieerd worden of het de verwachte resultaten oplevert.

6.2 Technieken, tools en software

- **Ontwikkelomgeving** - Op mijn werkstation is Microsoft Visual Studio 2010 Premium met het .NET 4.0 framework geïnstalleerd.
- **Test framework** - Het test framework dat ik gebruik om test-driven development toe te passen is NUnit.
- **Versiebeheer** - Voor versiebeheer maak ik gebruik van de Team Foundation Server bij ChipSoft.
- **UML** - Bij het maken van ontwerpdiagrammen gebruik ik de UML 2.x standaard.
- **Leadtools** - Leadtools is een commercieel softwarepakket bestaande uit een verzameling imaging, oftewel beeldverwerking, SDK's. Twain schrijft het protocol voor waar binnen afspraken gemaakt kunnen worden tussen applicaties en scanners om data over te dragen. De beeldverwerkingsfunctionaliteit is nodig om ingevoerde beelden te kunnen omzetten naar de gevraagde scaninstellingen. Zo kan een scanner bijvoorbeeld een specifiek kader van een beeld scannen (het beeld ondergaat dan een herkadrering), of een grijstintenbeeld opleveren wanneer een kleurenbeeld wordt ingevoerd.
- **Twacker** - Twacker is een TWAIN applicatie waarmee TWAIN scanners getest kunnen worden. Met behulp van deze tool kan geverifieerd worden of een scanner en de features daarvan een correcte werking hebben. De tool vereist veel handmatig werk en heeft

niet de meest gebruikersvriendelijke interface, maar biedt wel de mogelijkheid om op een laag niveau een scanner te testen.

- **Twirl TWAIN Probe** - Dit is ook een tool waarmee TWAIN scanners getest kunnen worden. Het is onmisbaar wanneer je aan het ontwikkelen bent met TWAIN. De status van het TWAIN protocol is hierin zichtbaar. Capabilities en hun waardes kunnen uitgelezen, veranderd en gereset worden. Informatie over overgedragen data wordt inzichtelijk. Tot slot geeft deze tool waardevolle informatie over excepties door wanneer de debugger in Visual Studio eraan wordt gekoppeld.
- **Twister** - Twister is ook een test programma voor TWAIN. Het voert een analyse uit op een scanner, door automatisch alle capabilities te doorlopen en een scan uit te voeren met bepaalde instellingen. Na de analyse levert het programma een rapport op, waarin te zien is in hoeverre de scanner en de capabilities voldoen aan de TWAIN specificatie.
- **DosadiLog** - DosadiLog levert een gedetailleerde log op over een lopende TWAIN sessie. Dit is voorwaardelijk aan het gebruik van Twirl TWAIN Probe of Twister, omdat deze producten door hetzelfde bedrijf zijn ontwikkeld. In de log zijn alle TWAIN operaties en de bijbehorende resultaten te zien van de TWAIN sessie.

7 Projectorganisatie

Hier wordt de organisatie van het project weergegeven door per betrokkene te vermelden hoe ze te bereiken zijn, en welke rol ze binnen het project hebben. Tot slot wordt toegelicht op welke wijze gecommuniceerd wordt binnen de opdracht.

7.1 Betrokkenen

Jurgen Deege Functie: Software Ontwikkelaar en Teamleider R&D Multimedia\PACS
Email: jdeege@chipsoft.nl
Telefoonnummer: 020 493 9000

Jurgen vertegenwoordigt de opdrachtgever en is de key stakeholder. Hij zal ook de rol van *Product Owner* op zich nemen. Hij is doorgaans elke werkdag beschikbaar en heeft het eindwoord bij besluiten binnen het project.

James Strickland Functie: Software Ontwikkelaar R&D
email: jstrickland@chipsoft.nl
Telefoonnummer: 020 493 9000

James is de opdrachtnemer en is elke werkdag beschikbaar.

7.2 Communicatie

Communicatie tussen opdrachtnemer en opdrachtgever gebeurt in een directe lijn. Daarnaast is er de mogelijkheid om via email te communiceren, en maakt het bedrijf gebruik van 'Skype voor bedrijven'. Naast de sprint reviews, retrospectives en planning vindt er wekelijks een voortgangsgesprek plaats tussen opdrachtgever en opdrachtnemer. Alle gewerkte uren worden in het interne urenregistratiesysteem geregistreerd.

8 Planning

In tabel 1 wordt de globale planning voor het project getoond. De sprints hebben een vaste lengte van twee weken. Dit is een lengte dat voldoende tijd biedt om functionaliteiten te ontwikkelen en frequent genoeg feedback momenten oplevert om in te spelen op wijzigingen.

Datums	Activiteit
14/11/2016 - 02/12/2016	Plan van aanpak opstellen Softwareonderzoek uitvoeren
05/12/2016 - 16/12/2016	Sprint 1
19/12/2016 - 30/12/2016	Sprint 2
02/01/2017 - 13/01/2017	Sprint 3
16/01/2017 - 27/01/2017	Sprint 4
30/01/2017 - 10/02/2017	Sprint 5
13/02/2017 - 24/02/2017	Sprint 6
27/02/2017 - 10/03/2017	Sprint 7
13/03/2017 - 17/03/2017	Afronden afstudeerverslag

Tabel 1: Globale projectplanning

8.1 Toelichting planning

De planning begint met Week 1 op 14 november, en elke week geeft vijf werkdagen weer. De balken gemarkeerd met *Afstudeerverslag* worden gebruikt voor uitloop en voor het voorbereiden en afhandelen van sprints. Sprint planning, reviews en retrospectives gebeuren of aan het eind van de sprint, of op de afstudeerverslagdagen of aan het begin van nieuwe sprints. Dit is afhankelijk van wat qua tijd het beste uitkomt voor de opdrachtgever omdat het voor kan komen dat die dan afwezig is. De sprints worden gebruikt voor het ontwerpen, ontwikkelen en testen van de op te leveren producten. Er is niet van tevoren bepaald welke activiteiten precies worden uitgevoerd per sprint, dit wordt telkens bepaald wanneer de betreffende sprint wordt gepland. Dit komt omdat er gebruik wordt gemaakt van scrum, en omdat we vaak pas zullen weten welke activiteiten uitgevoerd moeten worden na het opdoen van meer kennis en resultaten uit voorgaande sprints.

9 Kosten en baten

In dit hoofdstuk wordt eerst een schatting gemaakt van de kosten van dit project. Daarna volgt een schatting van de baten.

9.1 Kosten

De opdrachtnemer ontvangt een stagevergoeding van € 712,40 per maand op basis van 40 uur per week. Daarnaast worden de reiskosten van de opdrachtnemer vergoed. Dit komt grof neer op ongeveer € 1000,00 per maand.

Het uitvoeren van de pakketselectie kan ook resulteren in aanschaf en/of licentiekosten voor de te ontwikkelen tool. De kosten hiervoor kunnen niet van tevoren bepaald worden. Initiële aanschafkosten variëren van enkele honderden of duizenden euro's. Jaarlijkse licentiekosten variëren van tientallen tot honderden euro's per jaar. De kosten worden ter zijner tijd bepaald.

9.2 Baten

De huidige kosten worden voornamelijk vertegenwoordigd door gemaakte werkuren bij het testen en/of troubleshooten. Het doel van de tool is om deze kosten te verminderen. Daarnaast worden ook reiskosten bespaard wanneer de opdrachtgever niet hoeft af te reizen naar een klant. Tot slot kan de te ontwikkelen tool leiden tot de mogelijkheid om de multimediamodule geautomatiseerd te testen. Dit bespaart mogelijk veel tijd, en dus geld.

10 Risico's

Hier is een tabel te vinden met mogelijke risico's waar het project mee te maken kan krijgen.

Tabel 2: Projectrisico's

Risico	**Maatregelen**
Deadlines worden niet gehaald	▪ Planning baseren op basis van werkschattingen en sprints. ▪ Overgebleven werk meenemen naar volgende sprints. ▪ Functionaliteitenscope aanpassen.
Niet genoeg kennis van TWAIN en/of WIA	▪ Specificaties en documentatie lezen. ▪ Gericht kennis verdiepen op basis van uit te voeren taken.
Niet genoeg kennis van C#	▪ Collega's om hulp vragen. ▪ Zoeken op internet. ▪ Gericht inlezen over C# op basis van de huidige taak.
Te onervaren met test-driven development	▪ Literatuur raadplegen. ▪ Code simpel houden en alleen bouwen wat nodig is. ▪ Scope van de toepassingswijze van test-driven development aanpassen.
Third-party libraries niet goed gedocumenteerd	▪ API specificaties raadplegen. ▪ Het internet raadplegen.
Moeite met het begrijpen of toepassen van third-party libraries	▪ Simplificeren tot een niveau wat wel begrijpbaar is. ▪ Op te leveren functionaliteitenscope aanpassen. ▪ Beschikbare voorbeelden gebruiken om begrip te verbreden. ▪ Afhankelijkheden verminderen met bijvoorbeeld dependency injection.

Woordenlijst

API Een verzameling definities op basis waarvan een computerprogramma kan communiceren met een ander programma of onderdeel.. 7

capability Een TWAIN capability is een optie of instelling waar een acquisitieapparaat over beschikt. Met andere woorden, capabilities vormen het vermogen van wat een acquisitieapparaat allemaal kan doen.. 9

HiX Healthcare Information eXchange, het in-huis gebrouwen elektronisch patiëntendossier systeem en voornaamste product van ChipSoft. 1, 2, 6

Product Owner De Product Owner (producteigenaar) is de opdrachtgever of klant. Hij beheert ook de product backlog, hij bepaalt wat er moet gebeuren en in welke volgorde.. 8

SDK Een software development kit (vaak afgekort tot SDK) is een verzameling hulpmiddelen die handig zijn bij het ontwikkelen van computerprogramma's voor een bepaald besturingssysteem, type hardware, desktopomgeving of voor het maken van software die een speciale techniek gebruikt.. 8

sprint Een periode van 1 tot 4 weken waarin men een aantal zaken van de productbacklog oppakt. Dit is een time-box.. 8

test-driven development Een ontwikkelmethode voor software waarbij eerst tests worden geschreven en daarna de code.. 8

troubleshooten Het opsporen en oplossen van fouten. 3, 12

TWAIN TWAIN is de bekendste industriestandaard API om data via een scanner in een computer in te lezen. 3, 4, 8, 9, 13

WIA WIA is een API en driver model van Microsoft waarmee software met beeldhardware kan communiceren op Windows besturingssystemen. 3, 4, 13

E Adviesrapport

1 Inleiding

Dit adviesrapport is gebaseerd op een softwareonderzoek wat is uitgevoerd in het kader van de ontwikkeling van een test-tool voor scanners. Eerst zal dit hoofdstuk de aanleiding voor dit onderzoek beschrijven. De probleemstelling brengt het doel van het advies in kaart. Tot slot wordt hier omschreven aan welke selectiecriteria het te kiezen softwareproduct moet voldoen. Dit vormt de basis voor de onderbouwing van de gemaakte keuze.

1.1 Aanleiding

Dit advies is uitgebracht in het kader van een afstudeeropdracht bij ChipSoft die bestaat uit het ontwikkelen van een test-tool voor digitale scanners. Binnen deze opdracht is het gewenst om over een virtuele scanner te beschikken.

1.2 Probleemstelling

Binnen ChipSoft wordt op dit moment de koppeling tussen hun eigen patiënteninformatiesysteem en digitale scanners getest met fysieke scanners. Wanneer problemen voorkomen met deze koppeling is het nodig om in de broncode te debuggen, en moet er een fysieke scanner gebruikt worden. Dit kost ChipSoft veel tijd en zodoende heeft ChipSoft de opdracht gegeven om een test-tool te ontwikkelen die met behulp van een virtuele scanner de noodzaak om van fysieke scanners gebruik te maken wegneemt. Het doel van dit advies is om vast te stellen wat hiervoor op de markt beschikbaar is.

1.3 Selectiecriteria

Met behulp van de globale requirements uit het plan van aanpak heb ik de selectiecriteria bepaald. Daarbij heb ik ze uitgebreid na meer kennis te hebben over TWAIN en gespecificeerd wat wenselijk is in de bijbehorende API. De selectiecriteria zijn als volgt gedefinieerd:

- De software moet met C# werken.
- De software maakt gebruik van of is een virtuele scanner.
  - De capabilities (functionaliteiten waar een scanner over beschikt) hiervan kunnen naar gelang uitgebreid of aangepast worden.
- De software gebruikt TWAIN en/of WIA.
- De software beschikt over een API.
- Scans kunnen uitgevoerd worden zonder de UI te tonen.
- De API is goed gedocumenteerd.

- Met de API kunnen berichten naar de scanner vastgelegd worden.
- Met de API is het mogelijk meerdere variaties van scan configuraties en instelling op te slaan en geautomatiseerd te laden.

De selectiecriteria zijn met de opdrachtgever besproken en door hem goedgekeurd.

2 Alternatieven

Hier worden alle gekozen softwareproducten in lijsten op basis van de selectiecriteria tegenover elkaar afgewogen. Het proces en de onderbouwing van de keuzes is in het afstudeerverslag beschreven.

2.1 Aanpak

Om de software te kiezen is gebruikt gemaakt van internetonderzoek. Tijdens het onderzoek werd het al snel duidelijk dat er geen virtuele WIA scanners te vinden zijn, op een enkele na die eigenlijk gebaseerd is op TWAIN maar een WIA jasje heeft. Gezien deze is ontwikkeld in C++ en verder geen C# functionaliteiten biedt viel dit al snel af. Ook kreeg ik het betreffende product niet aan de praat. Het werd ook snel duidelijk dat er voor C# meer te vinden was met TWAIN dan voor WIA. Zodoende is de focus van het onderzoek daarna gericht op TWAIN.

Om ervoor te zorgen dat een passende oplossing geboden wordt is de werking van het TWAIN protocol onderzocht. Zo kon het gekozen softwareproduct beter geplaatst worden in de juiste context. Het proces daarvan en de omschrijvingen van de bevindingen zijn in het afstudeerverslag gedocumenteerd.

2.2 Softwarekandidaten

Na te hebben besloten hoe het eindproduct ontwikkelen zou worden, is er onderzocht of er virtuele scanners, of sources, beschikbaar zijn op de markt. De software die onderzocht is is te onderscheidden in drie categorieën:

- Kant en klare commerciële TWAIN sources.
- Publiekelijk beschikbare TWAIN sources die naar gelang uitgebreid en gecompileerd kunnen worden.
- Een C# class library, namelijk Saraff.Twain.DS, die de TWAIN specificatie volledig implementeert.

Saraff.Twain.DS is een volledige implementatie van de TWAIN specificatie in C# .NET. Het is een verzameling van klassen en code waarmee ontwikkelaars TWAIN sources kunnen ontwikkelen die voldoen aan de TWAIN specificatie.

In tabel 1 heb ik in de kolom 'TWAIN source' de kant-en-klare commerciële sources onderscheidden van andere categorieën door aan deze een enkele plus toe te kennen. Het laatste item in de tabel is de gevonden class library. Dit waren alle beschikbare software pakketten die ik kon vinden.

Tabel 1: Kandidaten

Softwareproduct	C#	Source	Capabilities	API	Scan zonder UI
UniTwain	-	+	-	-	-
Scanpoint Virtual TWAIN	-	+	-	-	-
TwainImporter	-	+	-	+	+
Dosadi GenDS	-	++	+	+	+
TWAIN Sample Data Source	-	++	+	+	+
Saraff.Twain.DS	++	++	+	++	+

2.3 Selectie

Om te bepalen in hoeverre de producten voldoen aan de criteria, heb ik ze op mijn werkstation geïnstalleerd en uitgeprobeerd. Ook heb ik, waar beschikbaar, bijbehorende documentatie doorgenomen. Helaas kon ik dit niet doen voor Scanpoint Virtual TWAIN. Er was zowel geen demoversie als documentatie te vinden. Dat betekent dat dit pakket afvalt, omdat ik niet kan vaststellen in hoeverre het aan de criteria voldoet.

Zowel TwainImporter als UniTwain bieden niet de mogelijkheid om capabilities naar wens aan te passen of uit te breiden. Dit is een belangrijk criterium, omdat het wenselijk is in de Source een grote hoeveelheid typen scanner-configuraties en scaninstellingen te bieden. Wanneer de capabilities niet zijn aan te passen, blijft de virtuele scanner beperkt tot de capabilities en hun mogelijke waardes die erin zijn gebouwd door de ontwikkelaar. Daarmee vallen deze kandidaten af.

De mogelijkheid om te scannen zonder UI is een belangrijk criterium, omdat het gewenst is bij het testen van de multimediacomponent te scannen op grote schaal en met batchverwerking. Dat maakt het lastig om geautomatiseerd verschillende scanner-configuraties en scaninstellingen op te roepen en uit te voeren, omdat een tester dan steeds tussen elke scan door in de UI de instellingen moet wijzigen. Daarmee valt UniTwain, naast dat het geen interface biedt in de vorm van een API, als kandidaat af.

De producten met een enkele plus bij de API zijn niet geïmplementeerd in C#. Dat betekent dat ondanks met deze producten een systeem te ontwikkelen is dat voldoet aan de requirements, het implementeren hiervan te complex is binnen het kader van deze opdracht. Daarnaast is het niet goed uit te breiden of overdraagbaar voor ChipSoft, omdat hier niet met C++ ontwikkeld wordt.

De reden dat het te complex is, is te wijten aan het feit dat TWAIN implementaties op besturingssystemen de basis hebben in unmanaged code, specifiek C. Er moet gebruik worden gemaakt van een TWAIN.H header bestand en C code om een applicatie of source te laten voldoen aan de TWAIN standaard. Unmanaged code is code die rechtstreeks naar machinetaal wordt gecompileerd en direct door het besturingssysteem gedraaid wordt. C# is een taal waarmee managed code gegenereerd wordt. Managed code draait in een virtuele

machine en is voorzien van functionaliteiten zoals memory management. Door het gebruik van gecompileerde managed code hoeft een ontwikkelaar niet bang te zijn om onveilige of onstabiele software te ontwikkelen, waardoor ze zich kunnen richten op bedrijfslogica. Typisch gezien kunnen ze dit ook met minder regels code doen.

Het mengen van beide vormen van code brengt een niveau van complexiteit met zich mee dat buiten de scope van deze opdracht valt. Daarvoor is veel kennis van beide talen en Microsoft-centrische interoperabiliteit nodig. In het kader van deze opdracht heb ik samen met de opdrachtgever besloten dat dit te complex is en niet realistisch is om te realiseren binnen de beschikbare tijd.

3 Advies

In dit hoofdstuk wordt het uiteindelijk advies uitgebracht over het gekozen pakket. De conclusie is gebaseerd op de resultaten verkregen uit het voorgaand omschreven proces en verfijning van kandidaten uit een longlist naar een shortlist.

3.1 Conclusie

Uit de shortlist wordt duidelijk dat er een enkele keuze overblijft. De opdrachtgever wil de onderhoudbaarheid en overdraagbaarheid van de te ontwikkelen tool binnen ChipSoft garanderen. Dat is de voornaamste reden dat de tool in C# ontwikkeld moet worden.

De gekozen tool ondersteunt de TWAIN 2.3 specificatie volledig en biedt binnen de class library genoeg mogelijkheden om de gewenste functionaliteiten te implementeren. Daar komt bij dat de ontwikkelaar van deze library het volgens object georiënteerde principes heeft ontwikkeld, en al het zware werk dat komt kijken bij het programmeren van interactie tussen managed en unmanaged code heeft gedaan. Dit is nodig omdat TWAIN is geïmplementeerd in managed code, en C# unmanaged code is. Dit zorgt ervoor dat de gebruiker van de library zich kan focussen op ontwikkelen van een TWAIN data source naar eigen wens zonder zich zorgen te maken hoeven maken over de implementatie van de TWAIN standaard en het benaderen van managed code vanuit unmanaged code.

Er is een klein minpuntje aan het gekozen pakket, namelijk dat het door een persoon uit Wit Rusland is ontwikkeld. Hierdoor zijn veel van de eigen geschreven classes en daarbij behorende methods en attributen niet heel goed gedocumenteerd in het Engels. Wel zijn er mooie klassendiagrammen en voorbeelden in documentatie beschikbaar, en is de opzet van de library zodanig dat het niet al te moeilijk te begrijpen is qua werking en structuur.