Het realiseren van een beheertooling binnen de standaard FileNet web interface (Content Navigator) voor het op FileNet gebaseerde documentmanagementsysteem Digidoc.

Afstudeerverslag

Auteur: Tony van Leeuwen

De Haagse Hogeschool
HBO ICT – Software Engineering

Afstudeerbedrijf: Ministerie van Binnenlandse Zaken – SSC-ICT
Bedrijfsbegeleider: Jeroen den Hollander
Expert examinator: Gé van Toorenburg
Begeleidend examinator: Rianne Bechet – Tjoonk

11 januari 2021

Referaat

Afstudeerverslag van Tony van Leeuwen, geschreven in het kader van het afstuderen bij de opleiding HBO-ICT Software Engineering aan de Haagse Hogeschool. Dit verslag beschrijft het proces dat Tony van Leeuwen heeft doorlopen tijdens het uitvoeren van zijn afstudeeropdracht in de periode van 31 augustus 2020 t/m 11 januari 2021.

Descriptoren:

- Afstudeerverslag
- Scrum
- IBM FileNet
- IBM Content Navigator
- API
- Plugin
- Maatwerk
- MVC
- Java
- JavaScript
- Dojo Toolkit

Voorwoord

Dit verslag is geschreven in het kader van mijn afstuderen voor mijn opleiding HBO-ICT Software Engineering aan de Haagse Hogeschool. Ik heb een opdracht uitgevoerd bij één van de grootste ICT-dienstverleners (SSC-ICT) van het Rijksoverheid. Binnen deze opdracht heb ik een beheertooling gerealiseerd binnen de standaard FileNet web interface genaamd Content Navigator voor het op FileNet gebaseerde documentmanagementsysteem Digidoc. De focus van de opdracht lag op het ontwikkelen van maatwerk in de vorm van een plugin voor de beheertooling in Content Navigator die beschikt over gewenste zoek functionaliteiten.

Ondanks ik niet naar kantoor kon komen door de COVID-19 pandemie, heb ik mijn afstuderen binnen de afdeling ECM FileNet in het Digidoc Online team bij SSC-ICT als zeer prettig ervaren. In het begin was het even wennen, omdat het voor mij een nieuwe omgeving was waar ik in terecht kwam en met mijn nieuwe collega's alleen online kon afspreken.

Graag wil ik alle personen die betrokken waren bij mijn afstudeeropdracht bedanken en in het bijzonder:

Luc Verstegen (teammanager van ECM FileNet), voor de hartelijkheid en mogelijkheid om binnen de afdeling ECM FileNet te kunnen afstuderen.

Jeroen den Hollander (bedrijfsbegeleider), voor de goede begeleiding, betrokkenheid, feedback en alle besprekingen.

De collega's binnen de afdeling ECM FileNet die mij op welke manier dan ook hebben geholpen tijdens mijn afstuderen.

Vanuit school wil ik Rianne Bechet – Tjoonk en Gé van Toorenburg bedanken voor de goede begeleiding en de tussentijdse feedback die zij mij gegeven hebben. Ook wil ik Nederlands docent Peter den Hollander bedanken voor de tussentijdse feedback op mijn Nederlandse taalkundigheid.

Ik wens u veel plezier toe bij het lezen van dit verslag!

Tony van Leeuwen
Den Haag, 11 januari 2021

Begrippenlijst

In dit hoofdstuk staan alle gebruikte begrippen die een speciale betekenis hebben in de context van IBM Content Navigator met betrekking tot beheer en ontwikkeling. Overige begrippen en termen zijn tussen de regels door geplaatst en kunnen worden overgeslagen als het begrip of de term al bekend is.

- **Content Navigator:** Content Navigator is de standaard web interface van IBM FileNet. Met Content Navigator is het voor gebruikers mogelijk om middels een web client ongestructureerde informatie te beheren in een desktop.
- **Repository:** Een repository in dit verslag is een documentmanagement-systeem waarmee Content Navigator verbinding kan maken om Content Navigator te voorzien van de benodigde data.
- **Desktop:** Een desktop is een op maat gemaakte weergave in Content Navigator waarin verschillende functionaliteiten beschikbaar gesteld kunnen worden.
- **Plugin:** Een plugin in Content Navigator biedt de mogelijkheid om maatwerk te implementeren in Content Navigator, zonder daarvoor de code van Content Navigator aan te moeten passen.
- **Extension point:** Een extension point is een extensie die gebruikt kan worden bij het ontwikkelen van een plugin, om een functionaliteiten toe te voegen of bestaande functionaliteit te wijzigen in de plugin. Het is feitelijk een verwijzing naar de Java klasse.
- **Feature:** Een feature verwijst naar een menu item op het hoogste niveau in Content Navigator, zoals bladeren, zoeken of favorieten. Met een plugin is het mogelijk een nieuwe menu item toe te voegen in het bestaande menu.
- **Lay-out:** De lay-out is de algehele look en feel van Content Navigator. Met een plugin is het mogelijk om de bestaande lay-out te wijzigen of een compleet nieuwe lay-out te bouwen.
- **Content Navigator API:** Dit is een verzameling van verschillende Java en JavaScript klassen die een ontwikkelaar kan gebruiken om een plugin te ontwikkelen voor Content Navigator.
- **Widget:** Een widget is een UI-component waarmee een gebruiker kan interacteren in Content Navigator. Bijvoorbeeld een button waar de gebruiker op kan klikken of een textbox waar een gebruiker iets kan invoeren.
- **Dojo Toolkit:** Dojo Toolkit is een JavaScript library voor het ontwikkelen van client-side code voor een webapplicatie. Dojo biedt een library aan waarin verschillende widgets gebruikt kunnen worden om een user interface in Content Navigator te bouwen.
- **Dijit:** Een Dijit is een term dat wordt gebruikt om een Dojo widget aan te duiden.

Figuren- en tabellenlijst

In dit hoofdstuk staan alle verwijzingen naar de gebruikte figuren en tabellen.

Figuren

Tabellen

Inhoudsopgave

1 Inleiding

Benieuwd hoe ik vrijwel zonder kennis van de IBM FileNet omgeving een nieuwe beheertooling binnen de standaard web interface van FileNet (Content Navigator) heb gerealiseerd, waarbij de focus van dit project lag op het ontwikkelen van maatwerk in de vorm van een plugin die beschikt over gewenste zoek functionaliteiten?

In dit verslag wil ik u meenemen in het proces dat ik heb doorlopen tijdens het uitvoeren van deze opdracht.

SSC-ICT beheert en (door) ontwikkelt verschillende op IBM FileNet gebaseerde documentmanagementsystemen, waaronder het documentmanagementsysteem genaamd Digidoc. Sinds de upgrade naar een nieuwe versie van FileNet is de beheerclient genaamd FileNet Enterprise Manager - die de beheerders van Digidoc gebruiken om beheerwerkzaamheden uit te voeren voor Digidoc - vervangen door een nieuwe webbased beheertooling genaamd ACCE. Echter beschikt de ACCE niet over alle functionaliteiten die de FileNet Enterprise Manager wel had. Daarnaast wordt de FileNet Enterprise Manager niet meer ondersteund. Dit is de aanleiding geweest om een nieuwe beheertooling te realiseren binnen de standaard web interface van FileNet. De focus van dit project lag op het ontwikkelen van maatwerk voor minimaal één niet out-of-the-box functionaliteit die ontbreekt in de ACCE.

Dit verslag is als volgt opgedeeld:

- Het bedrijf en de opdracht;
- Het plan van aanpak;
- De sprint werkzaamheden;
- Het projectresultaat;
- De evaluatie;
- De bijlagen;

Allereerst geef ik wat achtergrondinformatie met betrekking tot het bedrijf, mijn positie binnen het bedrijf en de opdracht. Vervolgens heb ik in het plan van aanpak beschreven op welke wijze het project wordt uitgevoerd, welke middelen daarvoor nodig zijn en welke afspraken zijn gemaakt met mijn begeleider. Daarna heb ik per hoofdstuk (in de vorm van een Scrum sprint) de werkzaamheden beschreven die ik heb uitgevoerd tijdens het gehele realisatie proces. Aan het begin van elk sprint hoofdstuk geef ik een korte samenvatting van de werkzaamheden die ik heb uitgevoerd tijdens de betreffende sprint. Aan het einde van de sprint hoofdstuk evalueer ik de sprint waarbij ik aangeef wat de betreffende sprint heeft opgeleverd, wat er goed en/of minder goed ging en wat de eventuele leerpunten zijn. Vervolgens beschrijf ik de link tussen het opgeleverde resultaat en de aanvankelijke probleemstelling. Tot slot eindig ik dit verslag met een evaluatie van het proces, de opgeleverde producten en de vooraf gekozen beroepstaken.

2 Bedrijf en opdracht

2.1 Informatie over SSC-ICT

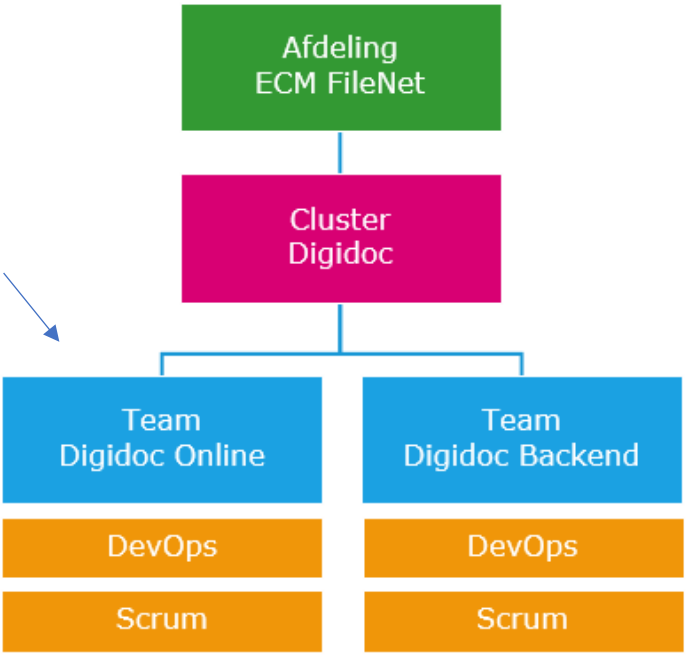
In dit hoofdstuk wordt het bedrijf SSC-ICT en de afdeling ECM FileNet geïntroduceerd waarbinnen dit project wordt uitgevoerd.

2.1.1 SSC-ICT

Het bedrijf Shared Service Center ICT (SSC-ICT) is één van de grootste ICT-dienstverleners van en voor het Rijk en valt onder het directoraat-generaal Vastgoed en Bedrijfsvoering Rijk (DGVBR) van het Ministerie van Binnenlandse Zaken en Koninkrijksrelaties (BZK) en is een baten-lastendienst. Bij SSC-ICT werken ruim 1.100 medewerkers die de ICT-dienstverlening verzorgen voor ruim 40.000 werkplekken. De missie van SSC-ICT is er om met ICT-services voor zorgen dat rijksmedewerkers hun werk in dienst van de samenleving altijd en overal veilig kunnen doen. SSC-ICT ondersteunt rijksmedewerkers met zijn digitale werkomgeving op 50.000 mobiele apparaten, op 50 locaties in Nederland en op ruim 150 ambassadeposten en consulaten over de hele wereld. Daarnaast beheert en ontwikkelt SSC-ICT een gevarieerd applicatielandschap, biedt SSC-ICT ICT-diensten met extra hoge beveiligingsniveaus en voeren SSC-ICT jaarlijks honderden ICT-projecten uit **(SSC-ICT, n.d.).**

2.1.2 Organogram

Ik heb een organogram **(Figuur 1: Organogram van de afdeling)** opgesteld van de omgeving waarin dit project is uitgevoerd. De oranje blokken geven aan welke methodieken er worden toegepast binnen het team. Naast het Digidoc cluster is er zoals beschreven ook DigiJust, VWS Ops en FN PAAS. Deze clusters heb ik niet meegenomen in de organogram, omdat deze clusters niet betrokken waren bij dit project. De pijl geeft aan wat mijn positie is binnen de ECM FileNet afdeling.



Figuur 1: Organogram van de afdeling

2.1.3 ECM FileNet

Zoals beschreven verzorgt SSC-ICT verschillende ICT-diensten binnen het Rijk. Het documentmanagementsysteem van IBM, FileNet, is één van deze ICT-diensten. FileNet is een documentmanagementsysteem waarin versiebeheer, toegang tot documenten en specifieke autorisatie centraal staat. Met de applicatie FileNet is het voor klanten mogelijk om ongestructureerde informatie op een centrale plek te beheren, denk aan het opslaan en beheren van documenten (bijvoorbeeld Office-documenten van het type Word (.doc), Excel (.xls) etc). Daarnaast is het ook mogelijk om werkstromen te koppelen aan de documenten.

Binnen SSC-ICT zorgt de afdeling ECM FileNet voor het beheer, ontwikkeling en doorontwikkeling van de verschillende FileNet documentmanagementsystemen. De afkorting ECM staat voor Enterprise Content Management en is een paraplubegrip waar meerdere deelgebieden onder vallen. Deze deelgebieden hebben gemeen dat er oplossingen worden gezocht om op een efficiënte wijze om te gaan met ongestructureerde informatie. Het documentmanagement-systeem is één van deze deelgebieden. De ministeries hebben elk een eigen inrichting van hun documentmanagementsysteem, voorbeelden van deze systemen zijn 'DigiJust' van het ministerie van Justitie en Veiligheid (JenV) en Digidoc, wat wordt gebruikt bij de ministeries van Binnenlandse Zaken en Koninkrijksrelaties (BZK), Sociale Zaken en Werkgelegenheid (SZW) en Financiën. Alle FileNet-applicaties die door de afdeling ECM FileNet worden beheerd, ontwikkeld en doorontwikkeld, kennen gezamenlijk ongeveer 15.000 eindgebruikers, waarvan er ongeveer 7.500 gebruik maken van Digidoc. De afdeling ECM FileNet bestaat uit 4 DevOps clusters. Het cluster dat DigiDoc levert

(Digidoc), het cluster dat DigiJust levert (DigiJust), het cluster dat FileNet-diensten levert aan VWS (VWS Ops) en FN PAAS dat de andere clusters ondersteund met gestandaardiseerde diensten rondom FileNet.

2.1.4 Digidoc Online

Binnen de afdeling van ECM FileNet wordt er gewerkt in verschillende DevOps teams. Tijdens het project maak ik onderdeel uit van het DevOps team 'Digidoc Online' dat onder het Digidoc cluster valt. De Digidoc cluster bestaat uit twee DevOps teams, Digidoc Online en Digidoc Backend. Het Backend team wordt ook wel Digidoc doorontwikkeling genoemd. Deze twee teams werken nauw samen en zal ik tijdens het project ook nauw samen werken met het Digidoc backend team. Het DevOps team Digidoc Online houdt zich vooral bezig met het (door) ontwikkelen en beheren van de webbased versie van Digidoc, genaamd Digidoc Online. Digidoc Online is de beoogde opvolger van de Digidoc desktop versie en wordt aangeboden op twee platforms. Allereerst op de browser en ten tweede op de iPad. De desktop versie wordt momenteel ontwikkeld, doorontwikkeld en beheerd door het backend DevOps team van Digidoc. Het Digidoc backend team houdt zich bezig met het door ontwikkelen van de desktop versie totdat deze is vervangen door de online versie Digidoc Online.

Het Digidoc Online team houdt zich dus bezig met het (door) ontwikkelen en beheren van de Web client, de user interface van Digidoc Online. Deze web client is ontwikkeld met Ionic op basis van Angular. Daarnaast ontwikkelen en beheren zij ook de Digidoc Services die zorgt voor de communicatie tussen de Web client en de onderliggende FileNet backend systemen. Deze service is ontwikkeld met Java. Het team werkt middels de Scrum methodiek die het voor hun mogelijk maakt om op een flexibele wijze problemen op te lossen of tegemoet te komen aan gebruikerswensen. Dit doet het team door een set aan user stories in een sprint van twee weken op te leveren waarna er na elke sprint in acceptatie wordt opgeleverd en om de twee sprints naar productie wordt opgeleverd. Het team bestaat uit een product owner, projectleider, architect, ontwikkelaars, functioneel ontwerpers, testers en beheerders.

2.2 Probleemstelling

Sinds de upgrade van FileNet is de standaard beheerclient van Digidoc genaamd FileNet Enterprise Manager (FEM), vervangen door een nieuwe webbased variant genaamd ACCE (Administrator Client Content Engine). In deze nieuwe webbased variant (ACCE) ontbreken grotendeels de beheerfunctionaliteiten die de beheerders van het documentmanagementsysteem Digidoc wel in de FEM beheerclient konden gebruiken om beheer uit te voeren voor Digidoc. Om toch goed beheer te kunnen uitvoeren wordt de oude FEM beheerclient nog steeds gebruikt in combinatie met de ACCE, terwijl de FEM beheerclient niet meer wordt ondersteund. Daarnaast worden er allerlei zelfgemaakte beheerscripts gebruikt op beheerservers. Dit is onwenselijk en daarnaast belemmerend voor de ambitie van de afdeling ECM FileNet om geen beheerders meer op servers toe te laten.

Na de upgrade van FileNet is er niet direct gekozen om een nieuwe beheertooling binnen de FileNet standaarden in te zetten, omdat de behoeften van de beheerpartij een lagere prioriteit had dan de behoeften van de reguliere eindgebruikers.

2.3 Doelstelling

De doelstelling van dit project is het realiseren van een nieuwe beheertooling binnen de standaard FileNet web interface (Content Navigator), die voorziet in de functionele behoeftes van de beheerders van het documentmanagementsysteem Digidoc. De focus van dit project ligt op het ontwikkelen van maatwerk voor minimaal één gewenste niet out-of-the-box functionaliteit in de beheertooling van Content Navigator. Met uiteindelijk als doel dat Content Navigator de FEM beheerclient en zelfgemaakte beheerscripts gaat vervangen.

2.4 Resultaat

Na afronding van dit project is er voor minimaal één gewenste niet out-of-the-box functionaliteit maatwerk ontwikkeld en zijn de gewenste out-of-the-box functionaliteiten geconfigureerd in de beheertooling van Content Navigator.

Het is voor de beheerders van het documentmanagement Digidoc mogelijk om de beheer functionaliteiten - die ontbreken in de ACCE en op basis van prioritering - uit te voeren in de gerealiseerde beheertooling binnen de standaard web interface van FileNet (Content Navigator).

De te realiseren beheertooling in Content Navigator zal breed worden ingezet, verschillend van configureren van Digidoc tot het analyseren en verhelpen van productieproblemen.

3 Plan van aanpak

Gedurende de eerste twee weken ben ik bezig geweest met het opstellen van een plan van aanpak. Het eerste product voor mijn opdrachtgever is hierdoor tot stand gekomen. Het plan van aanpak is ontstaan door onder andere de gehanteerde methoden en technieken van het Digidoc Online team te inventariseren en kennis te maken met de omgeving waarbinnen dit project wordt uitgevoerd. In het plan van aanpak heb ik beschreven welke project-managementmethode en softwareontwikkelmethode er wordt toegepast, welke werkzaamheden worden uitgevoerd om de afgesproken producten op te leveren, welke afspraken zijn gemaakt, wat de scope is van dit project en welke risico's zich kunnen voordoen. Het plan van aanpak is na meerdere besprekingen goedgekeurd door mijn begeleider.

3.1 Scrum werkwijze

Er zijn diverse projectmanagementmethodes die toegepast kunnen worden om het project in goede banen te leiden. Het Digidoc Online team maakt gebruik van de Agile Scrum methodiek. Scrum is een Agile aanpak die op een incrementele en iteratieve wijze verloopt en voor het team mogelijk maakt om op een flexibele wijze problemen op te lossen of tegemoet te komen aan gebruikerswensen. Deze methode staat lijnrecht tegenover de traditionele waterval ontwikkelmethode, waarbij eerst alle wensen en eisen in kaart worden gebracht voordat er ook nog maar iets ontwikkeld is. Met Agile Scrum worden deze wensen en eisen steeds duidelijker gedurende de ontwikkeling van de producten. Gaande het project zal het eindresultaat zichzelf gaan vormen. Het voordeel hiervan is dat er stukje flexibiliteit is om snel te reageren op veranderingen in eisen, wensen en prioriteiten.

Dit project zal op een iteratieve en incrementele wijze worden uitgevoerd volgens de Scrum methodiek die het Digidoc Online team hanteert, met als doel een volwaardig product op te leveren die voldoet aan de acceptatiecriteria en wensen van de opdrachtgever. Omdat mijn werkzaamheden niet aansluiten bij de werkzaamheden die het Digidoc Online team uitvoert, worden er aparte sprints aangemaakt voor dit project. Deze sprints gaan gelijklopen met de sprints van het Digidoc Online team.

3.1.1 Ceremonies

Het Digidoc Online team werkt tijdens het Scrumproces in sprints van twee weken, waarin verschillende ceremonies plaats vinden. Elke dag vindt er een daily stand-up meeting plaats van maximaal 15 minuten waarin de werkzaamheden de komende 24 uur op elkaar afgestemd worden. De daily stand-ups kan ik goed gebruiken om vragen te stellen en afspraken te maken met collega's. Aan het begin van elke sprint vindt er samen met mijn begeleider

en de product owner een sprint planning plaats waarin het werk voor de aankomende sprint op basis van prioriteren wordt geplaatst op de sprint backlog. Aan het einde van elke sprint vindt een retrospective plaats om terug te kijken op wat er die sprint behaald is, wat er eventueel (niet) goed ging en welke eventuele leerpunten ik hieruit heb gehaald. Naast de vaste Scrum ceremonies vindt er wekelijks (op woensdag) een voortgangsgesprek van een uur plaats met mijn begeleider om de voortgang van het project te bespreken.

In onderstaande tabel **(Tabel 1: Scrum ceremonies / meetings)** in één overzicht de ceremonies / meetings met daarbij op welke momenten deze plaatsvinden en wie deelnemen aan de ceremonie.

Ceremonies / meetings	*Moment*	*Deelnemers*
Sprint planning	Begin van de sprint	• Begeleider • Product owner
Daily stand-up	Dagelijks	• Digidoc Online team
Sprint review	Einde van de sprint	• Stakeholder • Begeleider • Product owner
Retrospective	Einde van de sprint	• Begeleider • Product owner
Voortgangsgesprek	Wekelijks (op woensdag)	• Begeleider

Tabel 1: Scrum ceremonies / meetings

3.1.2 Afspraken

Omdat het niet gebruikelijk is om Scrum als individu uit te voeren, zijn de volgende afspraken gemaakt met mijn begeleider en product owner over de wijze van de Scrum uitvoering binnen dit project:

- Het project volgt het Scrum proces dat het Digidoc Online team hanteert.
- Ik neem deels de rol Scrum Master aan. Mijn taak is het in goede banen leiden van de projectuitvoering. Mijn begeleider neemt ook deels de rol Scrum Master aan. Zijn taak is het wegnemen van eventuele blokkades zodat ik zo min mogelijk gehinderd word tijdens de project uitvoering.
- Ik sluit aan bij de daily stand-ups van het Digidoc Online team.
- Elke week een voortgangsoverleg met mijn begeleider om de voortgang van het project te bespreken.
- Items op de sprint backlog die niet af zijn worden indien nodig meegenomen naar de volgende sprint.
- Mijn begeleider vervangt de product owner bij afwezigheid.
- Ik stel de user stories op en deze worden herzien door de product owner.
- De product owner beheert de product backlog en bepaald de volgorde van de uitvoering.

3.2 Werkzaamheden

In dit hoofdstuk worden de werkzaamheden beschreven die worden uitgevoerd tijdens dit project om de afgesproken producten op te leveren. Mijn begeleider heeft aangegeven om de methoden en technieken van het Digidoc Online team toe te passen. Hiervoor heb de gebruikte methoden en technieken van het Digidoc Online team geïnventariseerd door informatie te verzamelen van de team workspace in Confluence en gesprekken te voeren met collega's binnen dit team. Daarbij heb ik aangeven welke beroepstaak zal worden aangetoond bij het uitvoeren van de werkzaamheden.

- ***Analyseren probleemdomein*** - Voor dit project is het noodzakelijk dat er een analyse wordt uitgevoerd van het probleemdomein. De functionaliteiten die niet in de ACCE uitgevoerd kunnen worden, maar wel in de FEM beheerclient, worden in kaart gebracht. Daarbij wordt aangegeven welke functionaliteiten (niet) out-of-the-box kunnen worden gerealiseerd. De functionaliteiten worden vervolgens samen met de stakeholder geprioriteerd middels de MoSCoW methode en vertaald naar duidelijk omschreven user story omschrijvingen. Het uitgangspunt van deze analyse is een lijst met geprioriteerde eisen / wensen en user story omschrijvingen die voldoen aan de Definition of Ready. *(**Beroepstaak A-1:** Analyseren probleemdomein & opstellen probleemstelling)*

- ***Oriënteren naar de mogelijkheden van IBM Content Navigator –*** Voorafgaand het realisatieproces ga ik me oriënteren op de mogelijkheden van Content Navigator omtrent beheer en ontwikkeling. Tijdens deze oriëntatie ga ik mezelf verdiepen in de standaard FileNet web interface, Content Navigator, omdat dit voor mij een nieuwe, onbekende en complexe omgeving is. Daarbij is het doel van deze oriëntatie is om zoveel mogelijk kennis op te doen, zodat ik passende oplossingen kan bieden tijdens het realisatie proces van de niet out-of-the-box functionaliteiten en het uitvoeren van de product backlog refinement. *(**Beroepstaak G-f:** Leren leren: voorbereiden op volgende studiefase & beroep)*

- ***Product backlog refinement*** – Het toevoegen van een detaillering, inschatting en structuur aan de product backlog is één van de beschreven Scrum regels **(Agilescrumgroup, n.d.)** die het Digidoc Online toepast en ik ook ga toepassen voor dit project. Voor diverse user stories ga ik onderzoeken welke werkzaamheden uitgevoerd moeten worden om de user story te maken c.q. voltooien. De product owner zal bepalen welke user stories dat zijn. Hiervoor zal ik informatie moeten verzamelen, verwerken en uiteindelijk toepassen. *(**Beroepstaak G-f:** Leren leren: voorbereiden op volgende studiefase & beroep)*

- ***Ontwerpen*** - Voor de niet out-of-the-box functionaliteiten worden er verschillende UML-diagrammen opgesteld. De relaties en interactie tussen

onderlinge klassen worden hierin vastgelegd en gemaakte ontwerp keuzes worden toegelicht. *(**Beroepstaak C-1:** Ontwerpen software)*

- ***Ontwikkelen*** – Voor minimaal één niet out-of-the-box functionaliteit wordt er maatwerk ontwikkeld. Maatwerk voor Content Navigator bestaat uit verschillende lagen waarbij ik gebruik zal maken van de Content Navigator API. *(**Beroepstaak D-1:** Realiseren van software)*

- ***Testen*** – Voor de (niet) out-of-the-box functionaliteiten worden testscripts opgesteld om het gewenste gedrag functioneel in Content Navigator te testen. Voor de niet out-of-the-box functionaliteiten worden unittests geschreven die minimaal 50% code coverage borgen. De resultaten en bevindingen van de testscripts en unittests worden beschreven in een testverslag, gevolgd door een advies voor mijn opdrachtgever. *(**Beroepstaak D-2:** Testen & evalueren)*

- ***Configureren*** - De out-of-the-box functionaliteiten worden geconfigureerd in Content Navigator. Hiervoor zijn 3 werkdagen gereserveerd.

3.3 Definition of Ready / Done

Voor dit project heb ik in samenwerking met mijn begeleider een Definition of Ready en Definition of Done opgesteld om te bepalen wanneer een user story van de product backlog opgepakt kan worden en wanneer een user story daadwerkelijk is afgerond. Voor de Definition of Ready hebben we een aantal criteria punten vastgelegd en voor de Definition of Done hebben we een aantal kwaliteitseisen vastgelegd. Een aantal punten hiervan zijn overgenomen van de Definition of Ready en Definition of Done die het Digidoc Online team hanteert. Punt 1 en 2 van de Definition of Done zijn alleen van toepassing voor de niet out-of-the-box functionaliteiten. Mijn begeleider heeft aangegeven om tijdens het project te kijken of het nodig is om de Definition of Ready en Definition of Done waar nodig aan te passen. In onderstaande tabel **(Tabel 2: Definition of Ready en Definition of Done)** staan de criteria punten en kwaliteitseisen beschreven.

Definition of Ready *Een user story is 'Ready' en voldoet aan de volgende criteria:*	**Definition of Done** *Een user story is 'Done' en voldoet aan de volgende kwaliteitseisen:*
1. Duidelijke beschrijving van het verwachte of gewenste gedrag. 2. User story is gereviewd door de product owner. 3. Backlog traceability document is bijgewerkt. Naast de boven genoemde criteria worden de standaarden van **(Definition of Ready, 2017)** toegepast.	1. Source code is geschreven volgens de coding standaarden die zijn vastgelegd. 2. Er zijn unittests geschreven die voldoen aan de kwaliteitscriteria die zijn vastgelegd in het testplan. 3. Eventuele (technische) documentatie is bijgewerkt. 4. Mijn begeleider heeft een functionele UI-test uitgevoerd a.d.v. opgestelde testscripts.

Tabel 2: Definition of Ready en Definition of Done

3.4 Kwaliteit

Om de kwaliteit van de producten te waarborgen worden er verschillende technieken gebruikt. Dit zijn technieken die het Digidoc Online team ook gebruikt.

- ***Coding standaarden*** - Ik schrijf de code volgens de opgestelde coding standaarden van het Digidoc Online team. De kwaliteit van de code wordt hiermee gewaarborgd. Deze code standaarden zijn opgenomen als bijlage **(Bijlage I: Coding standaarden)**.

- ***Definition of Ready / Done*** - Een Definition of Done geeft een duidelijke omschrijving van hoe een opgeleverde product backlog item er uit moet zien. Het is een manier om de kwaliteit van opgeleverde producten binnen Scrum te borgen **(Agilescrumgroup, n.d.)**.

- ***Testen*** – Zowel de out-of-the-box als de niet out-of-the-box worden functioneel getest door mijn begeleider middels opgestelde testscripts. Voor de niet out-of-the-box functionaliteiten worden unittests geschreven om de code (unit) afzonderlijk te testen. De kwaliteitscriteria van de testsoorten worden beschreven in het testplan. Zo moet de unittests voldoen aan een minimaal code coverage van 50%. De kwaliteit van de functionaliteiten worden hierdoor gewaarborgd.

- ***Retrospective*** – Tijdens de retrospective wordt de huidige sprint geëvalueerd waarbij wordt gekeken wat de sprint heeft opgeleverd, wat (niet) goed ging, wat ik heb geleerd en welke leerpunten ik hieruit heb

gehaald om mee te nemen naar de volgende sprint. Hierdoor wordt de kwaliteit van het Scrum proces gewaarborgd.

- ***Voortgangsbewaking*** - Tijdens de wekelijkse voortgangsgesprek met mijn begeleider wordt er feedback geleverd op de gemaakte producten. Hierdoor sluiten de producten beter aan op het gewenste eindresultaat.

3.5 Tools en technieken

- ***Jira*** – Jira is een softwareontwikkelingstool voor Agile teams. Het Digidoc Online team gebruikt Jira om het Scrum proces inzichtelijk te maken. Jira ga ik ook gebruiken om het Scrum proces van dit project inzichtelijk te maken. In Jira kan ik de sprint / product backlog beheren en heb ik een apart Scrum board die ik kan gebruiken om de status van de taken tijdens een sprint bij te houden.

- ***Confluence*** – Confluence is een teamworkspace tool waarin kennis en samenwerkingen samenkomen. Deze tool gebruikt het Digidoc Online team om informatie met het team en de opdrachtgever (Doc-Direkt) te delen. Confluence heeft een koppeling met Jira om het Scrum proces voor de opdrachtgever inzichtelijk te maken. Confluence ga ik ook gebruiken voor dit project om informatie van dit project te delen met het team en kan ik informatie verzamelen wat betrekking heeft tot dit project. Voor dit project heb ik een eigen ruimte toegewezen gekregen in Confluence waarin ik mijn project documenten kan uploaden, beheren en de status van de Scrum taken inzichtelijk kan maken voor andere.

- ***Ontwikkelomgeving*** - Op mijn ontwikkelomgeving (VDI) zijn de benodigde software ontwikkeltools geïnstalleerd die mij ondersteunen bij het ontwikkelen van de niet out-of-the-box functionaliteiten voor de beheertooling in Content Navigator. Via mijn ontwikkelomgeving heb ik toegang tot Jira, Bitbucket, Confluence en de Content Navigator omgeving.

- ***Content Navigator –*** Content Navigator is de nieuwe standaard web interface van FileNet voor het beheren van ongestructureerde informatie. Binnen Content Navigator krijg ik een beheerdesktop toegewezen waarin de functionaliteiten gerealiseerd gaan worden.

- ***Eclipse*** – Eclipse is een IDE die mij tijdens het ontwikkelen van de niet out-of-the-box functionaliteiten helpt bij het schrijven van de code.

- ***Bitbucket*** – Bitbucket is een tool die gebruik maakt van het Git versiebeheersysteem. Op Bitbucket upload ik mijn geschreven code in een toegewezen repository.

- ***UML*** – UML is een modelmatige taal om objectgeoriënteerde analyses en ontwerpen voor een informatiesysteem te kunnen maken. UML ga ik

gebruiken om voor de niet out-of-the-box functionaliteiten UML-diagrammen op te stellen.

- ***JUnit*** – JUnit is een unit testing framework voor de Java programmeertaal die het Digidoc Online team ook gebruikt voor het testen van geschreven Java code. JUnit ga ik ook gebruiken om de geschreven Java code van de niet out-of-the-box functionaliteiten afzonderlijk te testen.

3.6 Projectscope

Tijdens dit project wordt er gewerkt in sprints van twee weken waarin er werkzaamheden worden uitgevoerd om de vooraf afgesproken producten op te leveren. De opdrachtgever heeft aangegeven om voor minimaal één niet out-of-the-box functionaliteit maatwerk te ontwikkelen. De product owner bepaald per sprint in welke volgorde de product backlog items worden opgepakt. Het is daarom van ter voren niet duidelijk over welke (niet) out-of-the-box functionaliteiten de beheertooling in Content Navigator zal beschikken wanneer het project is afgerond.

3.7 Projectorganisatie

De volgende personen zijn betrokken bij dit project. Daarbij heb ik aangegeven wat zijn functie is en welke rol de persoon heeft tijdens dit project.

Naam	*Functie op werk*	*Rol tijdens dit project*
Jeroen Hollander	Architect	Opdrachtgever, begeleider en vervangend product owner
Otto Smook	Product owner	Product owner
Tony van Leeuwen	Developer	Scrummaster en projectuitvoerder
Peter Verpoorten	Digidoc beheerder	Key stakeholder (aanspreek-punt vanuit het Digidoc beheerteam)

Tabel 3: Projectorganisatie

3.8 Projectrisico's

De volgende risico's kunnen zich voordoen tijdens dit project.

Risico	*Maatregelen*
Onvoldoende kennis / ervaring van de techniek	• Indien nodig de planning aanpassen. • Meer tijd besteden aan bestuderen van de techniek.

Langdurige ziekte	• Hulp vragen aan collega's die kennis hebben van de techniek. • De planning aanpassen indien de extra speling in de planning onvoldoende is.

Tabel 4: Projectrisico's

3.9 Op te leveren producten

De volgende producten worden opgeleverd:

- Het plan van aanpak;
- De analyse van het probleemdomein;
- De UML-diagrammen;
- Testdocumentatie (testplan, testscripts en testverslag);
- Maatwerk voor minimaal één niet out-of-the-box functionaliteit;
- De beheertooling in Content Navigator;

3.10 Planning

Onderstaande planning is opgesteld voor dit project. In de planning heb ik tegen het einde van dit project rekening gehouden met eventuele uitloop indien zich bijvoorbeeld risico's voordoen.

Data	*Activiteit*
07-09-2020 – 09-10-2020	Plan van aanpak schrijven Analyse uitvoeren van het probleemdomein Oriëntatie Content Navigator Voorbereiden ontwikkelomgeving
14-10-2020 – 28-10-2020	Sprint 1
28-10-2020 – 11-11-2020	Sprint 2
11-11-2020 – 25-11-2020	Sprint 3
25-11-2020 – 09-12-2020	Sprint 4
09-12-2020 – 23-12-2020	Sprint 5
23-12-2021 – 11-01-2021	Extra speling indien zich risico's voordoen Afronden afstudeerdossier

Tabel 5: Planning

4 Sprint 0 – Voorbereiden

Tijdens sprint 0 heb ik de eerste echte sprint (sprint 1) voorbereidt, waarin ik een analyse van het probleemdomein heb uitgevoerd en mezelf heb georiënteerd op de mogelijkheden van Content Navigator om passende oplossingen te bieden tijdens het realisatie proces. Ik ben gestart met uitvoeren een analyse van het probleemdomein om te achterhalen wat de functionele behoeftes zijn van de Digidoc beheerders en waar de prioriteiten liggen. Hiervoor heb ik verschillende bestaande analyse documenten aangeleverd gekregen en hebben er verschillende gesprekken plaatsgevonden met de stakeholder. De aangeleverde analyse documenten waren dusdanig verouderd dat het uitvoeren van een nieuwe analyse noodzakelijk was. Het van ter voren vastleggen van de eisen / wensen lijkt op een ouderwetse stap uit het waterval proces, maar dat is niet zo. Agile projecten bevinden zich binnen bepaalde grenzen en moeten ook voldoen aan opgestelde richtlijnen. Met het vaststellen van wensen en eisen heb ik deze grenzen bepaald. Door niets te definiëren, komt dit niet ten goede van de kwaliteit van de producten. Sprint 0 heeft net zoals de andere sprints een vaste time-box van twee weken. Ik heb tijdens sprint 0 de balans gehouden tussen het vooraf definiëren van de functionele behoeftes en het starten zonder enige vorm van houvast. Op die manier blijf ik op een Agile manier werken en is er ruimte voor verandering. Naast de analyse en oriëntatie heb ik tegen het einde van de sprint mijn ontwikkelomgeving ingericht a.d.v. een bestaand stappenplan die het Digidoc Online team heeft opgesteld.

4.1 Analyseren van het probleemdomein

Voorafgaand heb ik een analyse uitgevoerd van het probleemdomein om te achterhalen wat de functionele behoeftes zijn van de stakeholder. De functionaliteiten die niet in de ACCE, maar wel in de FEM beheerclient uitgevoerd kunnen worden, zijn in kaart gebracht en geprioriteerd middels de MoSCoW methode. Vervolgens zijn de functionele behoeftes vertaald naar duidelijk omschreven user stories die voldoen aan de Definition of Ready. Mijn begeleider heeft veel kennis van Content Navigator en kon van ter voren al een inschatting maken welke functionaliteiten (niet) out-of-the-box gerealiseerd kunnen worden.

4.1.1 Totstandkoming

Het analyse probleemdomein document is tot stand gekomen door de verschillende analyse documenten die ik aangeleverd heb gekregen door de stakeholder. Deze analyse documenten heb ik opgenomen als bijlage **(Bijlage L: Aangeleverde analysedocumenten)**. Daarnaast hebben er verschillende gesprekken plaatsgevonden met de stakeholder waarin de analyse documenten zijn besproken. Een korte samenvatting van deze gesprekken zijn opgenomen als bijlage **(Bijlage K: Gesprekken).**

4.1.2 Functionele behoeftes

Zoals beschreven in de opdrachtomschrijving wordt de FEM beheerclient nog steeds gebruikt, omdat de beheerders van Digidoc met de nieuwe webbased ACCE beheerclient niet alle beheeractiviteiten kunnen uitvoeren die zij met de FEM beheerclient wel kunnen uitvoeren. In onderstaande tabel **(Tabel 6: Functionaliteiten die ontbreken in de ACCE)** staan de functionaliteiten beschreven die deels of helemaal niet beschikbaar zijn in de ACCE, maar wel in de FEM beheerclient. Elke functionaliteit is een functionele behoefte vanuit het oogpunt van de stakeholder. De functionaliteiten heb ik exact beschreven zoals de stakeholder dit beschrijft. De functionaliteiten met een * aan het begin wordt door de stakeholder als relatief belangrijk gezien. De stakeholder heeft de functionaliteiten verdeeld in verschillende secties.

ID	*Documenten, custom objects en folders*
1	Copy document
2	Copy custom object (resulteert in file in folder, geen kopie)
3	Copy folder (maakt kopie van de folderboom incl objecten, muv custom objects (worden gefiled in folder)
4	* Delete folder, geeft opties: • delete folders and containment relationships only • delete content of contained versions • delete content of all versions of all contained objects
5	Copy/paste dmv verslepen en ctl-c/v
6	* Bij browse en resultaat query sortering op kolommen mogelijk
7	* Bij browse wordt aantal subfolders en objecten vermeld bij geslecteerde folder in linker panel
8	* Bij browse naar het begin of einde springen
9	Bij (un)file from/in folder via browse de folder te selecteren
10	Create document: gaat snel na specificeren claas+properties
11	* Create folder, properties worden tijdens het creëren gespecificeerd
12	Containment naam kan gewijzigd worden
13	Op een pagina worden veel meer objecten getoond dan bij de ACCE, waardoor er veel minder gescrold hoeft te worden
14	Na het omlaag scrollen in een pagina blijft na het uitvoeren van een actie op een object de positie in de pagina ongewijzigd
	Search
15	Laatste zoekopdracht wordt bewaard
16	* Aantal gevonden objecten wordt vermeld
17	Voortgangsindicator

18	* Aantal te tonen objecten in resultset is te specificeren (aantal gevonden objecten wordt eveneens getoond)
19	* Ieder kolom is te sorteren
20	* In resultset: select all (ctl-a)
	Security
21	* Details inheritad security zichtbaar
22	Details overige security vormen direct zichtbaar bij selectie groep of account
23	* Security Policies kunnen worden gedefinieerd en onderhouden
	Overige
24	Springen in selectielijsten door typen letter

Tabel 6: Functionaliteiten die ontbreken in de ACCE

Momenteel maken de eindgebruikers nog gebruik van een oudere versie van ACCE (v5.5.0). Binnenkort stappen ze over naar de nieuwe versie van ACCE (v5.5.3) die een klein deel van de ontbrekende functionaliteiten in ACCE afdekt. In de nieuwe versie van ACCE is er een significante verbetering ten opzichte van de eerdere versie van ACCE. In onderstaande tabel **(Tabel 7: Verbetering van de functionaliteiten in ACCE versie 5.5.3)** heb ik beschreven welke verbeteringen er zijn doorgevoerd in de nieuwe versie van ACCE.

ID	*Functionaliteit*	*Verbetering in ACCE versie 5.5.3*
4	* Delete folder, geeft opties: • delete folders and containment relationships only • delete content of contained versions • delete content of all versions of all contained objects	Options zijn selecteer baar indien 'verwijderen' vanuit actions in hoofdmenu wordt gekozen na (sub)selectie zoekresultaat
6	* Bij browse en resultaat query sortering op kolommen mogelijk	Sortering niet mogelijk bij browse maar nu wel bij zoekresultaten
16	* Aantal gevonden objecten wordt vermeld	Aantal wordt vermeld, wel is er sprake van paging bij het ophalen van de resultaten. Indien vinkje wordt gezet bij 'selecteer alle zoekresultaten' dan worden alle items opgehaald en het bijbehorende aantal afgebeeld. Max 50.000
18	* Aantal te tonen objecten in resultset is te specificeren (aantal gevonden objecten wordt eveneens getoond)	Geen onderscheid tussen te tonen items en aantal items dat voldoet aan query.

		Aantallen wel inzichtelijk te maken (zie punt 13)
19	* Ieder kolom is te sorteren	Sortering op alle kolommen is mogelijk
20	* In resultset: select all (ctl-a)	Alles selecteren kan via vinkje 'selecteer alle zoekresultaten'. Kan niet via ctrl-a maar dat is logisch in webpplicatie.
21	* Details inherited security zichtbaar	Is mogelijk de details te zien. Ook voor inherited security. Kan zelfs direct i.p.v geinherit gemaakt worden.

Tabel 7: Verbetering van de functionaliteiten in ACCE versie 5.5.3

De stakeholder heeft aangegeven dat de verbeteringen in de nieuwe versie van ACCE voor functionaliteit nummer 4, 6, 19, 20 en 21 dusdanig zijn verbeterd dat deze als lage prioriteit opgenomen konden worden. Momenteel wordt er nog gewerkt met de oudere versie van ACCE. Echter op korte termijn zal er gewerkt worden met de nieuwe versie van ACCE. Deze laatste versie dekt dus een deel van de ontbrekende functionaliteiten in ACCE af, maar niet allemaal. De in de ACCE versie 5.5.3 ontbrekende functionaliteiten die wel in de FEM beheerclient kunnen worden uitgevoerd worden geprioriteerd en vertaald naar duidelijk omschreven user stories.

4.1.3 Prioriteren met de MoSCoW-methode

De functionaliteiten heeft de stakeholder geprioriteerd met een score van 0 t/m 5. Waarbij de score 0 kan komen te vervallen, 1 minst belangrijk en 5 meest belangrijk is. Op basis van deze scores heeft mijn begeleider en de stakeholder bepaald om de score 0 op te nemen als Won't Have, score 1-2 als Could Have, score 3-4 als Should Have en score 5 als Must Have. Op basis van een nummering is te herleiden welke functionaliteit het betreft.

Must Haves

ID	*Functionaliteit*
7	* Bij browse wordt aantal subfolders en objecten vermeld bij geslecteerde folder in linker panel
8	* Bij browse naar het begin of einde springen
11	* Create folder, properties worden tijdens het creëren gespecificeerd
23	* Security Policies kunnen worden gedefinieerd en onderhouden

Should Haves

ID	*Functionaliteit*
12	Containment naam kan gewijzigd worden
14	Na het omlaag scrollen in een pagina blijft na het uitvoeren van een actie op een object de positie in de pagina ongewijzigd

16	* Aantal gevonden objecten wordt vermeld
18	* Aantal te tonen objecten in resultset is te specificeren (aantal gevonden objecten wordt eveneens getoond)

Could Haves

ID	*Functionaliteit*
3	Copy folder (maakt kopie van de folderboom incl objecten, muv custom objects (worden gefiled in folder)
15	Laatste zoekopdracht wordt bewaard
1	Copy document
6	Sorteren op kolommen bij een browse en query resultaat
9	Bij (un)file from/in folder via browse de folder te selecteren
10	Create document: gaat snel na specificeren class+properties
13	Op een pagina worden veel meer objecten getoond dan bij de ACCE, waardoor er veel minder gescrold hoeft te worden
17	Voortgangsindicator
22	Details overige security vormen direct zichtbaar bij selectie groep of account
24	Springen in selectielijsten door typen letter

Won't Haves

ID	*Functionaliteit*
2	Copy custom object (resulteert in file in folder, geen kopie)
4	* Delete folder, geeft opties: delete folders and containment relationships only, delete folders and containment relationships only, delete content of contained versions • delete folders and containment relationships only • delete content of contained versions • delete content of all versions of all contained objects
19	* Ieder kolom sorteren
20	In resultset: select all (ctl-a)
21	Details inherited security zichtbaar
5	Copy/paste dmv verslepen en ctl-c/v

4.1.4 Bepalen van de functionele scope

De Must Have en Should Have niet out-of-the-box functionaliteiten vallen binnen de scope van dit project. Daarbij vallen alle out-of-the-box functionaliteiten ook binnen de scope van dit project. Dit is besloten in samenspraak met mijn begeleider, product owner en stakeholder. Met de opdrachtgever is afgesproken om voor minimaal één niet out-of-the-box functionaliteit maatwerk te ontwikkelen. De product owner bepaald op basis van prioriteit welke niet out-the-box functionaliteit(en) dat worden.

4.1.5 Vertaling naar user story omschrijvingen

De functionaliteiten die zijn beschreven en geprioriteerd zijn vertaald naar een duidelijk omschrijving van het verwachte of gewenste gedrag. Dit is één van de criteria van de Definition of Ready.

In onderstaande tabel **(Tabel 8: User story omschrijvingen)** is te zien hoe de functionaliteiten, beschreven in het vorige hoofdstuk, zijn vertaald naar een user story omschrijving. Daarbij is het nummer van de functionaliteit toegevoegd om te achterhalen aan welke functionaliteit(en) de user story is gekoppeld. Deze koppeling is ook bijgewerkt in het backlog traceability document **(Bijlage H: Backlog traceability).** Ik heb ook een extra kolom (OOTB) toegevoegd waarin ik heb aangegeven of de user story out-of-the-box gerealiseerd kan worden. Hiervoor heb ik de IBM documentatie gebruikt en voor elke functionaliteit gekeken wat de mogelijkheden zijn om te realiseren. Ook heeft mijn begeleider mij hierbij geassisteerd. Hij heeft veel ervaring met de FileNet omgeving en ook met Content Navigator en kon daarom van ter voren al grotendeels bepalen welke user stories out-of-the-box geïmplementeerd konden worden.

	User story omschrijving	*OOTB*
7	Als beheerder wil ik bij een browse het aantal objecten kunnen zien bij een geselecteerde folder	Nee
7	Als beheerder wil ik bij een browse het aantal sub folders kunnen zien bij een geselecteerde folder	Nee
8	Als beheerder wil bij een browse naar begin of einde kunnen springen	Nee
11	Als beheerder wil ik tijdens het creëren van een folder alle folder properties kunnen specificeren	Ja
23	Als beheerder wil ik Security Policies kunnen definiëren en onderhouden	Nee
12	Als beheerder wil ik de containment naam kunnen wijzigen	Nee
14	Als beheerder wil ik dat na het omlaag scrollen in een pagina het uitvoeren van een actie op een object de positie in de pagina ongewijzigd blijft	Nee
16	Als beheerder wil ik bij het zoeken het aantal gevonden objecten kunnen zien	Nee
18	Als beheerder wil ik bij het zoeken het aantal te tonen objecten in een resultset kunnen specificeren	Nee
1	Als beheerder wil ik documenten kunnen kopiëren	Ja
10	Als beheerder wil ik tijdens het creëren van een document alle document properties kunnen specificeren	Ja
5	Als beheerder wil ik kunnen kopiëren en plakken d.m.v. verslepen	Ja
5	Als beheerder wil ik kunnen kopiëren en plakken d.m.v. ctrl-c/v	Ja

Tabel 8: User story omschrijvingen

4.2 Oriënteren op de mogelijkheden van Content Navigator

Voorafgaand aan het realisatieproces heb ik mezelf georiënteerd op de mogelijkheden van de standaard FileNet web interface (Content Navigator), omdat dit voor mij een nieuwe, onbekende en wat mij duidelijk is gemaakt door mijn begeleider en collega's vooral een complexe omgeving is. Het doel van deze oriëntatie is om zoveel mogelijk kennis op te doen, zodat ik passende oplossingen kan bieden tijdens het realiseren van minimaal één niet out-of-the-box functionaliteit. In dit hoofdstuk beschrijf ik de belangrijke punten die ik tijdens het oriëntatie proces ben tegengekomen.

4.2.1 Bepalen van mijn zoekgebied

Mijn begeleider heeft veel ervaring met FileNet producten omtrent beheer en ontwikkeling en heeft mij geadviseerd om de informatie over Content Navigator op te zoeken in de documentatie van IBM. Content Navigator is volgens mijn begeleider goed gedocumenteerd en bevat alle informatie die ik nodig omtrent beheer en ontwikkeling van Content Navigator. Daarnaast heeft een collega van het DigiJust DevOps team een website gemaakt met blogs waarin hij uitleg geeft over verschillende onderwerpen omtrent ontwikkelen en beheer in Content Navigator. Hij is veel ervaring op het gebied van Content Navigator. Ik heb me dan ook georiënteerd binnen de documentatie van IBM **(IBM, 2020)** en de blogs van Ivo Jonker **(Jonker, 2017)**.

4.2.2 Werking van Content Navigator

Met de applicatie Content Navigator is het voor beheerders van diverse documentmanagementsystemen mogelijk om ongestructureerde informatie te beheren middels een web interface (Content Navigator). Denk hierbij aan het zoeken naar verschillende documenten en mappen. Deze informatie wordt opgeslagen in één of meerdere FileNet Repositories. Er zijn verschillende soorten typen repositories, afhankelijk van de behoeftes van het bedrijf. ECM FileNet maakt gebruik van de IBM FileNet P8 repository (FileNet Content Manager), omdat deze repository goed aansluit in de behoeftes van het ECM FileNet team. Deze type repository biedt namelijk meer mogelijkheden in vergelijking met andere repositories. Een voorbeeld is dat de FileNet Content Manager repository kan browsen naar content die zijn opgeslagen in deze repository. Een ander voorbeeld is dat het mogelijk is om documenten, folders en andere content op te slaan als favoriet. Dit zijn voorbeelden van features die de gebruiker met de Content Navigator web client kan uitvoeren. Een belangrijke en meest gebruikte feature van Content Navigator is het zoeken naar documenten en folders **(IBM, 2020).**

4.2.3 Architectuur van Content Navigator

Content Navigator is gebaseerd op het model-view-controller (MVC) architectuur. In de afbeelding **(Figuur 2: Architectuur van Content Navigator)** zijn de lagen van het MVC-structuur te zien **(IBM, 2020)**.

Model-view-controller (MVC): Dit is een ontwerppatroon dat het ontwerp van complexe toepassingen opdeelt in drie eenheden met verschillende verantwoordelijkheden.

Content Navigator bestaat uit de volgende lagen:

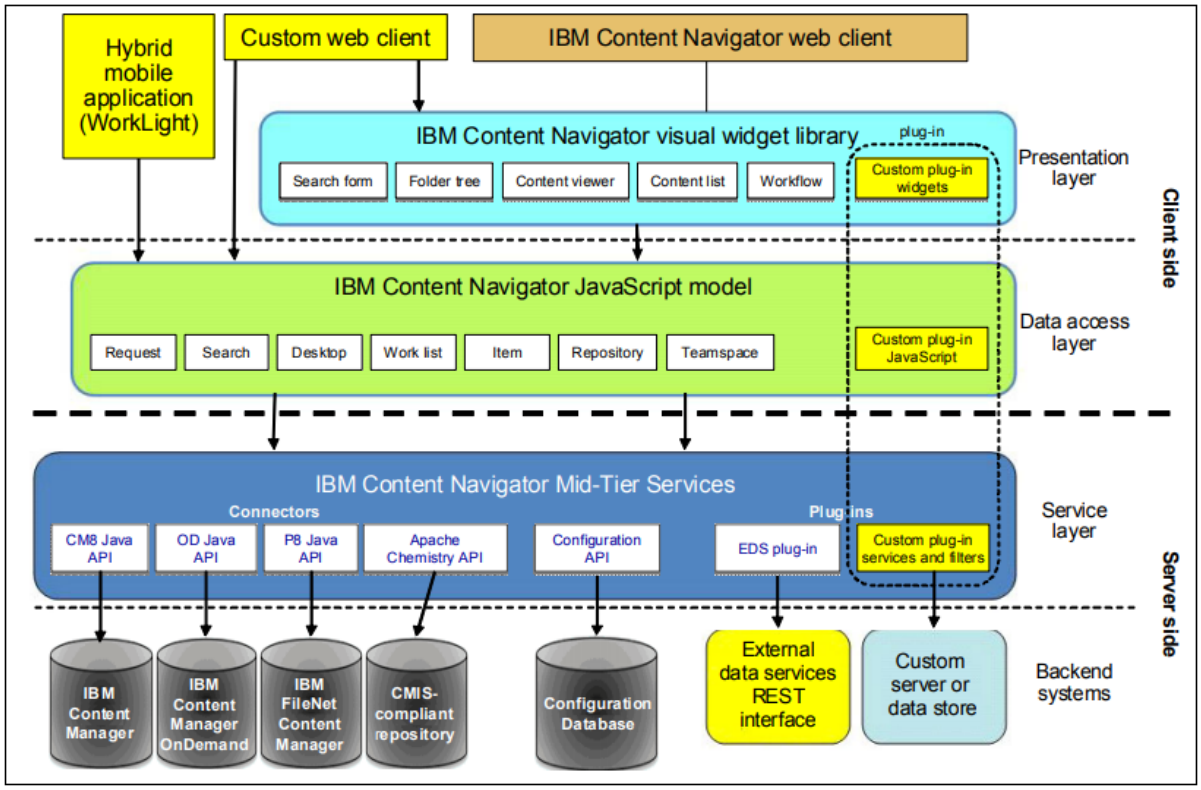
- **Content Navigator web client (controller)** Content Navigator (controller) is de web client waarmee de gebruiker met de diverse widgets interacteert. De web client fungeert als controller en is verantwoordelijk voor de communicatie tussen de view en model.

- **Presentatie laag (view)** De presentatie laag (view) bestaat uit diverse Dojo widgets waarmee de gebruiker kan interacteren. Het belangrijkste component van de presentatie laag is de visual widget library van de Content Navigator API. In deze library zijn een set aan widgets beschikbaar die gebruikt kunnen worden. Daarnaast is het ook mogelijk om de widgets van de Dojo Toolkit te implementeren.

- **Data access laag (model)** De data access laag (model) is verantwoordelijk voor het bijhouden van de applicatie data en voor de communicatie met de service laag. Middels event listeners is het mogelijk om wijzigingen aan te brengen in de model. Het belangrijkste component van de data access laag is de model library van de Content Navigator API. Hierin staan verschillende model klassen die geïmplementeerd kunnen worden.

- **Service laag (service)** De service laag (service) is verantwoordelijk voor de communicatie met de onderliggende FileNet repositories. Met een service is het mogelijk om de benodigde data op te halen uit de FileNet repository.

- **Backend laag (FileNet repository)** In de backend laag bevinden zich de FileNet repositories, databases en externe service interfaces.



Figuur 2: Architectuur van Content Navigator

In de architectuur afbeelding is te zien dat er op elke laag een 'custom plugin' geïmplementeerd kan worden. Op het moment dat ik de architectuur afbeelding zag, dacht ik dat er op elke laag een aparte plugin geïmplementeerd moet worden. Echter blijkt dit niet zo te zijn. Het is mogelijk om één custom plugin te ontwikkelen om één of meerdere lagen naar eigen wens in te richten.

Op elk van de genoemde lagen is het mogelijk om een plugin te implementeren om functionaliteiten uit te breiden of toe te voegen in Content Navigator, zoals een maatwerk widget, feature, lay-out, service of een ander component. Dit is ook te zien in de afbeelding **(Figuur 2: Architectuur van Content Navigator)**, waarin op elke laag is afgebeeld dat er een 'custom plugin' toegevoegd kan worden. Als de standaard functionaliteiten van Content Navigator niet voldoen aan de wensen / eisen van de gebruiker, dan is het met de Content Navigator API mogelijk om maatwerk te bouwen in de vorm van een plugin en de plugin vervolgens te deployen in Content Navigator. Via een desktop omgeving kan de plugin worden aangeboden voor de gebruiker.

4.2.4 Content Navigator API

De Content Navigator API bestaat uit de volgende packages en libraries:

- **Java API** – Bevat verschillende Java packages met daarin klassen die geïmplementeerd kunnen worden voor het ontwikkelen van een plugin. De volgende **Java packages** kunnen hiervoor worden gebruikt:
  - **com.ibm.ecm.configuration** – Bevat klassen voor het benaderen en wijzigen van de configuratie van de database (deze heb ik waarschijnlijk niet nodig)
  - **com.ibm.ecm.configuration.exception** -
  - **com.ibm.ecm.extension** – Bevat abstracte klassen die ik kan implementeren voor het ontwikkelen van een plugin
  - **com.ibm.ecm.json** – Bevat klassen voor het werken met JSON requests en responses.
  - **com.ibm.json.java** – Bevat klassen van de JSON4J toolkit (deze heb ik waarschijnlijk niet nodig)

- **JavaScript API** – Bevat de JavaScript klassen voor het maken van een model en widget. Hiervoor wordt gebruik gemaakt van de Dojo JavaScript library. In de documentatie van de Content Navigator API zijn deze klassen onderverdeeld in de modeling library en visual widget catalogus.

API: Application Programming Interface maakt de communicatie tussen verschillende applicaties mogelijk. In dit geval biedt de Content Navigator applicatie een API (interface) aan die een andere applicatie (plugin) kan gebruiken om te kunnen communiceren met Content Navigator.

4.2.5 Conclusie oriëntatie

Tijdens deze oriëntatie fase heb ik voor mijn gevoel een goede basiskennis gelegd en nieuwe inzichten gecreëerd om tijdens het ontwikkelen van de niet out-of-the-box functionaliteiten passende oplossingen te bieden omtrent de ontwikkeling van Content Navigator plugins. Ik heb kennis gemaakt met de werking, architectuur en ontwikkelmogelijkheden van Content Navigator. Voorafgaand het project heb ik aangegeven in het plan van aanpak dat er maatwerk ontwikkeld moet worden voor minimaal één niet out-of-the-box functionaliteit. Dit is mogelijk door een op maat gemaakte plugin te ontwikkelen voor Content Navigator. Een plugin wordt ontwikkeld m.b.v. de programmeertalen Java en JavaScript (Dojo Toolkit). Waarbij er gebruik wordt gemaakt van de API van Content Navigator.

4.3 Configureren van mijn ontwikkelomgeving

Aan het einde van deze sprint heb ik mijn ontwikkelomgeving geconfigureerd. Mijn begeleider heeft mij geadviseerd om het stappenplan te volgen die het

Digidoc Online team heeft gemaakt in Confluence. Hierin staat beschreven hoe ik tools als Bitbucket, Jira en Confluence goed kon instellen en welke Java versie ik nodig is.

4.4 Retrospective

Wat heeft deze sprint opgeleverd?

- Analysedocument (analyse van het probleemdomein).
    - Huidige situatie / functionele behoeftes zijn in kaart gebracht.
    - Verbeteringen in de nieuwe versie van de ACCE ten opzichte van de oudere versie zijn in kaart gebracht.
    - Functionele behoeftes zijn geprioriteerd.
    - Functionele behoeftes vertaald naar user story omschrijvingen.
- Nieuwe kennis omtrent ontwikkeling binnen Content Navigator.
- Nieuwe kennis over het ontwikkelen van maatwerk in de vorm van een plugin voor Content Navigator.
- Geconfigureerde ontwikkelomgeving met de benodigde tools voor het ontwikkelen van de niet out-of-the-box functionaliteiten.

Wat ging er goed?

- Communicatie met de stakeholder.
- Toegang tot de benodigde tools (Jira en Confluence) was snel geregeld.
- Goed in kaart kunnen brengen wat de functionele behoeftes zijn van de stakeholder.
- Ik heb voor mijn gevoel een goede basiskennis kunnen leggen van Content Navigator tijdens de oriëntatie.

Wat ging minder goed?

- Ik merk dat Content Navigator een vrij complexe omgeving is met veel mogelijkheden. Het was zeker in het begin even uitzoeken waar ik de juiste informatie vandaan kon halen.
- Het was even wennen om de collega's alleen online te spreken. Het even langs lopen bij een collega is op sommige momenten handig.

Wat zijn de leerpunten?

- Sneller aan de bel trekken als ik met een vraag zit.
- De redbook van IBM Content Navigator bevat relevante informatie omtrent ontwikkeling. Deze redbook kan ik gebruiken voor de vervolg sprints.

5 Sprint 1 – Product Backlog Refinement

Aan het begin van sprint 1 heb ik samen met de product owner de product backlog vastgelegd in Jira en Confluence. Vervolgens heeft de product owner de user stories op de product backlog geprioriteerd (lowest t/m highest). Tijdens de sprint uitvoering heb ik een product backlog refinement uitgevoerd. De product backlog refinement is één van de beschreven Scrum regels **(Agilescrumgroup, n.d.)** waarbij de user stories met een hoge prioriteit een detaillering, inschatting en structuur wordt aangebracht. Dit is een belangrijk Scrum proces binnen het Digidoc Online team. Ik heb voor de user stories uitgezocht welke werkzaamheden moeten worden uitgevoerd om de user story te maken c.q. voltooien. Niet alle work break downs die ik heb uitgevoerd zijn in dit verslag opgenomen, maar als bijlage **(Bijlage J: Work break downs)**. De work break downs voor de niet out-of-the-box functionaliteiten heb ik in dit verslag opgenomen.

5.1 Vastleggen van de product backlog

Op basis van de resultaten van de analyse van het probleemdomein zijn de user stories de product backlog vastgelegd in Jira. De product owner heeft de user stories geprioriteerd. In onderstaande tabel **(Tabel 9: Product backlog)** is de product backlog (gefilterd op prioriteit) te zien. De product backlog is ook opgenomen in de backlog document **(Bijlage H: Backlog traceability)** met daarbij een verwijzing naar de functionaliteiten.

Jira ID	*Item (user story)*	*Prioriteit*
DD2-4614	Als beheerder wil ik documenten kunnen kopiëren en plakken d.m.v. ctrl-c/v	Highest
DD2-4602	Als beheerder wil ik tijdens het creëren van een folder alle properties kunnen specificeren	High
DD2-4613	Als beheerder wil ik documenten kunnen kopiëren en plakken d.m.v. verslepen	High
DD2-4612	Als beheerder wil ik tijdens het creëren van een document alle properties kunnen specificeren	High
DD2-4600	Als beheerder wil ik bij een browse het aantal sub folders kunnen zijn bij een geselecteerde folder	Medium
DD2-4604	Als beheerder wil ik de containment naam kunnen wijzigen	Medium
DD2-4606	Als beheerder wil ik bij het zoeken het aantal gevonden objecten kunnen zien	Medium
DD2-4607	Als beheerder wil ik bij het zoeken het aantal te tonen objecten in een resultset kunnen specificeren	Medium
DD2-4601	Als beheerder wil ik bij een browse naar begin of einde kunnen springen	Medium

DD2-4603	Als beheerder wil ik Security Policies kunnen definiëren en onderhouden	Low
DD2-4605	Als beheerder wil ik dat na het omlaag scrollen in een pagina het uitvoeren van een actie op een object de positie in de pagina ongewijzigd blijft	Low
DD2-4511	Als beheerder wil ik bij een browse het aantal objecten kunnen zien bij een geselecteerde folder	Low
DD2-4608	Als beheerder wil ik documenten kunnen kopiëren	Lowest

Tabel 9: Product backlog

5.2 Kwaliteitscontrole van de user stories a.d.v. de Definition of Ready

De kwaliteit van de user stories op de product backlog zijn aan de hand van de Definition of Ready **(Tabel 2: Definition of Ready en Definition of Done)** op kwaliteit gecontroleerd. De volgende criteria punten heb ik hiermee afgevinkt:

- ✓ Beschrijving van het verwachte of gewenste gedrag. Elke user story beschrijft het gewenste gedrag. Daarbij heb ik de omschrijving structuur overgenomen van bestaande user stories van het Digidoc Online team.
- ✓ User story is gereviewd door de product owner.
- ✓ Backlog traceability (**Bijlage H: Backlog traceability**) document bijgewerkt. User stories toegevoegd aan het backlog document.

5.3 Toekennen van story points

De hoeveelheid werk van een product backlog item wordt ingeschat op basis van story points. De zwaarte van de story points is bepaald door alle teams van de ECM FileNet afdeling in het algemeen en niet specifiek voor dit project. In onderstaande tabel **(Tabel 10: Story points)** is te zien hoe de inschatting van de hoeveelheid werk gekoppeld is aan een hoeveelheid story points.

Story points	*Omschrijving*
0	Geen werk
0.5	Een beetje werk
1	Een klusje
2	Net geen dag
3	Iets meer dan een dag
5	Groot deel van de week
8	Een week
15	Work break down maken

Tabel 10: Story points

5.4 Sprint planning

Tijdens de sprint planning zijn de user stories geprioriteerd en geplaatst op de product backlog door de product owner. Daarbij zijn een set aan user stories op de refinement product backlog geplaatst om een detaillering, inschatting en structuur te geven aan de user story. Daarvoor is er voor elke user story op de refinement product backlog een taak aangemaakt in Jira om een zogeheten work break down uit te voeren. Het doel van een work break down is om te onderzoeken welke werkzaamheden er uitgevoerd moeten worden om de user story te maken c.q. voltooien en welke randvoorwaarden van toepassing zijn. Wanneer de werkzaamheden voor maken c.q. voltooien van een user story zijn beschreven en zijn gereviewd door mijn begeleider, worden er vervolgens taken aangemaakt en gekoppeld aan de user story waarvoor een work break down is gemaakt. Een work break down is voornamelijk bedoeld om een inschatting te maken van het hoeveelheid werk die ik moet uitvoeren om de user story te kunnen voltooien.

De items in onderstaande tabel **(Tabel 11: Sprint backlog van sprint 1)** zijn door de product owner op de sprint backlog geplaatst en worden deze sprint opgepakt. Het backlog traceability Excel document **(Bijlage H: Backlog traceability)** heb ik bijgewerkt.

Jira ID	*Item (Taak)*	*Prioriteit*	*Points*
DD2-4611	Work break down voor user story DD2-4602	Highest	1
DD2-4616	Work break down voor user story DD2-4612	High	1
DD2-4615	Work break down voor user story DD2-4614	Medium	1
DD2-4617	Work break down voor user story DD2-4613	Medium	1
DD2-4618	Work break down voor user story DD2-4607	Medium	5
DD2-4619	Work break down voor user story DD2-4606	Medium	5
DD2-4623	Beheerserver DDOT beschikbaar maken voor Tony op het beheerportaal	Highest	1
DD2-4616	Content Navigator configureren op de O8 ontwikkelomgeving	High	1

Tabel 11: Sprint backlog van sprint 1

5.5 Uitvoeren van de work break downs

De product owner heeft aangegeven om voor de niet out-of-the-box functionaliteiten die betrekking hebben tot het uitvoeren van zoek acties maatwerk te ontwikkelen. Het gaat om user story DD2-4606 en DD2-4607. De product owner heeft zijn keuze gebaseerd op prioriteit en functionaliteit. De zoek functionaliteit is voor de stakeholder één van de belangrijkere en meest gebruikte functies in de huidige FileNet Enterprise Manger (FEM). Beide user stories hebben betrekking tot het uitvoeren van een zoek actie in Content Navigator. Door de kennis van mijn begeleider kon hij van ter voren bepalen dat

deze twee functionaliteiten niet out-of-the-box geïmplementeerd kunnen worden. Zie onderstaande tabel voor de omschrijving van deze twee user stories.

DD2-4606	Als beheerder wil ik bij het zoeken het aantal gevonden objecten kunnen zien	Medium
DD2-4607	Als beheerder wil ik bij het zoeken het aantal te tonen objecten in een resultset kunnen specificeren	Medium

In dit hoofdstuk heb ik de uitvoering van de work break downs beschreven voor deze twee user stories. De overige work break downs heb ik opgenomen in de bijlage **(Bijlage J: Work break downs).**

5.5.1 User story DD2-4606

Voor de user story DD2-4606 "*Als beheerder wil ik bij het zoeken het aantal gevonden objecten kunnen zien*" heb ik een work break down uitgevoerd en beschreven in dit hoofdstuk.

Gesprek met de stakeholder

Voor deze user story heb ik een gesprek gevoerd met een stakeholder met de vraag of deze user story verder toegelicht kan worden en daarbij stapsgewijs uit te leggen welke acties momenteel worden uitgevoerd in de FEM beheerclient. De stakeholder heeft het volgende toegelicht:

Ik wil kunnen aangeven hoeveel resultaten ik wil zien bij het uitvoeren van een zoekactie naar bijv. een documentnaam uit 2012. Met een zelf ingevoerde waarde. Daarbij wil ik ook kunnen zien hoeveel resultaten het totaal zijn.
Voor dit stukje toelichting "Ik wil kunnen aangeven hoeveel resultaten ik wil zien bij het uitvoeren van een zoekactie naar bijv. een documentnaam uit 2012" is al een user story (DD2-4607) aangemaakt, waarvoor ik in deze sprint ook een work break maak. Bij deze user story gaat het er om dat de stakeholder wil kunnen zien hoeveel resultaten een zoekactie teruggeeft.

Standaard beschikt Content Navigator over een zoek functionaliteit die het mogelijk maakt om in Content Navigator te zoeken naar ongestructureerde informatie. Met een administration tool is het ook mogelijk om de standaard zoek functionaliteit aan te passen, maar dat is wel beperkt **(IBM, 2020)**. In de documentatie kon ik niets vinden of het mogelijk is om bij het zoeken het totaal aantal objecten te tonen. Mijn begeleider heeft dat bevestigd waarbij hij heeft aangegeven dat de standaard zoek functionaliteit niet beschikt over de mogelijkheid om het aantal gevonden objecten van een zoekactie te tonen. De standaard zoek functionaliteit valt dan ook af, omdat het daarmee niet mogelijk is om de user story te maken en voltooien.

Op basis van oriëntatie naar de mogelijkheden van Content Navigator dient er voor deze functionaliteit een plugin ontwikkeld te worden, omdat de standaard feature van Content Navigator het niet mogelijk maakt om deze zoek functionaliteit te implementeren.

Werkzaamheden

Onderstaande werkzaamheden dienen uitgevoerd te worden om deze user story te maken c.q. voltooien:

- Ontwerpen van de plugin.
- Ontwikkelen van maatwerk in de vorm van een plugin.
- Functioneel testen van de functionaliteit / plugin.
- Unittests schrijven.
- Plugin beschikbaar stellen in Content Navigator (build & deploy).

Randvoorwaarden

- Content Navigator omgeving beschikbaar.
- Kennis van Content Navigator omtrent ontwikkeling.

5.5.2 User story DD2-4607

Voor de user story DD2-4607 "*Als beheerder wil ik bij het zoeken het aantal te tonen objecten in een resultset kunnen specificeren* " heb ik een work break down uitgevoerd en beschreven in dit hoofdstuk.

Tijdens het uitvoeren van de work break down voor user story DD2-4606 in hoofdstuk **(User story DD2-4606)** heb ik gesprek gevoerd met de stakeholder om de user story verder toe te lichten. De stakeholder heeft aangegeven graag een zelf ingevoerd waarde in te vullen bij een zoekactie om te kunnen specificeren hoeveel resultaten er getoond moeten worden. Deze functionaliteit is gekoppeld aan deze user story (DD2-4607).

Voor deze user story heb ik eerst onderzocht wat er precies wordt bedoeld met een resultset. Volgens de documentatie van IBM is een resultset een set van zoek resultaten, folder contents of andere items die bij een zoek actie worden weergegeven **(IBM, 2020)**. Een resultset is een JavaScript model klas uit de Content Navigator API interface die gebruikt kan worden in een maatwerk plugin.

Ook is dit geen standaard mogelijkheid in Content Navigator, dus dient er voor het realiseren van deze functionaliteit een plugin voor ontwikkeld te worden.

Werkzaamheden

Onderstaande werkzaamheden dienen uitgevoerd te worden om de user story DD2-4607 te maken c.q. voltooien:

- Ontwerpen van de plugin.
- Ontwikkelen van maatwerk in de vorm van een plugin.
- Functioneel testen van de functionaliteit / plugin.
- Unittests schrijven.
- Plugin beschikbaar stellen in Content Navigator (build & deploy).

Randvoorwaarden

- Content Navigator omgeving beschikbaar.
- Kennis van Content Navigator omtrent ontwikkeling.

5.6 Retrospective

Wat heeft deze sprint opgeleverd?

- Geprioriteerde product backlog.
- Product backlog refinement document.
    - Work break downs.
- Nieuwe inzichten omtrent ontwikkeling binnen Content Navigator.

Wat ging er goed?

- De kennis die ik heb opgedaan tijdens de oriëntatie in sprint 0 heeft mij geholpen tijdens het uitvoeren van de work break downs.
- Communicatie met de stakeholder.
- Begeleider is tevreden met de uitvoering van de work break downs.

Wat ging minder goed?

- Ik vond het lastig om de werkzaamheden te bepalen, omdat de Content Navigator omgeving nog vrij nieuw is voor mij. De uitvoering van de work break downs duurde daarom langer dan gepland. Mijn begeleider had hier al rekening mee gehouden en heeft aangegeven dat dit bij het leerproces hoort.

Wat zijn de leerpunten?

6 Sprint 2 – Zoek plugin bouwen

Tijdens deze sprint heb ik de Content Navigator omgeving geconfigureerd op de WebSphere Applicatie Server van Digidoc. Helaas duurde het langer dan gepland om de Content Navigator omgeving beschikbaar te stellen door de vertraging bij derde partijen. Nadat de Content Navigator omgeving gereed was, kon ik aan de slag met het ontwikkelen van een plugin. Ik heb een klassendiagram gemaakt van de plugin die laat zien hoe de verschillende Java klassen van de plugin tot stand zijn gekomen en wat de relaties zijn tussen deze Java klassen. Echter bleek tijdens het maken van de klassendiagram en de kennis die ik heb opgedaan tijdens deze sprint, dat het handiger is om de zoek functionaliteiten van user story DD2-4606 en DD2-4607 die op de product backlog staan, bij elkaar te betrekken en te realiseren in één plugin (zoek plugin). Tijdens de oriëntatie in sprint 0 dacht ik dat er voor elke functionaliteit een nieuwe plugin ontwikkeld moest worden. Dat blijkt niet zo te zijn. Hierdoor heb ik tijdens de sprint uitvoering moeten bijsturen en aan het einde van de sprint de taken een andere benaming gegeven. Deze sprint is hierdoor ook anders verlopen dan we gepland hadden. Daarnaast ben ik deze sprint ook bezig geweest met het uitproberen van de Dojo Toolkit om hier alvast kennis over op te doen.

6.1 Sprint planning

De items in onderstaande tabel **(Tabel 12: Sprint backlog van sprint 2)** zijn door de product owner op de sprint backlog geplaatst en worden deze sprint opgepakt. Het backlog traceability Excel document **(Bijlage H: Backlog traceability)** heb ik bijgewerkt.

Jira ID	*Item (Taak)*	*Prioriteit*	*Points*
DD2-4624	Content Navigator configureren op de O8 ontwikkelomgeving	Hoogst	1
DD2-4679	Plugin ontwikkelen (user story DD2-4606)	Medium	5
DD2-4680	Unittest schrijven voor de plugin (user story DD2-4606)		3
DD2-4698	Plugin builden & deployen in Content Navigator	Medium	3
DD2-4681	Functioneel testen van de user story DD2-4606	Medium	1
DD2-4615	Work break down voor user story DD2-4614	Medium	1

Tabel 12: Sprint backlog van sprint 2

6.2 Configureren van de Content Navigator omgeving

Tijdens deze sprint heb ik de Content Navigator omgeving geconfigureerd op de Websphere Application Server van Digidoc. Binnen Content Navigator heb ik een

nieuwe desktop aangemaakt waarin de (niet) out-of-the-box functionaliteiten worden gerealiseerd. De desktop vormt de beheertooling in Content Navigator waarmee de beheerders van Digidoc beheerwerkzaamheden kunnen uitvoeren met de functionaliteiten die tijdens dit project worden gerealiseerd door mij.

WebSphere Application Server (WAS): Maakt het gemakkelijker om omvangrijke, complexe dynamische webapplicaties te deployen, te integreren en vervolgens te beheren

De volgende stappen hebben ik doorlopen tijdens de configuratie van de Content Navigator omgeving:

- Connectiviteit getest naar de Content Navigator omgeving vanuit mijn ontwikkelomgeving.
- Mijn begeleider heeft een admin account voor mij aangemaakt zodat ik kan inloggen in de Content Navigator omgeving.
- Repository (FileNet Content Manager) gekoppeld in Content Navigator.
- Nieuwe desktop aangemaakt in Content Navigator waarin de functionaliteiten gerealiseerd worden.
- Getest of een plugin kan worden ingeladen en deze wordt getoond in Content Navigator. Hiervoor heb ik een bestaande plugin gebruikt van een ander team.

6.3 Opstellen van een klassendiagram

Tijdens het maken van de klassendiagram heb ik nieuwe inzichten gekregen over hoe de niet out-of-the-box functionaliteiten van user story DD2-4606 en DD24607 gerealiseerd kunnen worden in de beheertooling van Content Navigator. Tijdens de uitvoering van de work break downs heb ik aangegeven dat voor beide user stories maatwerk ontwikkeld moet worden in de vorm van een plugin. Door de nieuwe inzichten die ik heb gecreëerd tijdens deze sprint kunnen de twee user stories samen gerealiseerd worden in één plugin. De werkzaamheden heb ik dan ook niet helemaal goed ingeschat tijdens de work break downs. De plugin die ik hiervoor ga maken noem ik de zoek plugin, omdat er twee gewenste zoek functionaliteiten in de plugin worden gerealiseerd. De nieuwe inzichten heb ik gekregen door dieper in de documentatie te duiken van IBM en daarnaast heb ik een meeting gehad met mijn collega Ivo Jonker. Ivo Jonker is een ervaren FileNet ontwikkelaar op het gebied van Content Navigator. Hij heeft mij uitleg gegeven over hoe er maatwerk ontwikkeld kan worden in Content Navigator door een korte demo te geven in zijn Eclipse omgeving. Dit heeft mij heel erg geholpen.

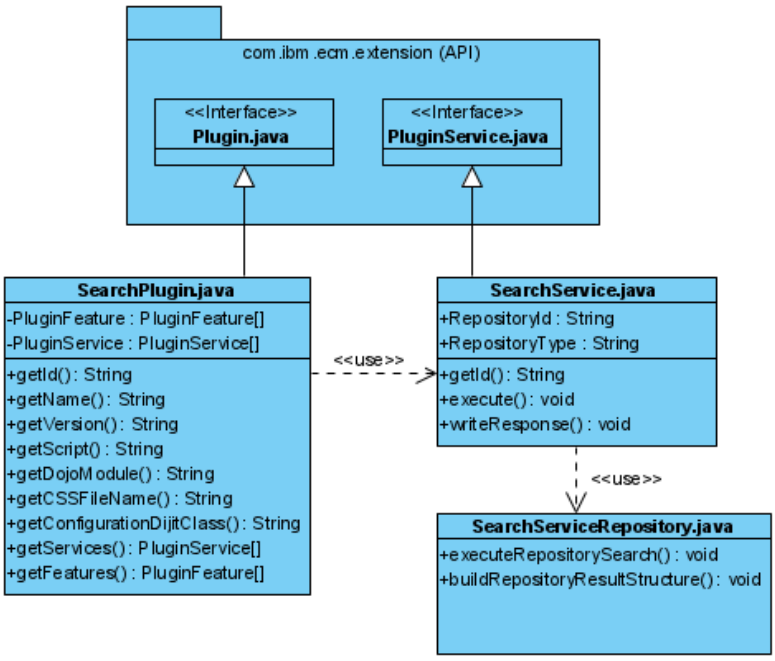
Voor de plugin heb ik een ontwerp gemaakt in de vorm van een klassendiagram. Dit is een eerste opzet en kan gedurende het realisatie proces nog wijzigen door nieuwe inzichten die worden gecreëerd. De klassendiagram is tot stand gekomen door de documentatie van IBM te gebruiken en bestaande Content Navigator

plugins op te zoeken om de structuur daarvan te bestuderen. IBM heeft een aantal handige voorbeelden van Content Navigator plugins op GitHub.

Wat ik als eerst heb ik gedaan is onderzoeken welke API interfaces van de Content Navigator API de zoek plugin nodig zal hebben. Elke maatwerk plugin dient de standaard Java interface van de plugin (plugin.java) te implementeren en alle variabelen en methodes van deze plugin interface te implementeren. In de IBM documentatie staat het volgende aangegeven over de plugin interface.

Provides the main class of an IBM Content Navigator plug-in. The abstract methods provide the name and version of the plug-in, and identify the components such as actions, services, and scripts that are included in the plug-in.

Ik heb een aparte package in het ontwerp aangemaakt om duidelijk aan te geven welke interfaces van de Content Navigator API worden geïmplementeerd. Vervolgens moest ik bepalen welke API interface(s) de plugin nog meer nodig heeft van de Content Navigator API. Dit bleek lastiger dan ik van ter voren had gedacht. Er zijn namelijk veel verschillende Java interfaces die geïmplementeerd kunnen worden in de plugin. Hiervoor heb ik de architectuur plaat gebruikt van Content Navigator die ik heb beschreven tijdens mijn oriëntatie in sprint 0. Om data te kunnen ophalen uit de FileNet Content Manager repository dien ik de service interface van de Content Navigator API te implementeren. De service interface verzorgt de communicatie tussen de repository en Content Navigator. Hiervoor heb ik een SearchService.java klasse aangemaakt die de PluginService.java implementeert van de Content Navigator API interface.



Figuur 3: Klassendiagram van de zoek plugin

6.4 Opzetten project structuur in Eclipse a.d.v. de klassendiagram

Aan de hand van de opgestelde klassendiagram heb ik de onderstaande Java packages, folders en bestanden aangemaakt in de Eclipse IDE. Een Content Navigator plugin heeft volgens de documentatie van IBM een standaard structuur die gehanteerd moet worden. Daarbij heb ik aangegeven waarvoor de Java package en folder voor dient.

- **com.ibm.ecm.searchplugin** – Bevat alle Java klassen van de plugin.
  - SearchService.java extends PluginService.java
  - SearchPlugin.java extends Plugin.java
  - SearchServiceRepository.java
- **com.ibm.ecm.searchplugin.WebContent** – Bevat de JavaScript en CSS bestanden van de plugin.
  - SearchPlugin.css
  - SearchPlugin.js
- **com.ibm.ecm.searchplugin.WebContent.images** – Bevat de afbeeldingen die worden gebruikt in de plugin.
- **com.ibm.ecm.searchplugin.WebContent.searchPluginDojo** – Bevat de Dojo Toolkit JavaScript klassen.
  - ConfigurationPane.js
- **com.ibm.ecm.searchplugin.WebContent.searchPluginDojo.templates** – Bevat de Dojo Toolkit template HTML bestanden.

Vervolgens heb ik de SearchPlugin.java klasse aangemaakt. Dit is de standaard interface van de zoek plugin waarin o.a. de componenten (services en features) worden geregistreerd middels de verschillende methodes die standaard geïmplementeerd worden door het extenden van de Plugin.java interface. Daarbij worden er ook methodes geïmplementeerd voor o.a. het aangeven van de plugin naam en versie.

6.5 Uitdagingen

Tijdens deze sprint ben ik een aantal uitdagingen tegengekomen. In dit hoofdstuk beschrijf ik deze uitdagingen.

6.5.1 Bepalen van de Content Navigator API interfaces

Tijdens het ontwerpen merkte ik dat ik in het begin vast liep en niet wist welke ontwerpkeuzes ik moest maken. De Content Navigator API biedt namelijk talloze mogelijkheden van Java interfaces en ik wist op dat moment niet welke interfaces ik moest gaan inzetten. Het was voor mij een uitdaging om van ter voren te bepalen welke Java interfaces ik nodig zal hebben voor de plugin, omdat ik nog nooit een plugin heb ontwikkeld. Mijn begeleider merkte dat ik vast liep tijdens dit proces en schakelde voor mij hulp in van een FileNet ontwikkelaar van

een ander DevOps team. Hij heeft mij een aantal voorbeelden laten zien en uitgelegd hoe de structuur van de plugin in elkaar zit. De uitleg en demo van de FileNet ontwikkelaar heeft mij een duw in de goede richting gegeven, waardoor ik voor mijn gevoel de juiste ontwerpkeuzes kon gaan maken. Ik merkte dat ik er complexer over dacht dan nodig was. Een plugin heeft zijn complexe logica maar de basis structuur is voor elke plugin hetzelfde. De FileNet ontwikkelaar adviseerde mij ook om de GitHub repository van IBM eens te bekijken. Daar staan namelijk een aantal voorbeelden van plugins voor Content Navigator.

6.5.2 Werken met de Dojo Toolkit

Het werken met de Dojo JavaScript library was voor mij een nieuwe maar ook een leuke uitdaging. Ik heb Dojo nooit gebruikt voor het ontwikkelen van een applicatie. Het was daarom voor mij – zeker in het begin – veel uitzoeken hoe de Dojo Toolkit precies in elkaar zit en hoe de verschillende klassen ingezet kunnen worden. Vooral hoe de service laag communiceert met de Dojo laag. Het wilde in het begin maar niet lukken om widgets te tonen in Content Navigator en dummy data van een service in te laden en te tonen in de widget. Het heeft zeker een aantal dagen geduurd totdat ik eindelijk een widget kon tonen met daarin dummy data die vanuit een service werd opgehaald. Ik begon de structuur en de samenhang tussen de Dojo klassen en service laag steeds beter te begrijpen.

6.6 Retrospective

Wat heeft deze sprint opgeleverd?

- Geconfigureerde Content Navigator omgeving met daarin een beschikbare desktop om de gewenste functionaliteiten in te implementeren.
- Een eerste opzet van de klassendiagram van de zoek plugin. De relaties tussen de verschillende Java klassen zijn vastgelegd.
- Nieuwe inzichten over de mogelijkheden voor het bouwen van de niet out-of-the-box functionaliteiten.
- Nieuwe kennis over de Dojo Toolkit toepassingen.
- Nieuwe kennis over de Content Navigator API mogelijkheden door het maken van een klassendiagram voor de zoek plugin.
- Project structuur a.d.v. gemaakte klassendiagram aangemaakt in de Eclipse IDE op mijn ontwikkelomgeving.

Wat ging er goed?

- Door de hulp van de FileNet ontwikkelaar en de oriëntatie die ik heb uitgevoerd in sprint 0 kon ik voor mijn gevoel de juiste keuzes maken voor het ontwerp van de klassendiagram van de zoek plugin. Keuzes om te bepalen welke interfaces van de Content Navigator API ingezet moesten en relatie tussen de onderlinge Java klassen van de zoek plugin.
- Van ter voren had ik een heel ander beeld van hoe de niet out-of-the-box functionaliteiten ingezet kunnen worden. Door de nieuwe inzichten die ik

deze sprint heb gecreëerd is het mij veel helderder geworden en blijkt het heel anders in elkaar te zitten dan ik van ter voren had gedacht. Het positieve hieraan is dat ik een beter beeld heb gekregen van hoe het uiteindelijk eind product gebouwd kan worden.

Wat ging er minder goed?

- De Content Navigator omgeving was niet beschikbaar bij de start van de sprint. Dit resulteerde in vertraging waardoor ik niet gelijk aan de slag kon met de beheer omgeving binnen Content Navigator.
- De story points waren van ter voren verkeerd ingeschat.
- Het werken met de Dojo Toolkit library was voor mij volledig nieuw. Hierdoor ging er meer tijd naar het leren van de Dojo Toolkit library dan ik van ter voren had gedacht. Deze sprint ben ik bezig geweest met het proberen te bouwen van een dummy plugin om de Dojo Toolkit te leren begrijpen.
- Het maken van een klassendiagram was lastiger dan van ter voren gedacht. De story points waren dan ook niet helemaal goed ingeschat.
- Deel van de sprint backlog taken zijn niet voltooid door de nieuwe inzichten die ik gecreëerd heb.

Wat zijn de leerpunten?

- Als de randvoorwaarden (bijv. Content Navigator omgeving) niet beschikbaar is, dan wordt het item niet in de sprint opgenomen. Dat heb ik deze sprint toch gedaan, omdat mij was verteld dat de Content Navigator omgeving snel beschikbaar was.
- Focus leggen op één taak i.p.v. verschillende taken combineren. Hierdoor lopen werkzaamheden door elkaar heen. Dit komt mede doordat ik te snel de materie rondom Content Navigator wil begrijpen.
- Door nieuwe inzichten de Product Backlog herzien en aanpassen. De gewenste zoek functionaliteiten van user story DD2-4606 en DD2-4607 worden gerealiseerd in één plugin.
- Meer achter mijn ontwerp beslissingen staan. Ik merkte deze sprint dat ik daar soms onzeker over was door de verschillende mogelijkheden die de Content Navigator API biedt.
- Story points beter inschatten door de kennis die ik deze sprint heb opgedaan.

7 Sprint 3 – Zoek plugin bouwen

Tijdens deze sprint heb ik veel werkzaamheden kunnen uitvoeren. In samenwerking met de stakeholder heb ik een UI-ontwerp gemaakt van de zoek plugin lay-out. Mijn doel van het UI-ontwerp was om met de stakeholder de user interface te schetsen en te kijken waar de stakeholder bepaalde elementen wilt hebben. Het maken van een UI-ontwerp is een belangrijk proces die het Digidoc Online team ook toepast. Naast het UI-ontwerp ben ik ook bezig geweest met het maken van een sequentiediagram om de interactie tussen de verschillende lagen van de zoek plugin vast te leggen. Een plugin gebruikt namelijk het model-view-controller design patteren en is opgedeeld in drie verschillende lagen met verschillende verantwoordelijkheden. In het sequentiediagram heb ik deze drie lagen verwerkt en de interactie tussen deze lagen beschreven. Tijdens de vorige sprint ben ik tot nieuwe inzichten gekomen over de wijze van het realiseren van de niet out-of-the-box functionaliteiten in de zoek plugin. Hierdoor heb ik de klassendiagram deze sprint moeten aanpassen. Daarna ben ik gestart met het ontwikkelen van de zoek plugin. De werkzaamheden die ik hiervoor heb uitgevoerd zijn beschreven in dit hoofdstuk.

7.1 Sprint planning

De items in onderstaande tabel **(Tabel 13: Sprint backlog van sprint 3)** zijn door de product owner op de sprint backlog geplaatst en worden deze sprint opgepakt. Het backlog traceability Excel document **(Bijlage H: Backlog traceability)** heb ik ook bijgewerkt.

Sommige taken heb ik een andere benaming gegeven, omdat de zoek functionaliteiten van user story DD2-4606 en DD2-4607 worden gerealiseerd in één plugin. Dit is één van de leerpunten uit de vorige sprint. Daarnaast heb ik de hoeveelheid werk van de taken deze sprint beter kunnen inschatten door de ervaring die ik heb opgedaan in de vorige sprint.

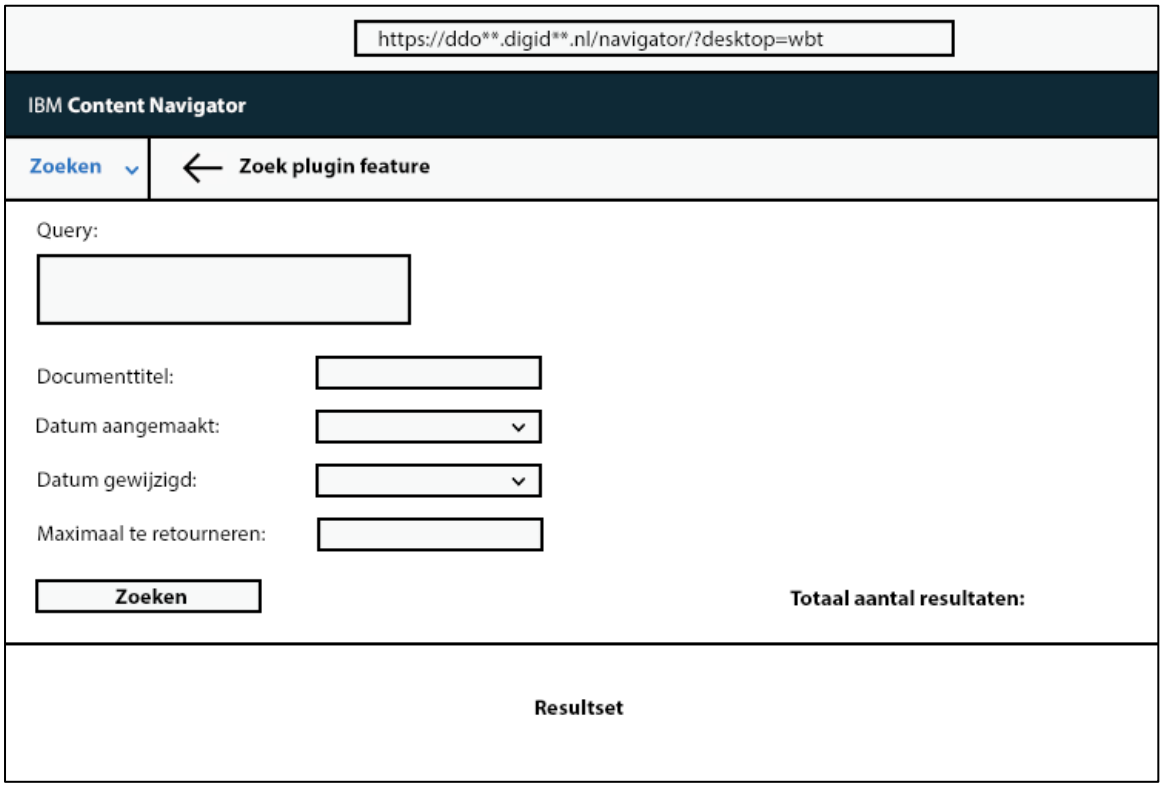
Jira ID	*Item (Taak)*	*Prioriteit*	*Points*
DD2-4771	UI-ontwerp maken voor de zoek plugin	Highest	2
DD2-4772	Sequentiediagram maken voor de zoek plugin	High	3
DD2-4773	Klassendiagram bijwerken voor de zoek plugin	High	2
DD2-4679	Zoek functionaliteit van user story DD2-4606 realiseren in de zoek plugin	Medium	5
DD2-4682	User story DD2-4606 realiseren in de zoek plugin	Medium	8
DD2-4680	Unittest schrijven voor de zoek plugin	Medium	5
DD2-4698	Plugin builden & deployen in Content Navigator	Medium	1

DD2-4681	Functioneel testen van de user story DD2-4606	Medium	1

Tabel 13: Sprint backlog van sprint 3

7.2 Maken van een UI-ontwerp

Van ter voren heb ik in samenwerking met de stakeholder en de product owner een UI-ontwerp gemaakt van de user interface van de zoek plugin. Het maken van een UI-ontwerp is een proces dat het Digidoc Online team ook toepast om te schetsen hoe de stakeholder de user interface wenst te hebben. Tijdens het gesprek heb ik mijn scherm gedeeld waarin ik een leeg scherm in Content Navigator had openstaan. Vervolgens heb ik de stakeholder gevraagd welke zoek criteria in de zoek plugin moet komen te staan. Daarbij heeft de product owner aangegeven voor de scope van dit project maximaal vier zoek criteria mogen zijn. De stakeholder gaf aan dat het zoeken op een documenttitel, datum aangemaakt / gewijzigd de meest gebruikte zoek criteria zijn. Daarbij heb ik aangegeven dat het wellicht een vrije query zoek veld een goede toevoeging kan zijn. De stakeholder en product owner vonden dit een goed idee. Uiteindelijk zijn we tot het onderstaande UI-ontwerp gekomen. Beide user stories (DD2-4606 en DD2-4607) heb ik verwerkt in het UI-ontwerp.



Figuur 4: UI-ontwerp van de zoek plugin

7.3 Opstellen van een sequentiediagram

Tijdens de oriëntatie die ik heb uitgevoerd in sprint 0 **(Oriënteren op de mogelijkheden van Content Navigator)** ben ik een architectuur afbeelding van Content Navigator tegengekomen. In deze architectuur afbeelding zijn verschillende lagen afgebeeld die elk zijn eigen verantwoordelijkheden hebben binnen Content Navigator. Deze lagen zijn gebaseerd op het model-view-controller design pattern. De architectuur afbeelding van Content Navigator **(Figuur 2: Architectuur van Content Navigator)** heb ik gebruikt om de interactie tussen de verschillende lagen in de zoek plugin vast te leggen in een sequentiediagram. De presentatie en data access en laag heb ik beide verwerkt in één levenslijn. Beide lagen vormen samen de web laag van de zoek plugin. Hierdoor is de scheiding tussen de client en backend goed zichtbaar.

7.3.1 Nieuwe inzichten

Tijdens de oriëntatie in sprint 0 was ik van de veronderstelling dat er voor elke laag in de architectuur van Content Navigator een aparte plugin ontwikkeld moest worden. Dit dacht ik omdat er op elke laag een 'custom plugin' is afgebeeld. Echter nu ik verder in het proces zit en nieuwe kennis heb opgedaan rondom Content Navigator, ben ik er achter gekomen dat het andersom werkt. Met een Content Navigator plugin is het mogelijk om voor de verschillende lagen maatwerk te bouwen en implementeren.

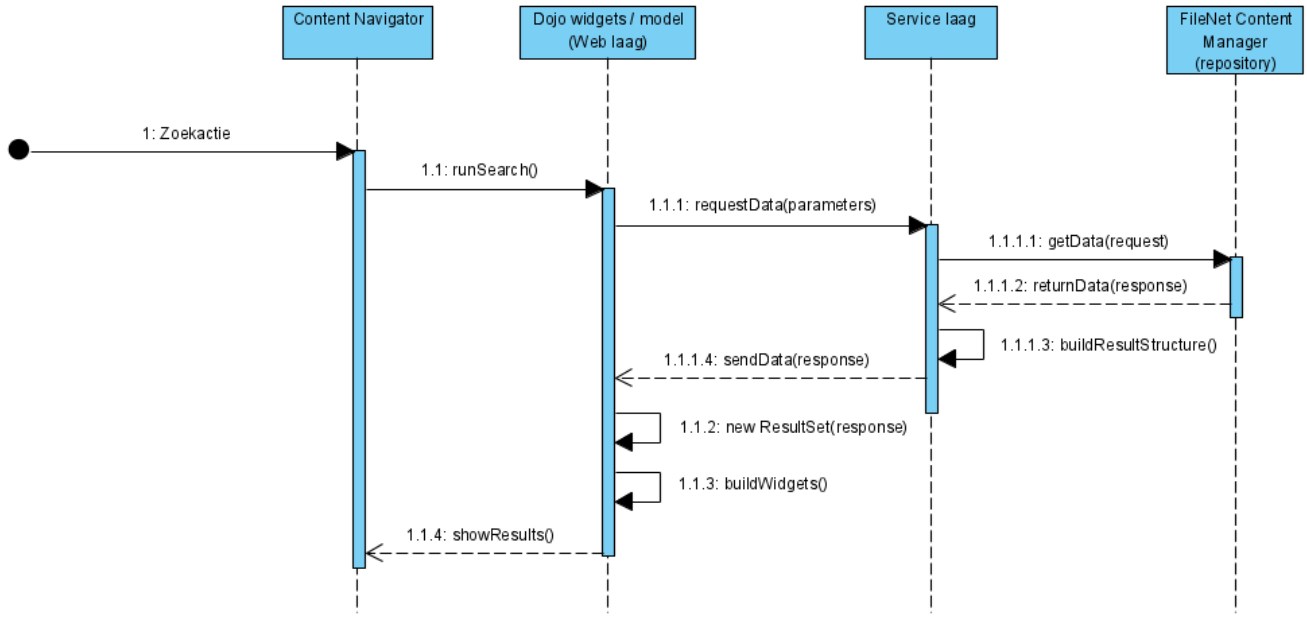
7.3.2 Totstandkoming

De sequentiediagram is tot stand gekomen door de kennis die ik heb opgedaan tijdens de oriëntatie en de nieuwe kennis die ik heb opgedaan in de vorige sprint. Ik ben steeds meer gaan begrijpen van de structuur van een Content Navigator plugin en welke verantwoordelijkheden elke laag heeft. Ik vind het mooi om te zien dat ik van helemaal geen kennis van Content Navigator plugins naar dit punt ben gekomen. Hierdoor heb ik gemotiveerde ontwerpkeuzes kunnen maken en gaf het mij extra motivatie om meer kennis op te bouwen rondom Content Navigator. Daarnaast heb ik ook de documentatie van IBM Content Navigator **(IBM, 2020)** gebruikt om de juiste ontwerpkeuzes te kunnen maken.

7.3.3 Interactie proces

Het startblok begint bij een zoek actie die de gebruiker wordt uitgevoerd met de zoek plugin in Content Navigator. De gebruiker interacteert in Content Navigator met Dojo widgets die op de presentatie laag (view) zijn geïmplementeerd. Content Navigator fungeert als controller en reageert op handelingen die door gebruikers worden uitgevoerd met Dojo widgets. De runSearch() methode van de Dojo Toolkit wordt aangeroepen zodra de gebruiker op de zoek button klikt. Deze handeling activeert namelijk een onClick event. Vervolgens verzameld en verwerkt de data access laag (model) de waardes die zijn ingevuld in de Dojo widgets naar de service laag. Dit doet de data access laag met de Request model

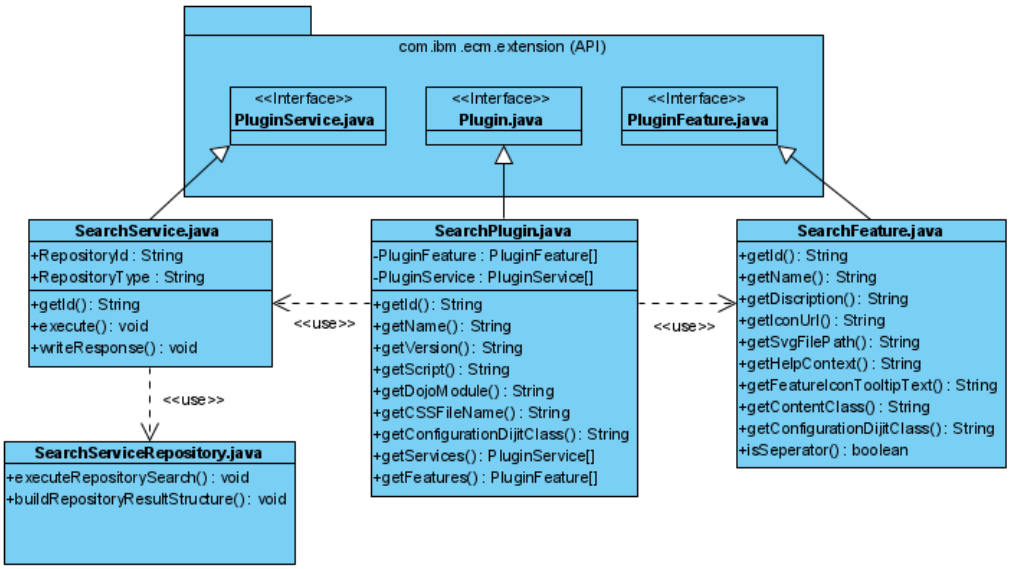
klasse van de Content Navigator API. De service laag communiceert vervolgens met de FileNet Content Manager repository om te benodigde data op te halen, te verwerken en te retourneren (JSON) naar de data access laag. De data access laag werkt vervolgens de model klasse bij. De data access laag werkt vervolgens de Dojo widgets op de presentatie laag bij met de gewenste resultaten.



Figuur 5: Sequentiediagram van de zoek plugin

7.4 Bijwerken van de klassendiagram

Door nieuwe inzichten die zijn gecreëerd heb ik de klassendiagram moeten wijzigen. Om te kunnen interacteren met de zoek plugin heb ik een feature interface toegevoegd. De feature interface is een nieuwe menu item die wordt toegevoegd in het bestaande menu van Content Navigator waarmee de gebruiker met geïmplementeerde Dojo widgets de mogelijkheid heeft om te kunnen interacteren met de zoek plugin. In de bijgewerkte klassendiagram **(Figuur 6: Bijgewerkte klassendiagram van de zoek plugin)** heb ik de feature interface toegevoegd. De feature Java klasse (SearchFeature.java) die ik heb toegevoegd, implementeert de plugin feature interface (PluginFeature.java) van de Content Navigator API en alle bijbehorende methodes.



Figuur 6: Bijgewerkte klassendiagram van de zoek plugin

7.5 Implementeren en registreren van de API interfaces

De service en feature interfaces die vanuit de Content Navigator API zijn geïmplementeerd, dienen te worden geregistreerd in de hoofdklasse (SearchPlugin.java) van de zoek plugin. De zoek plugin weet door deze registratie welke interfaces van de Content Navigator API zijn geïmplementeerd. De registratie van de service interface is zichtbaar in onderstaande code.

```
public PluginService[] getServices() {
    if (pluginServices.length == 0) {
        pluginServices = new PluginService[] {new com.ibm.ecm.searchplugin.SearchService()};
    }
    return pluginServices;
}
```

Door de nieuwe inzichten die ik heb gecreëerd tijdens de vorige sprint heb ik de klassendiagram van de zoek plugin moeten aanpassen. De aanpassing betrof het toevoegen van een feature interface die het mogelijk maakt om een nieuw menu item in Content Navigator toe te voegen met daarbij een lay-out die naar eigen wens ingericht kan worden met Dojo widgets. In de bijgewerkte klassendiagram **(Figuur 6: Bijgewerkte klassendiagram van de zoek plugin)** heb ik een nieuwe Java klasse (SearchFeature.java) toegevoegd die de feature interface van de Content Navigator API implementeert. Net zoals het registreren van een service interface dient in de SearchPlugin.java klasse middels de getFeatures() methode de SearchFeature.java klasse geregistreerd te worden. De methodes getFeatures() en getServices() heb ik beide geïmplementeerd vanuit beide interfaces. De registratie van de feature interface is zichtbaar in onderstaande code.

```
public PluginFeature[] getFeatures() {
    if (pluginFeatures.length == 0) {
        pluginFeatures = new PluginFeature[] {new com.ibm.ecm.searchplugin.SearchFeature()};
    }
    return pluginFeatures;
}
```

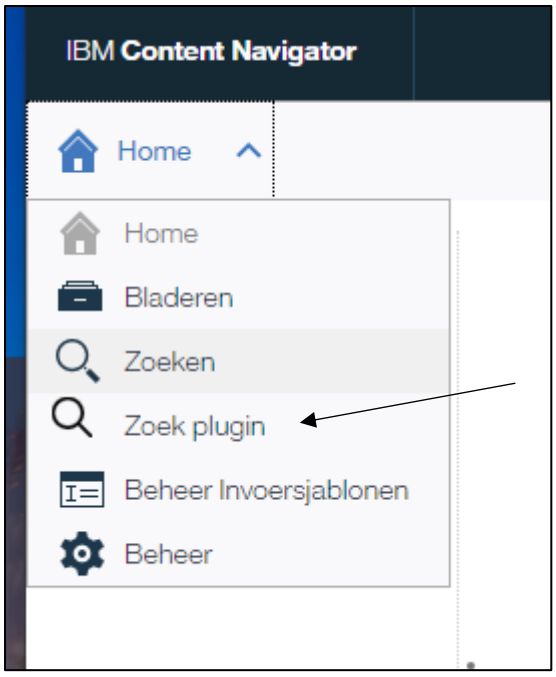
7.6 Builden & deployen van een plugin in Content Navigator

De geschreven code voor de zoek plugin was pas zichtbaar in beheer desktop binnen Content Navigator, nadat er een build was gemaakt van de plugin in de vorm van een .JAR file. Ik wist op dit moment nog niet hoe er een build gemaakt kon worden van de source code. Door het volgen van de installatie stappen in de redbooks van IBM **(IBM, 2014)** kwam ik erachter dat Eclipse standaard beschikt over een op Java gebaseerde build tool genaamd Apache Ant. Deze tool heb ik gebruikt om een build te maken van de plugin om deze vervolgens te kunnen deployen in Content Navigator en te koppelen aan de beheer desktop.

Na mijn eerste build liep ik tegen een voor mij aantal onbekende foutmeldingen aan. Door het lezen van de documentatie van IBM Content Navigator kwam ik erachter dat er een aantal benodigde Java libraries ontbraken in de build.xml. De navigatorAPI.jar, j2ee.jar, Jave-5.5.0.0.jar en de commons-codec-1.15.jar files ontbraken in de build.xml configuratie en moesten in de lib folder geplaatst worden. Deze benodigde JAR bestanden stonden op de Websphere Application Server van Content Navigator. De navigatorAPI.jar is één van de JAR files die ontbrak die ervoor zorgt dat de Content Navigator API aangeroepen kan worden door de plugin.

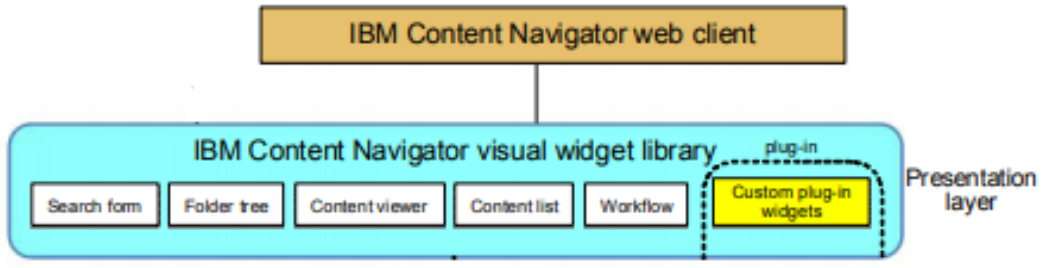
```
<path id="classpath">
    <pathelement location="./lib/navigatorAPI.jar" />
    <pathelement location="./lib/j2ee.jar" />
    <pathelement location="./lib/Jace-5.5.0.0.jar" />
    <pathelement location="./lib/commons-codec-1.15.jar" />
```

Nadat de benodigde JAR files waren geconfigureerd in de build.xml kon ik vervolgens een build maken van de zoek plugin en deployen in Content Navigator. Na het deployen kon ik in het menu van Content Navigator zien dat de zoek plugin succesvol was gedeployed. In onderstaande afbeelding is de feature interface van de zoek plugin zichtbaar in het menu van Content Navigator. De gebruiker klikt hierop om de gewenste zoek functionaliteiten te kunnen gebruiken in Content Navigator. De zoek plugin is gekoppeld aan de beheerdesktop die ik tijdens de configuratie van Content Navigator heb aangemaakt. Deze desktop vormt de beheertooling binnen Content Navigator.



7.7 Bouwen van de feature lay-out met de Dojo Toolkit

De presentatie laag (view) van een maatwerk plugin voor Content Navigator bestaat uit diverse Dojo widgets waarmee de gebruiker kan interacteren via de web client (Content Navigator). Het belangrijkste component van de presentatie laag is de visual widget library van de Content Navigator API. Deze library biedt een set aan Dojo widgets die een ontwikkelaar kan gebruiken om een lay-out voor de plugin te bouwen. De visual widget library is gebouwd op de bestaande Dojo Toolkit **(IBM, 2014)**. In onderstaande afbeelding is de presentatie laag zichtbaar. Deze afbeelding is afkomstig van de architectuur afbeelding van Content Navigator **(Figuur 2: Architectuur van Content Navigator).**



Figuur 7: Presentatie laag van de zoek plugin

Tijdens het bouwen van de feature lay-out voor de zoek plugin heb ik het UI-ontwerp **(Figuur 4: UI-ontwerp van de zoek plugin)** gebruikt om te bepalen welke Dojo widgets ik moest gaan implementeren.

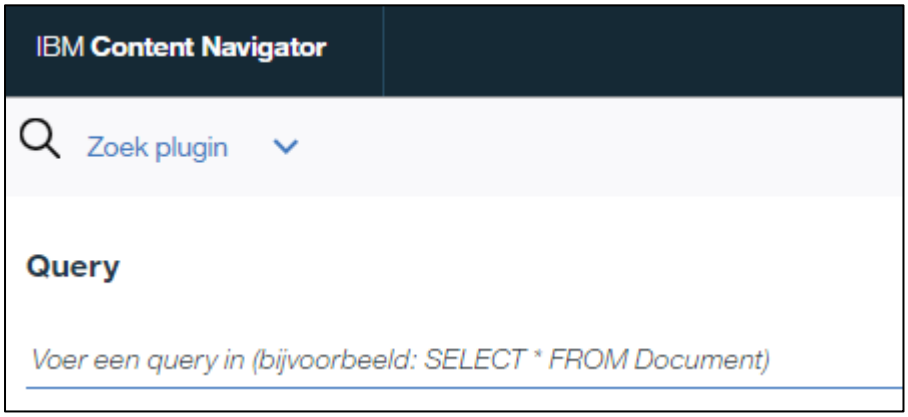
Het was voor mij de eerste keer dat ik de Dojo Toolkit ging gebruiken om een user interface te bouwen. Ik ben daarom aan het begin veel bezig geweest met het lezen van de documentatie van de Dojo Toolkit en het uitproberen van

verschillende Dojo widgets om de uitkomst hiervan te testen. Ook de structuur van de Dojo Toolkit wilde in het begin niet landen, waardoor ik langzaam van start ging. Op dit moment had ik niet de mogelijkheid om even snel wat te vragen aan een collega die hier meer vanaf weet. Dit kwam omdat meerdere collega's het erg druk hadden met een belangrijk project. Dit was een moment dat ik het even naar iemand toe lopen om een vraag te stellen erg handig had gevonden. Ik ben daarom zelfstandig dingen gaan uitzoeken en uitproberen met de Dojo Toolkit. Hierdoor heb ik in een korte tijd meer kunnen leren van de Dojo Toolkit en ook kunnen toepassen.

Van ter voren had ik besloten om het UI-ontwerp in een bepaalde volgorde na te bouwen met de Dojo Toolkit. Ik ben bovenaan begonnen en werkte langzaam naar beneden toe. Eén van de wensen van de stakeholder was het zoeken naar content middels een vrije zoek query. De gebruiker kan dan bijv. met de query SELECT * FROM Document zoeken naar alle documenten die aanwezig zijn in de FileNet Content Manager repository. Om dit te realiseren heb ik onderstaande Dojo widget geïmplementeerd. De ingevulde query wordt opgeslagen in de data-dojo-attach-point property van de Dojo widgets.

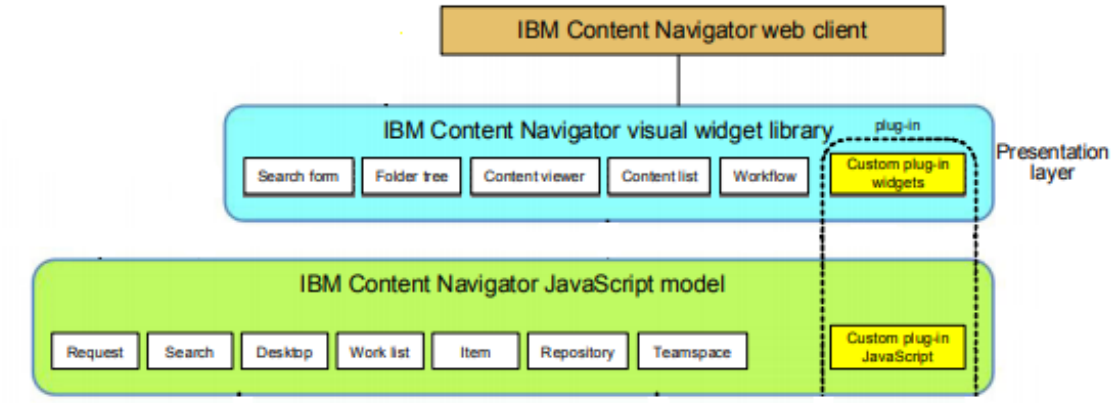
```
<h3>Query</h3>
<input id="queryString" data-dojo-attach-point="querySearch" data-dojo-type="dijit/form/TextBox"
placeholder="Voer een query in (bijvoorbeeld: SELECT * FROM Document)"></input>
```

Het wilde op dit moment niet lukken om de Dojo widget zichtbaar te krijgen in Content Navigator. Na een tijdje troubleshooten kwam ik erachter dat de Dojo widgets in Content Navigator binnen de geplaatst dienen te worden. De Content Navigator API beschikt namelijk maatwerk Dojo widgets die specifiek zijn gebouwd voor Content Navigator. In onderstaande afbeelding is de Dojo widget zichtbaar waarmee de gebruiker kan zoeken middels een vrije query zoek veld. Op dit moment kon ik hier nog niets mee doen. Het lukt me wel om de ingevoerde waarde op te halen en in de console te tonen.



7.8 Bouwen van de data access laag

Een groot deel van de presentatie laag heb ik gebouwd met de Dojo Toolkit. Dit heb ik gedaan door op feature lay-out van de zoek plugin verschillende Dojo Widgets te implementeren. Een volgende stap in het realisatie proces was het bouwen van de data access laag. Het belangrijkste component van de data access laag is de model library van de Content Navigator API. Hierin staan verschillende model klassen die geïmplementeerd kunnen worden om data bij te houden in een model klasse en te kunnen communiceren met de onderliggende service laag. In de onderstaande afbeelding is in de groene laag de data access laag zichtbaar. Deze afbeelding is afkomstig van de architectuur afbeelding van Content Navigator **(Figuur 2: Architectuur van Content Navigator)**.



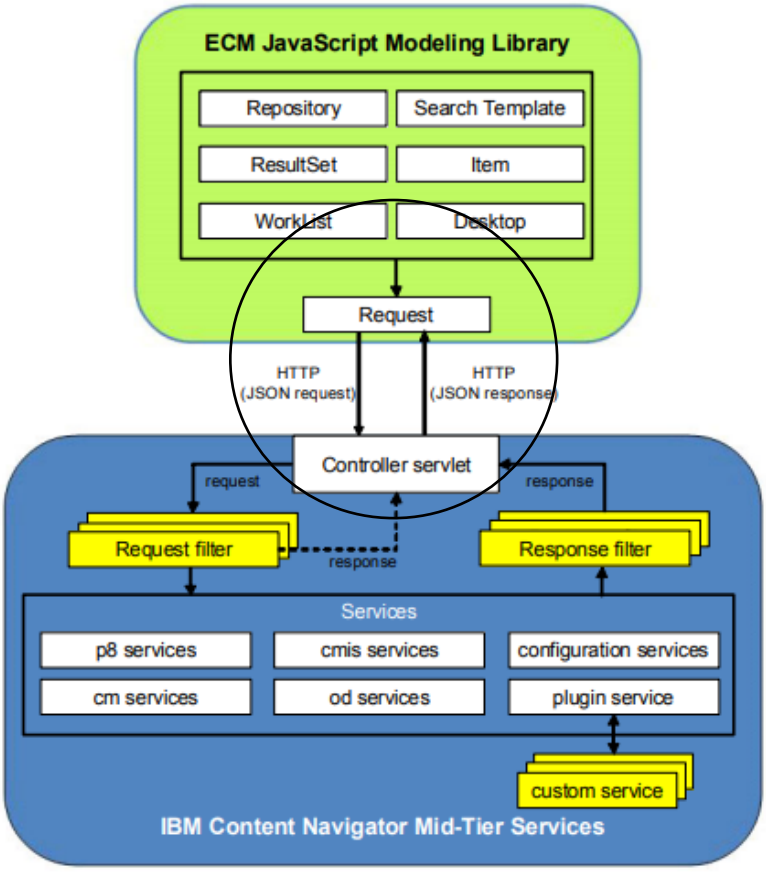
Figuur 8: Data access laag van de zoek plugin

In de sequentiediagram heb ik de presentatie en data access laag in één tijdlijn geplaatst. Volgens de IBM Content Navigator documentatie bevinden beide lagen zich namelijk aan de web kant.

Tijdens het bouwen van de data access laag heb ik gekeken welke model klassen geïmplementeerd moeten worden van de Content Navigator API. Op dit moment had ik nog totaal geen idee welke model klassen ik moest gaan implementeren. Op dit punt liep ik een beetje vast. Op advies van mijn begeleider heb ik de redbooks van IBM Content Navigator doorgelezen om hier meer informatie over te verzamelen. Mijn begeleider gaf namelijk aan dat de redbooks goed gedocumenteerd zijn en hier een aantal plugin voorbeelden in staan die wellicht aansluiten bij mijn opdracht. In de redbooks werd bij elk voorbeeld gebruik gemaakt van de ResultSet model om data op te slaan. Ik ben deze model gaan uitproberen en kreeg het gewenste resultaat. In de redbooks staat het volgende beschreven over de ResultSet model klasse:

Represents a set of search results, folder contents, or other items that are returned by a query to the content server. A ResultSet object contains data about the items found by the query and about how these items are to be displayed.

De data access laag is niet alleen verantwoordelijk voor het bijhouden van data, maar ook voor de communicatie met de service laag. Onderstaande afbeelding **(Figuur 9: Communicatie tussen de data access en service laag)** kwam ik tegen in de redbooks waarin staat aangegeven dat de Request model klasse wordt gebruikt om een HTTP request uit te voeren naar de service laag. Dit was precies waar ik op dat moment naar op zoek was. Onderstaande afbeelding toont de interactie de data access laag (groene gedeelte) en de service laag (blauwe gedeelte). De interactie tussen beide lagen is het uitvoeren van een request naar de service laag en het ontvangen van een response op de data access laag. De response wordt door de ResultSet model verwerkt.



Figuur 9: Communicatie tussen de data access en service laag

In onderstaande code is zichtbaar op welke wijze ik de Request en ResultSet model heb geïmplementeerd. De Request model voert middels de invokePluginService() methode een request uit naar de service laag (SearchService.java) met daarbij de requestParams (ingevoerde waardes in de Dojo widgets). Met de requestCompleteCallback() methode wordt de geretourneerde response data vanuit de service laag in de ResultSet model verwerkt.

```
Request.invokePluginService("SearchPlugin", "SearchService",
    {
        requestParams: requestParams,
        requestCompleteCallback: lang.hitch(this, function(response) {
            var resultSet = new ResultSet(response);
            this.searchResults.setResultSet(resultSet);
        })
    }
);
```

Op de presentatie laag heb ik onderstaande code geïmplementeerd voor het tonen van de ResultSet data in de zoek plugin. Hiervoor heb ik de ContentList widget moeten implementeren. Anders kan de ResultSet niet getoond worden in een de Dojo Widget. De ResultSet wordt onderaan de lay-out van de zoek plugin getoond. Hoe de Content List widget deze data verwerkt, is zichtbaar in de afbeelding van de zoek plugin lay-out **(Bijlage M: Zoek plugin lay-out)**.

```
<div data-dojo-attach-point="resultsArea" data-dojo-type="dijit/layout/ContentPane"
    data-dojo-props="region:'center', splitter:true">
    <div data-dojo-attach-point="searchResults" data-dojo-type="ecm/widget/listView/ContentList"
        data-dojo-props="emptyMessage:'${messages.folder_is_empty}'"></div>
</div>
```

7.9 Schrijven van JUnit tests

Aan het einde van deze sprint ben ik bezig geweest met het schrijven van verschillende unittests voor de geschreven Java code in de zoek plugin. Ik heb twee unittests geschreven om de registratie van beide interfaces te testen en een unittest om de naam van de zoek plugin te controleren. Op dit moment was er nog geen testplan. Mijn begeleider adviseerde mij om alvast unittests te schrijven om dadelijk aan het einde de 50% code corverage te kunnen borgen. Volgens de planning ben ik sprint 5 bezig met het opstellen van een testplan waarin de testaanpak wordt beschreven.

7.9.1 Testen registratie feature & service interface

```
@Test
public void testIfServiceIsRegisterd() {

    SearchPlugin plugin = new SearchPlugin();
    PluginService[] pluginServices = plugin.getServices();

    assertTrue("Service moet een object zijn van de SearchService klasse",
            pluginServices[0] instanceof SearchService);

}
```

```
@Test
public void testIfFeatureIsRegisterd() {

    SearchPlugin plugin = new SearchPlugin();
    PluginFeature[] pluginFeatures = plugin.getFeatures();

    assertTrue("Feature moet een object zijn van de SearchFeature klasse",
            pluginFeatures[0] instanceof SearchFeature);

}
```

7.9.2 Testen feature benaming in Content Navigator

```
@Test
public void testFeatureName() {

    SearchFeature searchFeature = new SearchFeature();
    Locale locale = new Locale("nl");
    String featureName = searchFeature.getName(locale);
    assertEquals("De naam van de menu item moet 'Zoek plugin' zijn",
            "Zoek plugin", featureName);
}
```

7.10 Retrospective

Wat heeft deze sprint opgeleverd?

- Een UI-ontwerp van de lay-out van de zoek plugin.
- Sequentiediagram die de interactie tussen de verschillende lagen in de zoek plugin beschrijft.
- Bijgewerkte klassendiagram van de zoek plugin.
- Groot deel van de Dojo widgets geïmplementeerd op de presentatie laag van de zoek plugin.
- Data access laag gebouwd met twee model klassen van de Content Navigator API.
- Een deel van de service laag gebouwd. Nog niet werkend gekregen deze sprint.
- Nieuwe kennis over de mogelijkheden van de Content Navigator API.

- Nieuwe bron **(IBM, 2014)** die mij kan helpen bij de verdere ontwikkeling van de zoek plugin.

Wat ging er goed?

- Informatie van Content Navigator omtrent ontwikkeling kon ik makkelijker vinden door de ervaring die ik heb opgedaan in de vorige sprints.
- Communicatie met de stakeholder verliep goed tijdens het opstellen van het UI-ontwerp.
- Het zelfstandig leren en uitproberen van de Dojo Toolkit.

Wat ging er minder goed?

- Het bepalen van de juiste model klassen voor de data access laag was lastiger dan ik van ter voren had ingeschat. Ik ben hier langer mee bezig geweest waardoor ik de service laag deze sprint niet werkend heb gekregen.

Wat zijn de leerpunten?

- Meer achter mijn genomen ontwerp/ontwikkel beslissingen staan. Ik ben vooral zelfstandig aan het werk waardoor ik soms twijfel over bepaalde beslissingen die ik maak tijdens het bouwen / ontwerpen van de zoek plugin. Dit komt grotendeels doordat ik voor het eerst werk met de technieken rondom Content Navigator.

8 Sprint 4 – Zoek plugin bouwen

Tijdens de vorige sprint was ik al begonnen met het bouwen van de service laag om de communicatie tussen de data access laag en de service laag werkend te krijgen. Dit was niet gelukt tijdens de vorige sprint door de vertraging die ik opliep tijdens het bouwen van de data access laag. Deze sprint ben verder gegaan met het bouwen van de service laag. Nadat de service laag gebouwd was kwam ik erachter dat de Dojo widgets waarin de datum aangemaakt en gewijzigd niet de juiste data formats retourneren naar de service laag. Deze sprint ben ik bezig geweest met het vinden van een passende oplossing hiervoor. Deze sprint heb ik ook gewenste zoek functionaliteiten die zijn gekoppeld aan user story DD2-4606 en DD2-4607 werkend gekregen.

8.1 Sprint planning

De items in onderstaande tabel **(Tabel 14: Sprint backlog van sprint 4)** zijn door de product owner op de sprint backlog geplaatst en worden deze sprint opgepakt. Het backlog traceability Excel document **(Bijlage H: Backlog traceability)** heb ik ook bijgewerkt.

Jira ID	*Item (Taak)*	*Prioriteit*	*Points*
DD2-4843	Datum velden in de zoek plugin ontwikkelen	High	5
DD2-4679	User story DD2-4606 realiseren in de zoek plugin	Medium	5
DD2-4680	Unittests schrijven voor de zoek plugin	Medium	5
DD2-4681	Functioneel testen van de user story DD2-4606	Medium	1
DD2-4684	Functioneel testen van de user story DD2-4607	Medium	1

Tabel 14: Sprint backlog van sprint 4

8.2 Implementeren van de HttpServletRequest interface

In de afbeelding **(Figuur 9: Communicatie tussen de data access en service laag)** waarin de communicatie tussen data access en service laag is afgebeeld, is zichtbaar dat er communicatie plaats vind middels een request en response. Ik wist nog niet hoe dit op de service laag eruit komt te zien en op welke wijze dit geïmplementeerd moest worden. Na het lezen van de redbooks en het bekijken van plugin samples op de GitHub van IBM, kwam ik erachter dat de HttpServletRequest interface wordt gebruikt om de requests tussen beide lagen te verzorgen. Deze interface is in de afbeelding **(Figuur 9: Communicatie tussen de data access en service laag)** beschreven als request filter. Middels de getParamater() methode van de HttpServletRequest worden de paramaters die vanuit de data access laag worden meegegeven opgehaald. In

onderstaande code is zichtbaar dat er twee variablen worden opgehaald middels de getParamater() methode van de HttpServletRequest interface. Volgens de coding standaarden **(Bijlage I: Coding standaarden)** mogen er geen hardcoded variablen gebruikt worden. Daarom heb ik FINAL variablen aangemaakt voor het getParamater() methode.

```
String querySearch = request.getParameter(QUERY_SEARCH);
int totalSearch = Integer.parseInt(request.getParameter(TOTAL_SEARCH));
```

8.3 Ophalen data uit de FileNet Content Manager repository

Een volgende stap was het ophalen van de benodigde informatie uit de FileNet Content Manager repository. Hiervoor heb ik een aparte Java klasse voor aangemaakt genaamd SearchServiceRepository. Hier had ik al rekening mee gehouden tijdens het opstellen van de klassendiagram voor de zoek plugin **(Figuur 3: Klassendiagram van de zoek plugin)**. Het vermijden van een te lange Java klasse is de reden geweest om een aparte Java klasse hiervoor te maken. Dit is ook één van de coding standaarden van het Digidoc Online team waar ik van ter voren rekening mee heb gehouden. Dit voorkomt dat de Java klasse SearchService in de toekomst alsnog aangepast moet worden, waardoor deze Java klasse te groot wordt.

Vervolgens kwam ik bij het complexe gedeelte van de service interface. Namelijk het ophalen van de benodigde data uit de FileNet Content Manager Repository en vervolgens het vertalen van de geretourneerde data naar een JSON object. Op dit punt heb ik veel code geprobeerd te schrijven, maar wilde het maar niet werken. Ik heb hulp ingeschakeld van een FileNet ontwikkelaar die mij wellicht een duw in de goede richting kon geven. De FileNet ontwikkelaar heeft mij een aantal handvatten gegeven over de mogelijkheden om de service interface op te bouwen. Daarbij gaf de FileNet ontwikkelaar ook aan dat ik op de goede weg zit. De FileNet ontwikkelaar gaf mij als tip om de samples van Content Navigator plugins eens te bekijken op de GitHub repository van IBM. Uiteindelijk heeft het bouwen van dit gedeelte van de service interface meer tijd gekost dan ik van ter voren had gedacht. Op het moment dat ik resultaat te zien kreeg, gaf dat mij enorm veel motivatie om verder door te pakken. Het volgende heb ik geïmplementeerd in de SearchServiceRepository Java klasse.

8.3.1 SearchSQL

De manier om de benodigde data op te vragen bij een FileNet repository is door het uitvoeren van een SQL query **(IBM, 2020)**. De SearchSQL interface van de FileNet API wordt gebruikt om queries tegen de FileNet repositories uit te voeren. De SearchSQL interface kwam ik ook tegen in meerdere plugin samples op de GitHub repository van Content Navigator.

In de service interface (SearchServiceRepository.java) heb ik een SearchSQL object aangemaakt om vervolgens met de setQueryString() methode een query op te bouwen om vervolgens uit te voeren. Onderstaande code heb ik geïmplementeerd voor het aanmaken van een SearchSQL object en het uitvoeren van een query. De querySearch paramater is door de gebruiker in de vrije zoek query veld (Dojo widget) ingevoerd in Content Navigator.

```
SearchSQL searchSQL = new SearchSQL();

if (!querySearch.isEmpty()) {
    searchSQL.setQueryString(querySearch);
}
```

De resultaten van de uitgevoerde SQL query worden vervolgens opgeslagen in een ArrayList. De PageIterator dient te worden geïmplementeerd om de paging functionaliteit toe te passen **(IBM, 2014)**. Onderstaande code heb ik geïmplementeerd om de resultaten van de query met de paging functionaliteit op te slaan in een ArrayList.

```
List<Object> searchResults = new ArrayList<Object>();

PageIterator pageIterator = resultsObjectSet.pageIterator();

if (pageIterator.nextPage()) {
    for (Object obj : pageIterator.getCurrentPage()) {
        searchResults.add(obj);
    }
}
```

8.3.2 JSONResultSetResponse

Volgens de redbooks wordt een response naar de data access laag opgebouwd middels een JSONResultSetResponse object klasse. Het volgende heb ik hierover kunnen vinden in de documentatie van de Content Navigator API:

This class structures the JSON used to represent a result set response. These responses are returned for searches, folder contents, worklists, and other requests that return a list of content or work items.

Ik heb in de vorige sprint op de presentatie laag de ResultSet model verwijzing in de ContentList widgets geplaatst. Op dat moment had ik nog geen idee hoe de resultaten uiteindelijk getoond worden in de zoek plugin. Tijdens deze sprint ben ik erachter gekomen dat de response naar de data access laag een bepaalde structuur moet hebben, zodat de ResultSet model op de data access laag de data kan inlezen en tonen in Content Navigator. Door het bestuderen van de samples op de GitHub repository van Content Navigator en het lezen van de redbooks heb ik het volgende geïmplementeerd.

Middels een for loop iterateer ik door de opgehaalde query resultaten heen. Vervolgens heb ik door onderstaande code te implementeren de rijen opgebouwd in een JSONResultSetRow object klasse.

```
row.addAttribute("Name", doc.get_Name().toString(), JSONResultSetRow.TYPE_STRING, null,
        doc.get_Name().toString());
```

Vervolgens de rijen één voor één toegevoegd aan de JSONResultSetResponse object klasse middels onderstaande code. Hierdoor wordt de opgehaalde data uit de FileNet Content Manager repository toegevoegd aan een JSON object. Dit JSON object wordt geretourneerd naar de data access laag, zodat de ResultSet model klasse deze data kan verwerken en laten zien op de presentatie laag. In de bijlage **(Bijlage M: Zoek plugin lay-out)** heb ik een afbeelding toegevoegd waarin is te zien hoe de resultaten in de zoek plugin lay-out worden getoond in de ContentList Dojo widget die ik heb geïmplementeerd.

```
jsonResultSet.addRow(row);
```

8.4 Bouwen van Dojo widgets m.b.t. ingevoerde data

In de vorige sprint was het mij niet gelukt om de datum velden werkend te krijgen. Dit bleek lastiger te realiseren dan ik van ter voren had gedacht. De stakeholder wil een document klasse kunnen opzoeken op datum aangemaakt en- of datum gewijzigd. Deze sprint ben ik daar mee verder gegaan.

Als eerste heb ik een Dojo variabele gekoppeld aan de DateTextBox widget door gebruik te maken van de data-dojo-attach-point=”<variabelen>”. Middels JavaScript kan ik vervolgens de ingevoerde waarde ophalen en daar een actie mee uitvoeren. Zie onderstaande code voor de implementatie hiervan.

```
in-right: 20px;">Datum aangemaakt:</label>
 data-dojo-attach-point="dateCreated" data-dojo-type="dijit/form/DateTextBox"


in-right: 33px;">Datum gewijzigd:</label>
 data-dojo-attach-point="dateChanged" data-dojo-type="dijit/form/DateTextBox"
```

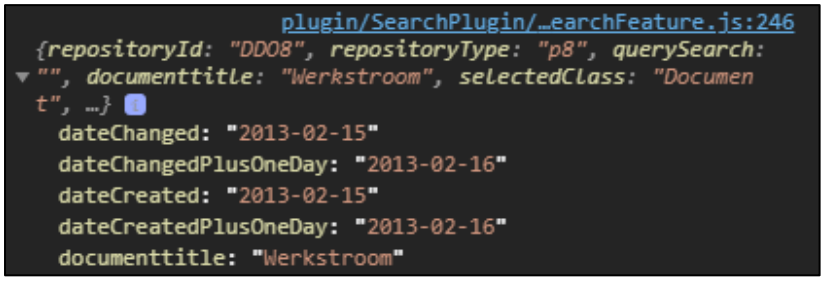
Eén van de problemen waar ik in de vorige sprint tegen aanliep is het converteren van het formaat van de datum naar het juiste formaat (yyyy-MM-dd). Dit heb ik uiteindelijk kunnen oplossen door de waarde op een andere manier op te halen. Het bleek namelijk dat het ook mogelijk is om de daadwerkelijke ingevoerde waarde op te halen door de “displayedValue” te gebruiken i.p.v. “Value”. Zie onderstaande code voor de implementatie.

```
requestParams.dateCreated = this.dateCreated.get("displayedValue");
requestParams.dateChanged = this.dateChanged.get("displayedValue");
```

Daarna liep ik tegen een ander probleem aan. In de service interface wordt middels een query de benodigde data opgehaald. Maar met een datum is het niet mogelijk om de exacte datum op te halen, ofwel de operator "=" werkt natuurlijk niet. Een oplossing die ik hiervoor gevonden heb is het ophalen van de datum van een dag later. Hierdoor kon ik de juiste datum ophalen door de operator ">" en "<" te gebruiken in de query. Het ophalen van de datum van een dag later heb ik met onderstaande code gerealiseerd.

```
var dateCreatedNew = new Date(this.dateCreated.get("displayedValue"));
var dateCreatedPlusOneDay = new Date(dateCreatedNew);
dateCreatedPlusOneDay.setDate(dateCreatedPlusOneDay.getDate()+1);
var year = dateCreatedPlusOneDay.getFullYear().toString();
var month = (dateCreatedPlusOneDay.getMonth() + 101).toString().substring(1);
var day = (dateCreatedPlusOneDay.getDate() + 100).toString().substring(1);
var dateCreatePlusOneValue = year + "-" + month + "-" + day;
requestParams.dateCreatedPlusOneDay = dateCreatePlusOneValue;
```

In de console log kon ik testen of de juiste datum format werd opgeslagen. Het JSON object is te zien in onderstaande console log afbeelding.



8.5 Bouwen van de totaal aantal te retourneren resultaten

In het UI-ontwerp **(Figuur 4: UI-ontwerp van de zoek plugin)** heeft de stakeholder aangeven waar de elementen op de lay-out van de plugin moeten komen te staan. Eén van deze elementen is om aan te geven hoeveel resultaten er geretourneerd moeten worden tijdens het uitvoeren van een zoek actie.

Op de presentatie laag heb ik hiervoor een Dojo widget geïmplementeerd waarin de gebruiker een getal waarde kan invoeren. In onderstaande code is de implementatie hiervan te zien. De variable totalSearch houdt de ingevoerde waarde vast.

```
<input id="" data-dojo-attach-point="totalSearch" data-dojo-type="dijit/form/TextBox"
    placeholder="Aantal te retourneren"></input>
```

Het ophalen van de waarde ging niet zoals verwacht. Als de gebruiker namelijk niets invoerde, dan kon de totalSearch variable niet opgevraagd worden. Om dit op te lossen heb ik een simpele if else statement gemaakt waarin wordt gecontroleerd of er iets is ingevuld. De implementatie hiervan is zichtbaar in onderstaande code.

```
if(this.totalSearch.get("value")){
    requestParams.totalSearch = this.totalSearch.get("value");
}else{
    requestParams.totalSearch = 0;
}
```

8.6 Retrospective

Wat heeft deze sprint opgeleverd?

- Service laag van de zoek plugin gerealiseerd.
- De datum velden werkend gekregen.
- Zoek plugin beschikt over de gewenste zoek functionaliteiten.

Wat ging er goed?

- Het vinden van een oplossing voor de juiste format voor de datum velden. Tijdens de vorige sprint had ik hier niet genoeg tijd aan besteed. Ik wist al snel een oplossing te vinden hiervoor.

Wat ging er minder goed?

Wat zijn de leerpunten?

- De volgende keer wil ik meer aandacht besteden aan het bouwen van de service laag. Sommige onderdelen in de service laag kunnen naar mijn mening iets netter gebouwd worden.

9 Sprint 5 – Testen en afronden

Tijdens sprint 2, 3 en 4 ben ik bezig geweest met het ontwerpen en ontwikkelen van maatwerk in de vorm van een plugin voor de niet out-of-the-box functionaliteiten die betrekking hebben tot de gewenste zoek functionaliteiten in Content Navigator (user story DD2-4606 en DD2-4607). Tijdens deze sprint ben ik bezig geweest met het testen van de geïmplementeerde (niet) out-of-the-box functionaliteiten. Tijdens het testen heb ik op advies van mijn begeleider het testproces gevolgd dat het Digidoc Online team hanteert. Ik ben begonnen met het opstellen van een testplan om een ieder die betrokken is bij het testproces te informeren over de aanpak, de activiteiten, de scope en de op te leveren eindproduct met betrekking tot het testtraject voor de beheertooling in Content Navigator. Vervolgens ben ik na goedkeuring van het testplan bezig geweest met het opstellen van testscripts om de gerealiseerde (niet) out-of-the-box functionaliteiten functioneel te testen in Content Navigator. Ook ben ik deze sprint bezig geweest om de laatste unittests te schrijven om de minimale code coverage van 50% te borgen. De resultaten, bevindingen van de testscripts en unittests heb ik vervolgens beschreven in het testverslag, gevolgd door een advies voor mijn opdrachtgever.

9.1 Sprint planning

De items in onderstaande tabel **(Tabel 15: Sprint backlog van sprint 5)** zijn door de product owner op de sprint backlog geplaatst en worden deze sprint opgepakt. Het backlog traceability Excel document **(Bijlage H: Backlog traceability)** heb ik ook bijgewerkt.

Jira ID	*Item (Taak)*	*Prioriteit*	*Points*
DD2-4915	Configureren van de out-of-the-box functionaliteiten	High	2
DD2-4911	Opstellen van een testplan	High	3
DD2-4934	Opstellen van testscripts	High	2
DD2-4935	Uitvoeren van de testscripts	High	2
DD2-4936	Verwerken van de testresultaten	High	2
DD2-4937	Schrijven van de benodigde unittests	High	2

Tabel 15: Sprint backlog van sprint 5

9.2 Configureren van de out-of-the-box functionaliteiten

Tijdens de eerste sprint heb ik diverse work break downs uitgevoerd om te bepalen welke werkzaamheden er uitgevoerd moeten worden om de user story te maken c.q. voltooien. Het bleek dat vier van de zes user stories waarvoor ik een work break down heb uitgevoerd, out-of-the-box geïmplementeerd kunnen worden. Voor die user stories heb ik configuratie werkzaamheden uitgevoerd in

Content Navigator om ze te kunnen realiseren. Dit zijn de user stories met de hoogste prioriteit. Zie onderstaande tabel.

Jira ID	*Item (user story)*	*Prioriteit*
DD2-4614	Als beheerder wil ik documenten kunnen kopiëren en plakken d.m.v. ctrl-c/v	Highest
DD2-4602	Als beheerder wil ik tijdens het creëren van een folder alle properties kunnen specificeren	High
DD2-4613	Als beheerder wil ik documenten kunnen kopiëren en plakken d.m.v. verslepen	High
DD2-4612	Als beheerder wil ik tijdens het creëren van een document alle properties kunnen specificeren	High

Echter bleek dat user story DD2-4614 niet geïmplementeerd kon worden. In Content Navigator is het niet mogelijk om een ctrl-c/v actie uit te voeren. Dit blijkt uit de documentatie van IBM. Dit heb ik ook tijdens het uitvoeren van de work break down voor deze user story beschreven.

Voor user story DD2-4613 en DD2-4612 heb ik tijdens de work break down aangegeven dat er voor deze twee user stories een entry template aangemaakt moet worden. Uit de documentatie van IBM blijkt dat een entry template geconfigureerd kan worden om documenten en/of folders aan te maken. Daarbij kan er in een entry template worden aangegeven welke eigenschappen er gedefinieerd (moeten) worden tijdens het aanmaken van een document of folder.

9.3 Controleren user stories a.d.v. de Definition of Done

In het plan van aanpak heb ik een aantal kwaliteitseisen **(Tabel 2: Definition of Ready en Definition of Done)** opgesteld om te bepalen wanneer een user story is afgerond. Deze kwaliteitseisen heb ik tijdens de realisatie van de user stories regelmatig gecontroleerd. Voor twee niet out-of-the-box functionaliteiten (user story DD2-4606 en DD2-4607) heb ik maatwerk ontwikkeld in de vorm van een plugin. Beide user stories zijn functioneel getest a.d.v. opgestelde testscripts. De code is geschreven volgens de coding standaarden. Deze standaarden heb ik tijdens het schrijven van de code constant bijgehouden om ervoor te zorgen dat ik voldoe aan de coding standaarden. Hiermee heb ik voor deze twee user stories de volgende kwaliteitseisen afgevinkt:

- ✓ Source code is geschreven volgens de coding standaarden die zijn vastgelegd.
- ✓ Er zijn unittests geschreven die voldoen aan de kwaliteitscriteria die zijn vastgelegd in het testplan.
- ✓ Eventuele (technische) documentatie is bijgewerkt.
- ✓ Mijn begeleider heeft een functionele UI-test uitgevoerd a.d.v. opgestelde testscripts.

9.4 Opstellen van een testplan

Aan het begin van het testproces ben ik bezig geweest met het opstellen van een testplan om een ieder die betrokken is bij het testproces te informeren over de aanpak, de activiteiten, de scope en de op te leveren eindproduct met betrekking tot het testtraject voor de beheertooling. Mijn begeleider heeft mij geadviseerd om het huidige testplan en de testprocessen van het Digidoc Online te inventariseren en toe te passen voor dit project. Twee belangrijke testsoorten die het Digidoc Online team gebruikt is de UI-test en unittest. Een UI-test bestaat uit het functioneel testen van de user story. Aan de hand van opgestelde testscripts voer ikzelf en mijn begeleider de verschillende testscripts uit. Unittesten wordt toegepast om de geschreven code afzonderlijk te testen. Voor de geschreven unittests geldt een minimale code coverage van 50%. Dit is ook de minimale code coverage die het Digidoc Online team hanteert.

9.4.1 Testsoorten

De volgende testsoorten worden uitgevoerd:

Test	Omgeving	Input	Output
UI-test	O8 (ontwikkelomgeving)	Testscripts	Testverslag
Unittest	O8 (ontwikkelomgeving)	Testcode	Testverslag

9.5 Opstellen van de testscripts

Voor alle user stories op de product backlog heb ik testscripts opgesteld om het gewenste gedrag van de functionaliteit in Content Navigator te testen. Ik heb de structuur van de huidige testscripts van het Digidoc Online gebruikt om testscripts op te stellen. Daarbij heb ik één extra kolom toegevoegd waarin wordt aangegeven of de functionaliteit out-of-the-box gerealiseerd is. Ik heb dit gedaan om in één oogopslag te zien of de user story out-of-the-box is gerealiseerd.

Omschrijving	Een omschrijving van de te testen functionaliteit
User story	De titel van de user story die functioneel wordt getest
Jira ID	Het id van de te testen user story
Out-of-the-box in Content Navigator	Hier wordt aangegeven of de user story out-of-the-box geïmplementeerd kon worden
Te ondernemen Acties	Hier staan de stappen beschreven die uitgevoerd moeten worden om te testen of de functionaliteit werkt zoals verwacht
Verwacht resultaat	Het verwachte resultaat wordt hier beschreven

9.6 Verwerken van alle testresultaten

In dit hoofdstuk beschrijf ik de verwerking van de testresultaten.

9.6.1 Testscripts

Mijn begeleider heeft de testscripts uitgevoerd en de volgende bevindingen genoteerd. De oranje gekleurde status kolommen zijn niet uitgevoerd, omdat de functionaliteit van de testscript buiten scope zijn. Voor deze functionaliteiten dient er nieuw maatwerk ontwikkeld moet worden.

Case ID	Status	Opmerkingen	Datum	Tester
WBT_UI_1		Niet gedaan		
WBT_UI_2	Succes		17-12-20	JdH
WBT_UI_3	Succes		17-12-20	JdH
WBT_UI_4	Niet geslaagd	Fout met de backend	17-12-20	JdH
WBT_UI_5		Niet gedaan		
WBT_UI_6		Niet gedaan		
WBT_UI_7	Succes		17-12-20	JdH
WBT_UI_8	Niet geslaagd	Ik zoek er 16, hij vindt er 500 en toont er 25. Zie screenshot	17-12-20	JdH
WBT_UI_9		Niet gedaan		
WBT_UI_10		Niet gedaan		
WBT_UI_11		Niet gedaan		
WBT_UI_12		Niet gedaan		
WBT_UI_13		Niet gedaan		

UI-testcase WBT_UI_8 is niet geslaagd, omdat ik een variabelen was vergeten te definiëren in de service interface van de zoek plugin. Dit had ik over het hoofd gezien.

UI-testcase WBT_UI_4 is niet geslaagd, omdat er een fout optrad met de backend van FileNet. Volgens mijn begeleider is dit niet een fout door verkeerde configuratie aan de backend kant. Ik heb dit dan ook als bug gemeld bij het backend team van Digidoc.

9.6.2 Unittests

Door de benodigde connectiviteit met de FileNet Content Manager repository is het niet gelukt om voor de service interface unittests te schrijven en de minimale code coverage van 50% te borgen. Ik heb geprobeerd op de Spring mock framework gebruiken. Echter wilde dit niet werken en merkte ik dat ik er teveel tijd aan spendeerde. Mijn begeleider heeft aangegeven dat het backend team

van Digidoc momenteel tegen hetzelfde probleem aanloopt en weigeren om een mock framework te gebruiken. De service interface heb ik op advies van mijn begeleider buiten scope gelaten bij het schrijven van de unittests. Hierdoor kom ik op een totale code coverage van 51% excl. de service interface. De totale code coverage incl. service interface komt op 30,2%. Zie onderstaande afbeelding voor het totale overzicht van de code coverage.

Element	Coverage	Covered Lines	Missed Lines	Total Lines
SearchPlugin	30,2 %	89	206	295
src	30,2 %	89	206	295
com.ibm.ecm.searchplugin	13,4 %	32	206	238
SearchServiceP8.java	0,0 %	0	115	115
SearchServiceP8	0,0 %	0	115	115
SearchService.java	3,0 %	2	65	67
SearchService	3,0 %	2	65	67
SearchPlugin.java	57,1 %	24	18	42
SearchPlugin	57,1 %	24	18	42
SearchFeature.java	42,9 %	6	8	14
SearchFeature	42,9 %	6	8	14
test	100,0 %	57	0	57
SearchPluginTest.java	100,0 %	57	0	57
SearchPluginTest	100,0 %	57	0	57

Omdat de service interface buiten scope is gelaten is de minimale code coverage van 50% behaald. Unittests voor de service interface kunnen worden geschreven zodra er een passende oplossing is gevonden om de service interface op een juiste manier te unittesten.

9.7 Schrijven van een advies voor de opdrachtgever

De scope van dit project was het implementeren van minimaal één niet out-of-the-box functionaliteit. Voor twee niet out-of-the-box functionaliteiten heb ik maatwerk ontwikkeld in de vorm van een plugin. Mijn advies is om voor de overige niet out-of-the-box functionaliteiten maatwerk te ontwikkelen. Daarbij kunnen gewenste zoek functionaliteiten in de bestaande zoek plugin geïmplementeerd worden. Mijn advies is om voor de functionaliteiten die geen betrekking hebben tot het uitvoeren van zoek acties een nieuwe plugin te bouwen. De functionaliteiten in Content Navigator zijn namelijk verdeeld over verschillende secties. Dit komt ook terug in de analyse die ik heb uitgevoerd, zie **(Tabel 6: Functionaliteiten die ontbreken in de ACCE)**. Ook komt deze verdeling van functionaliteiten terug in de documentatie van Content Navigator. Door niet alle functionaliteiten in één plugin te ontwikkelen wordt de geschreven code ook beter leesbaar en onderhoudbaar. Dit zijn één van de coding standaarden **(Bijlage I: Coding standaarden)** van het Digidoc Online team.

Door de benodigde connectiviteit met de FileNet Content Manager repository is het niet gelukt om voor de service interface unittests te schrijven. Hierdoor is de totale code coverage incl. de service interface van 50% niet behaald. Het backend team loopt momenteel tegen hetzelfde probleem aan en weigeren het om een mock framework te gebruiken. Ik heb een poging gewaagd om de Sping framework te gebruiken, maar dit ging zonder succes. Mijn advies aan het

backend team is om meer aandacht te besteden aan het vinden van een andere oplossing hiervoor. Mijn begeleider heeft aangegeven dat ze hier mee bezig zijn. Maar omdat ik ook tegen dit probleem aan ben gelopen heb ik het gevoel gekregen dat er nog meer vaart achter wordt gezet.

9.8 Retrospective

Wat heeft deze sprint opgeleverd?

- Een gedetailleerde testplan voor het testen van de gerealiseerde (niet) out-of-the-box functionaliteiten
- Opgestelde testscripts om de functionaliteiten functioneel te testen in Content Navigator.
- Een testverslag waarin de bevindingen van de testresultaten zijn beschreven, gevolgd door een advies voor mijn opdrachtgever.
- Nieuwe kennis van de Java Sprint Mock framework (ondanks dit uiteindelijk niet het gewenste resultaat gaf).
- De beheerders van Digidoc hebben direct na oplevering gebruik gemaakt van de beheertooling in Content Navigator.

Wat ging er goed?

- Ik ben zeer tevreden over hoe ik het algehele testproces heb uitgevoerd.

Wat ging er minder goed?

- Het testen van de service laag is door de benodigde FileNet connectiviteit niet gelukt. Ik heb veel tijd besteed aan het vinden van een oplossing door het gebruik van een mock framework. Helaas zonder succes.

Wat zijn de leerpunten?

10 Projectresultaat

In dit hoofdstuk beschrijf ik de link tussen het opgeleverde resultaat en de aanvankelijke probleemstelling.

10.1 Doelstelling

De doelstelling van dit project is het realiseren van een nieuwe beheertooling binnen de standaard FileNet web interface (Content Navigator), die voorziet in de functionele behoeftes van de beheerders van het documentmanagementsysteem Digidoc. Daarbij ligt de focus van dit project op het ontwikkelen van maatwerk voor minimaal één niet out-of-the-box functionaliteit voor de beheertooling in Content Navigator. Met uiteindelijk als doel dat Content Navigator de FEM beheerclient en zelfgemaakte beheerscripts vervangt.

10.2 Opgeleverde resultaat

De opdrachtgever wilde een beheertooling in Content Navigator die voorziet in de functionele behoeftes van de beheerders van het documentmanagementsysteem Digidoc. De focus van dit project lag op het ontwikkelen van minimaal één niet out-of-the-box functionaliteit. Ik heb voor twee niet out-of-the-box functionaliteiten maatwerk ontwikkeld in de vorm van een plugin die beschikt over gewenste zoek functionaliteiten. Deze plugin is beschikbaar gesteld in de nieuwe desktop die aan het begin van het project is aangemaakt in Content Navigator. De lay-out van de plugin heb ik als bijlage toegevoegd **(Bijlage M: Zoek plugin lay-out)**. In deze bijlage is het resultaat te zien van de user interface van de zoek plugin die ik dit project heb gerealiseerd. Daarnaast heb ik voor de out-of-the-box functionaliteiten met een hoge prioriteit configuratie werkzaamheden uitgevoerd en gerealiseerd.

Het eindproduct voldoet aan de verwachting van de opdrachtgever. Een achterliggende gedachte van de opdrachtgever was om op dit niveau te komen om zo handvatten te hebben om de beheertooling in Content Navigator uit te bouwen. De FEM wordt op dit moment nog minimaal ingezet om de werkzaamheden uit te voeren die nog niet in de beheertooling in Content Navigator zijn gerealiseerd. Er wordt ook al gekeken welke mogelijkheden er zijn om de beheertooling generiek toe te passen op alle FileNet omgevingen.

11 Evaluatie

In dit hoofdstuk beschrijf ik de evaluatie van het proces, de vooraf afgesproken producten en de beroepstaken.

11.1 Procesevaluatie

Voor het uitvoeren van deze opdracht was ik voor de duur van mijn afstudeerperiode binnen de afdeling ECM FileNet gedetacheerd bij het Digidoc Online team. De afdeling ECM FileNet was voor mij een nieuwe omgeving waar ik in terecht kwam met niet alleen nieuwe collega's, maar ook daarbij veel nieuwe technieken rondom de FileNet en Content Navigator omgevingen. Het was voor mij aan het begin even wennen, omdat ik met mijn nieuwe collega's alleen online kon afspreken. Het even naar elkaar toe lopen om iets te vragen aan een collega had ik op sommige momenten erg handig gevonden. Vooral op de momenten dat ik even snel een korte vraag wilde stellen aan een collega. Naarmate verder in het proces ging het afspreken met collega's steeds soepeler en sneller doordat ik de collega's steeds beter leerde kennen.

Ik merkte aan het begin van het proces dat ik te snel vooruit wilde gaan, omdat ik nog helemaal geen beeld had van het eindproduct en wat ik precies moest gaan uitvoeren om het eindproduct te vormen. Dit kwam vooral door de nieuwe omgeving waarin technieken worden gebruikt waarmee ik nog nooit eerder had gewerkt. Ik ben iemand die op dat soort momenten het liefst gelijk alles wil begrijpen van de techniek en ben hier dan ook bewust van. Mijn begeleider merkte dit ook op en gaf aan dat het zeker goed is om zoveel mogelijk informatie te verzamelen over de nieuwe technieken en vond dit leuk om te zien, maar gaf ook aan dat de leercurve van de FileNet technieken stijl is en het tijd kost om daarover kennis op te bouwen. Dit heb ik gaande het proces zeker gemerkt. Ik stelde aan het begin ook al vragen omtrent de techniek van Content Navigator die op dat moment nog helemaal niet relevant waren. Gaande weg het proces merkte ik dat ik steeds meer kennis omtrent de Content Navigator en FileNet omgeving opbouwde. Het kwartje begon steeds meer te vallen. Daarnaast heb ik tijdens het proces momenten gehad dat ik onzeker was over bepaalde beslissingen die ik had genomen. Bijvoorbeeld beslissingen over welke interfaces toegepast moeten worden van de Content Navigator API of beslissingen die ik nam tijdens het schrijven van de code voor de plugin. Achteraf gezien bleek dat deze beslissingen resulteerde in het gewenste resultaat. Nu ik terug kijk op dit proces vind ik het mooi om te zien dat ik vrijwel zonder kennis van de technieken die worden gebruikt rondom FileNet en Content Navigator in een korte periode zo veel verder ben gegroeid op dit gebied. Dan heb ik het over technieken zoals de mogelijkheden die de API van Content Navigator biedt, de structuur en mogelijkheden binnen Content Navigator omtrent ontwikkelen en beheren en de wijze van het ontwikkelen van maatwerk voor Content Navigator door het toepassen van de Dojo Toolkit.

Ik ook gemerkt dat ik het op sommige momenten lastig vond om de werkzaamheden die ik had uitgevoerd tijdens de realisatie van de zoek plugin te beschrijven in mijn verslag. Door het zelfstandig werken was ik namelijk vaak dingen aan het uitproberen en bouwen om het gewenste resultaat te behalen.

Tijdens het gehele proces heb ik volgens de Scrum methodiek gewerkt dat het Digidoc Online team hanteert. Ik vond het fijn om in vaste time boxes van twee weken te werken en hierin concrete taken te hebben. Door het gebruik van Scrum vond er regelmatig communicatie plaats met mijn opdrachtgever. Hierdoor heb ik regelmatig feedback ontvangen op de producten. Een aantal Scrum technieken waren voor mij nieuw. Dat was het inschatten van de hoeveelheid werk van een product backlog taak a.d.v. story points die het Digidoc Online team gebruikt en het uitvoeren van een product backlog refinement. Het inschatten van de hoeveelheid werk van een taak was aan het begin erg lastig. Ik had namelijk helemaal geen ervaring met de werkzaamheden die uitgevoerd moesten worden. Na sprint 2 merkte ik dat het inschatten van de hoeveelheid werk mij steeds beter af ging door de ervaring die ik heb opgedaan tijdens de vorige sprints. Het uitvoeren van een product backlog refinement was voor mij ook nieuw. Ik wist ook helemaal niet dat dit een onderdeel was van het Scrum proces. De aangebrachte details aan de user story hebben mij geholpen om de hoeveelheid tijd en werk van een user story beter te kunnen in te schatten.

11.2 Productevaluatie

In dit hoofdstuk evalueer ik terug op de vooraf afgesproken producten voor mijn opdrachtgever.

Analysedocument

De opstart van de analyse ging wat moeizaam, omdat de het in het begin nog niet duidelijk was wie de key stakeholder was tijdens het project. Als eerst heb ik een kennismakingsgesprek gehad met de key stakeholder van de Digidoc beheerders. Tijdens deze kennismaking kwam ik erachter dat er al eerder een analyse is uitgevoerd, maar dat een nieuwe analyse noodzakelijk is. Na dit gesprek was het mij een stuk duidelijker geworden waarom er een nieuwe analyse uitgevoerd moest worden en wat er verwacht wordt. Een nieuwe versie van Content Navigator en de ACCE waren één van de belangrijkste redenen. In vervolg gesprekken met de key stakeholder hebben we gezamenlijk de bestaande analysedocumenten doorgenomen om te kijken in hoeverre de bestaande analyses nog kloppen. Vervolgens zijn de functionaliteiten geprioriteerd en zijn deze vertaald naar user story omschrijvingen. De analyse heeft bij de stakeholder en bij mij een duidelijk beeld gecreëerd van het probleemdomein waarbij de gewenste functionaliteiten in kaart zijn gebracht. Op basis van de analyse is de product backlog tot stand gekomen en zijn de user stories geprioriteerd door de product owner.

Technische documentatie (UML-diagrammen)

In het afstudeerplan is er aangeven dat er technische documentatie wordt opgeleverd voor de opdrachtgever. Onder technische documentatie vallen de verschillende UML-diagrammen die ik heb opgesteld tijdens de projectuitvoering. Mijn begeleider vond het niet nodig om deze in een aparte bijlage op te nemen. Om die reden zijn de losse UML-diagrammen met de beschrijvingen opgenomen in het afstudeerverslag. Ik ben over het algemeen zeer tevreden over het resultaat van de verschillende ontwerpen die ik heb gemaakt voor de plugin. Ondanks het bepalen van de juiste API interfaces niet helemaal ging zoals gepland. Hier dacht ik namelijk iets te makkelijk over. Tussentijds heeft het ontwerp nog een wijziging ondergaan door nieuwe inzichten die werden gecreëerd. De wijziging had verder geen impact op de duur van het project.

Testrapport

Tijdens het testproces heb ik twee belangrijke testsoorten gebruikt die het Digidoc Online team ook gebruikt bij het testen van de applicatie Digidoc. Het gewenste gedrag van de geïmplementeerde user stories zijn a.d.v. opgestelde testscripts functioneel getest in Content Navigator. Voor de geschreven code voor de plugin heb ik unittests geschreven om de minimale code coverage van 50% te borgen. Door de benodigde connectiviteit met FileNet is het helaas niet gelukt om unittests te schrijven voor de service laag. Gezien de tijd kon ik hier helaas niks meer aan doen. Op advies van mijn begeleider heb ik daarom de service laag buiten scope gelaten en heb ik in het testverslag het advies gegeven aan het backend team om meer aandacht te besteden aan het vinden van een passende oplossing hiervoor. De bevindingen / resultaten van de uitgevoerde testscripts en unittests heb ik beschreven in het testverslag gevolgd door een advies voor de opdrachtgever.

Eindproduct

Ik ben zeer tevreden over het eindproduct. Uiteindelijk heb ik niet alle eisen / wensen kunnen implementeren. Echter alle out-of-the-box functionaliteiten zijn dat wel. De focus van dit project lag op het ontwikkelen van minimaal één niet out-of-the-box functionaliteit. Voor twee niet out-of-the-box functionaliteiten heb ik maatwerk ontwikkeld in de vorm van een plugin die beschikt over gewenste zoek functionaliteiten. Met de plugin kunnen de Digidoc beheerders zoeken naar ongestructureerde informatie waarbij er ook kan worden aangegeven hoeveel resultaten geretourneerd moeten worden bij het uitvoeren van een zoek actie. Ook is er zichtbaar hoeveel resultaten de zoek actie daadwerkelijk heeft geretourneerd. Omdat er nieuw maatwerk is ontwikkeld in de vorm van een plugin biedt het de mogelijkheid om de plugin in de toekomst verder te kunnen uitbouwen met nieuwe wensen.

Uiteindelijk durf ik te zeggen dat de doelstelling van dit project behaald is. De noodzakelijke functionaliteiten zijn geïmplementeerd en de opdrachtgever is tevreden met het eindproduct. Een achterliggende gedachte van de

opdrachtgever was om op dit niveau te komen om zo handvatten te hebben om de beheertooling in Content Navigator uit te bouwen. Er wordt ook al gekeken welke mogelijkheden er zijn om de beheertooling generiek toe te passen op alle FileNet omgevingen.

11.3 Beroepstaken

Ik heb de STARR-methode gebruikt om te reflecteren op de gekozen beroepstaken, waarbij ik antwoord geef op de vragen over de situatie, taken, actie, het resultaat en de reflectie. In de reflectie geef ik ook aan wat ik de volgende week weer zou doen en- of wat ik de volgende keer anders zou doen.

Analyseren probleemdomein & opstellen probleemstelling

STARR	Uitwerking
Situatie	Van ter voren heb ik voor de opdracht in samenwerking met de opdrachtgever een probleemstelling opgesteld. Als onderdeel van deze opdracht moest er een analyse van het probleemdomein uitgevoerd worden om de gewenste functionaliteiten in kaart te brengen.
Taken	Het was mijn taak om het probleemdomein in kaart te brengen door bestaande analyse documenten te verzamelen en te verwerken. Daarbij was het mijn taak om de stakeholder hierbij te betrekken.
Actie	Ik heb diverse analyse documenten verzameld en verwerkt in de analyse die ik heb uitgevoerd. Daarbij hebben er in een korte periode verschillende gespreken plaatsgevonden met de stakeholder om de analyse documenten verder toe te lichten en de gewenste functionaliteiten in kaart te brengen.
Resultaat	Het uitgangspunt van deze analyse was een lijst met geprioriteerde eisen en wensen met daarbij opgestelde user story omschrijvingen. Het analyseren van het probleemdomein was een belangrijk onderdeel van dit project. Hierdoor heb ik de huidige situatie en gewenste situatie goed in kaart kunnen brengen.
Reflectie	Het in kaart brengen van het probleemdomein om te achterhalen wat de eisen en wensen zijn van de stakeholder was een belangrijk onderdeel van dit project. Over het algemeen ben ik zeer tevreden over de uitvoering van de analyse. De communicatie tussen mij en de stakeholder verliep goed.

Kritisch, onderzoekend & methodisch werken

STARR	Uitwerking
Situatie	Aan het begin van het project is er door mijn begeleider bepaald dat ik volgens de Scrum methodiek ga werken die het Digidoc Online team hanteert. Scrum is vooral methodisch werken, het legde een raamwerk neer hoe ik op een Agile manier producten kon opleveren en software kon ontwikkelen. Tijdens het schrijven van het afstudeerverslag en de producten is er op een kritische wijze gekeken naar de geschreven teksten. Ook is er tijdens het project periodiek om feedback gevraagd over het inhoudelijke afstudeerverslag en de producten.
Taken	Tijdens de Scrum uitvoering heb ik deels de rol Scrum master vervuld en de rol als projectuitvoerder / ontwikkelaar. Mijn taak was het in goede banen leiden van het project en het uitvoeren van de werkzaamheden. Daarnaast was het mijn taak om de user stories op te zetten en te werken in sprints van bepaalde lengte. Ook was het mijn taak om kritisch te kijken naar de geschreven teksten van de producten en het afstudeerverslag.
Actie	Tijdens het project heb ik volgens de Scrum methodiek gewerkt die het Digidoc Online team hanteert. Daarnaast heb ik tijdens het project periodiek om feedback gevraagd. Ook tijdens het wekelijkse overleg met mijn begeleider hebben we de producten bekeken. Aan het einde van elke sprint heb ik samen met mijn begeleider de sprint geëvalueerd, waarbij we ook kritisch hebben gekeken wat er (niet) goed ging, wat de sprint heeft opgeleverd en welke leerpunten ik hieruit heb gehaald. Ook is er regelmatig op een kritische wijze gekeken naar de geschreven teksten in het afstudeerverslag.
Resultaat	Het methodisch werken met Scrum heeft geresulteerd in het opleveren van de vooraf afgesproken producten. Ook heeft het werken met Scrum geholpen in het op een iteratieve en incrementele wijze werken aan de niet out-of-the-box functionaliteiten. Dit heeft geresulteerd in een maatwerk plugin die beschikt over gewenste zoek functionaliteiten. Door het gebruik van Scrum kon ik ook snel inspelen in veranderingen. Tijdens sprint 2 kwam ik er namelijk achter dat de niet out-of-the-box functionaliteiten in één plugin gerealiseerd kunnen worden. Hierdoor heb ik in sprint 3 moeten bijsturen.
Reflectie	Ik ben voldoende methodisch te werk gegaan door het Scrum proces dat ik heb gehanteerd tijdens het project. Door de

	duidelijk afspraken die vooraf zijn gemaakt verliep de Scrum uitvoering vrij soepel. De volgende keer zou ik alleen wat meer tijd willen besteden aan de retrospective. Voor mijn gevoel ging dit proces soms wat te snel terwijl dit een belangrijk onderdeel is van het Scrum proces. Ik was achteraf blij dat er wekelijkse overleggen plaatsvonden met mijn begeleider. Hierdoor compenseerde dit weer.

Leren leren: voorbereiden op volgende studiefase & beroep

Tijdens dit project zijn er veel nieuwe dingen aan het licht gekomen. Op het gebied van software ontwikkeling heb ik veel geleerd over de programmeertaal Java in combinatie met de Content Navigator API. Het toepassen van de Dojo Toolkit was één van de grootste en spannende uitdaging tijdens dit project. Ik had namelijk nooit met de Dojo Toolkit gewerkt, ook niet tijdens de studie. Om te leren hoe de Dojo Toolkit werkt, moest ik dingen gaan uitzoeken, uitproberen en toepassen. Ook de mogelijkheden van Content Navigator API kende ik nog niet. Op dit gebied heb ik in een korte tijd veel kunnen leren en toepassen. Ik heb tijdens bouwen van de zoek plugin vooral zelfstandig gewerkt, omdat ik niet de mogelijkheid had om even snel dingen te vragen aan collega's die daar kennis over hebben. Het zelfstandig werken heeft ervoor gezorgd dat ik in een korte tijd veel nieuwe kennis heb opgedaan over technieken die ik voor het project nog niet kende.

Ontwerpen software

STARR	Uitwerking
Situatie	Van ter voren is bepaald dat er voorafgaand het realisatie proces van de niet out-of-the-box functionaliteiten een technisch ontwerp gemaakt moet worden met UML.
Taken	Mijn taak was het maken van de UML-diagrammen en de gemaakte ontwerpkeuzes toe te lichten.
Actie	Ik heb twee klassendiagrammen gemaakt die de onderlinge relaties tussen de Java klassen in de plugin vastleggen. Daarbij heb ik een sequentiediagram gemaakt die de interactie tussen de verschillende lagen in de Content Navigator plugin toont. De UML-diagrammen heb ik gemaakt in Visual Paradigm.
Resultaat	Dit heeft geresulteerd in drie UML-ontwerpen. Twee klassendiagrammen van de zoek plugin en één sequentiediagram die de verschillende interacties tonen tussen de MVC-lagen van de zoek plugin.
Reflectie	De ontwerpkeuzes die ik heb moeten maken waren lastiger dan ik van ter voren had gedacht. De API van Content Navigator is

	vrij uitgebreid en biedt een groot scala aan Java / JavaScript interfaces die gebruikt kunnen worden. Door nieuwe inzichten heb ik de klassendiagram tussentijds nog moeten aanpassen. Met het ontwerpen ben ik dan ook langer bezig geweest dan van ter voren ingeschat. Uiteindelijk ben ik tevreden met het resultaat en met de ontwerpkeuzes die ik heb gemaakt. De klassendiagrammen en sequentiediagram gaven mij een duidelijk beeld van de relaties tussen de onderlinge Java klassen van de zoek plugin. De volgende keer zou ik meer tijd spenderen aan het bestuderen van de Content Navigator API documentatie. Ook wil ik de volgende keer meer achter mijn ontwerpkeuzes staan. Tijdens dit project was ik namelijk soms nogal onzeker over bepaalde ontwerpkeuzes, terwijl de keuzes uiteindelijk goed bleken.

Realiseren van software

STARR	Uitwerking
Situatie	Een groot onderdeel van de opdracht bestond uit het ontwikkelen van maatwerk in de vorm van een plugin voor de beheertooling in Content Navigator.
Taken	Vooraf heb ik me georiënteerd op de mogelijkheden van Content Navigator, waarbij ik kennis heb opgedaan omtrent het ontwikkelen van een plugin in Content Navigator. Het was mijn taak om voor gewenste zoek functionaliteiten maatwerk te ontwikkelen in de vorm van een plugin en beschikbaar te in de desktop van Content Navigator.
Actie	Ik heb voor twee gewenste zoek functionaliteiten maatwerk ontwikkeld in de vorm van een plugin. Hiervoor heb ik een presentatie en data access laag gerealiseerd met de Dojo Toolkit en heb ik een service laag ontwikkeld met een object georiënteerde aanpak. Voor de service laag heb ik een service interface gerealiseerd die de communicatie verzorgt tussen de FileNet Content Manager repository en de data access laag. De presentatie, data access en service laag implementeren diverse interfaces / klassen van de Content Navigator API.
Resultaat	De werkzaamheden die ik hiervoor heb uitgevoerd hebben geresulteerd in een maatwerk plugin die beschikt over twee gewenste niet out-of-the-box zoek functionaliteiten.
Reflectie	Het was voor mij de eerste keer dat ik een plugin heb ontwikkeld voor een applicatie. Dit bleek lastiger dan ik van ter voren had gedacht. Mijn begeleider gaf ook al van ter voren aan dat de FileNet omgeving vrij complex in elkaar zit en dat de

	realisatie van niet out-of-the-box functionaliteiten een uitdaging is als je nog geen kennis hebt van FileNet en Content Navigator. Als ik terug kijk op het realisatie proces dan is dat ook zo geweest. Ik liep regelmatig tegen dingen aan waar ik op dat moment geen oplossing voor had. Hierdoor heb ik veel tijd gespendeerd aan het vinden van een oplossing hiervoor. Telkens als ik weer een stapje verder kwam gaf dat mij extra motivatie om weer door te gaan en te werken naar het eind product.

Testen & Evalueren

STARR	Uitwerking
Situatie	Van ter voren heb ik met mijn begeleider afgesproken om de gerealiseerde (niet) out-of-the-box functionaliteiten te testen a.d.v. opgestelde testscripts. Met de testscripts wordt het gewenste gedrag functioneel getest in Content Navigator. Voor de geschreven Java code dien ik unittests te schrijven die minimaal moeten voldoen aan de 50% code coverage.
Taken	Mijn taak was het schrijven van een testplan, de testscripts, de unittests en een testverslag gevolgd door een advies voor de opdrachtgever.
Actie	Ik heb voor elke user story een testscript gemaakt waarin de uit de voeren stappen en het gewenste gedrag zijn beschreven om de functionaliteit functioneel te testen in Content Navigator. Voor de plugin heb ik unittests geschreven om de geschreven code afzonderlijk te testen. De resultaten en bevindingen hiervan heb ik beschreven in een testverslag.
Resultaat	Tijdens sprint 5 heeft dit geresulteerd in een testplan, testscripts, unittests en een testverslag met een advies voor de opdrachtgever.
Reflectie	Over het testproces ben ik zeer tevreden. Ik vind het echter wel jammer dat het niet gelukt is om de totale code coverage van minimaal 50% te borgen. Door de benodigde connectiviteit met de FileNet Content Manager repository was dit helaas niet mogelijk. De volgende keer zou ik het testproces eerder van start willen laten gaan. Hierdoor is het van ter voren al duidelijk wat het testplan is, zodat ik tijdens het realisatie proces al bepaalde onderdelen kan gaan testen. Ik ben van mening dat dit zorgt voor een hogere kwaliteit van het eind product.

12 Referenties

Agilescrumgroup. (sd). *Agile Scrum Group*. Opgehaald van Agilescrumgroup: https://agilescrumgroup.nl

Definition of Ready. (2017). Opgehaald van Scrumbook: https://www.scrumbook.org/value-stream/product-backlog/definition-of-ready.html

IBM. (2014). *Customizing and Extending IBM Content Navigator.* Opgehaald van Redbooks: http://www.redbooks.ibm.com/redbooks/pdfs/sg248055.pdf

IBM. (2020). *IBM Content Navigator documentation*. Opgehaald van IBM: https://www.ibm.com/support/knowledgecenter/SSEUEX_3.0.8/KC_ditamaps/contentnavigator.htm

Jonker, I. (2017). *Continue to develop*. Opgehaald van Ivo Jonker: https://www.ivojonker.nl/?p=571

SSC-ICT. (sd). Opgehaald van SSC-ICT: https://www.ssc-ict.nl/

13 Bijlagen

Bijlagen zijn in een apart document opgenomen i.v.m. complicaties met de nummering.

Bijlage A: Afstudeerplan

Bijlage B: Evaluatieformulier

Bijlage C: Plan van aanpak

Bijlage D: Analysedocument

Bijlage E: Testplan

Bijlage F: Testscripts

Bijlage G: Testverslag

Bijlage H: Backlog traceability

Bijlage I: Coding standaarden

Bijlage J: Work break downs

Bijlage K: Gesprekken

Bijlage L: Aangeleverde analysedocumenten

Bijlage M: Zoek plugin lay-out