Afstudeerverslag

Verbeteren van usability van Spider CMS

Student:	Michiel Verheul
Examinatoren:	Dhr. J.J. van Beijnum
	Dhr. W. van Vliet
afstudeerbedrijf:	Marbel Software B.V. te Noordwijkerhout
bedrijfsmentor:	Dhr. Ing. G. Scheeres
datum:	10 juni 2005
afstudeerrichting:	Informatica en Informatiekunde: VIA
afstudeerblok	2005-1.1
versie:	3.0

Referaat

Verheul, M.

Verbeteren van de usability van Spider-CMS.

Eindverslag in het kader van afstuderen in de opleiding Informatiekunde & Informatica (richting VIA) aan de Haagse Hogeschool te 's-Gravenhage.

De afstudeerstage vond plaats bij Marbel Software B.V. te Noordwijkerhout in de periode van 8 februari 2005 tot en met 10 juni 2005. Doel van het project was het verbeteren van de usability van Spider, een content management systeem dat ontwikkeld is door Marbel Software. In dit kader zijn er 3 web-based editors ontwikkeld waarmee gebruikers eenvoudig CSS en XML-documenten kunnen aanpassen.

De volgende descriptoren zijn van toepassing op dit document:

XML, CMS, XSLT, ASP.NET, C#, Usability, IAD

Voorwoord

Voor u ligt het eindverslag van mijn afstudeerstage die ik in de periode van 8 februari 2005 tot en met 10 juni 2005 bij Marbel Software in Noordwijkerhout heb doorlopen.
Ik heb de opdracht die redelijk technisch van aard was met veel plezier uitgevoerd.
Ik wil mijn bedrijfsmentor dhr. Scheeres en andere werknemers van Marbel Software bedanken voor de begeleiding en in het bijzonder de vrijheid die ik heb gekregen tijdens het uitvoeren van mijn afstudeeropdracht. Door deze vrijheid ben ik in staat geweest om me te ontwikkelen op nieuwe vlakken.
Daarnaast wil ik ook mijn examinatoren dhr. J.J. van Beijnum en dhr. W. van Vliet bedanken voor de begeleiding en feedback die ik van hen heb gekregen.

Noordwijkerhout, juni 2005

Inhoudsopgave

Externe bijlagen

1. Inleiding

Het doel van dit eindverslag is om lezers inzicht te geven in het doorlopen afstudeerproces. De keuzes die ik tijdens dit proces gemaakt heb, worden in dit verslag toegelicht. De afstudeeropdracht draaide om het verbeteren van Spider, een content management systeem voor websites dat is ontwikkeld door Marbel Software, zodat gebruikers zelfstandig de vormgeving en structuur van hun website aan kunnen passen. Om dit te bereiken heb ik 3 web-based editors ontwikkeld waarmee CSS en XML-documenten kunnen worden aangepast.
In hoofdstuk 2 beschrijf ik de organisatie waar de afstudeeropdracht is uitgevoerd. Vervolgens kunt u lezen hoe de opdracht tot stand is gekomen. In hoofdstuk 4 beschrijf ik hoe ik mijn aanpak heb bepaald bij het uitvoeren van de opdracht. Na het bepalen van mijn aanpak heb ik de uitgangssituatie in kaart gebracht en systeemeisen opgesteld waaraan de oplossing moest voldoen. Hierover kunt u lezen in hoofdstuk 5. De hoofdstukken 6 tot en met 8 gaan over de ontwikkeling van de editors. De afronding van de opdracht wordt beschreven in hoofdstuk 9. Tot slot evalueer ik in hoofdstuk 10 het proces dat ik doorlopen heb en de producten die ik heb opgeleverd.
Het verslag is in de eerste plaats geschreven voor mijn examinatoren en gecommitteerde. Er wordt van uitgegaan dat lezers van dit verslag enige kennis hebben van de technieken die in dit verslag aan bod komen. Mochten er in dit verslag toch termen voorkomen die enige verduidelijking vereisen, dan verwijs ik u graag naar de verklarende woordenlijst, achter in dit verslag.

2. Organisatie

In dit hoofdstuk geef ik een algemene beschrijving van de organisatie waar de afstudeeropdracht is uitgevoerd. Verder zal ik in het kort beschrijven wat voor soort projecten er zoal bij Marbel Software lopen. Tot slot kunt u in dit hoofdstuk lezen wat mijn plaats binnen de organisatie was tijdens het afstudeertraject.

2.1. Algemene beschrijving

Marbel is opgericht in november 1990 door Martin Weijers en Ronald de Koning. In 2000 is het bedrijf opgesplitst in de volgende twee bedrijven.

- Marbel Consult, dat zich bezig houdt met de advisering en het beheer op het gebied van ICT-infrastructuur en de beveiliging daarvan.
- Marbel Software, dat zich bezig houdt met het ontwikkelen van zowel standaard als maatwerk software.

Er werken nu ongeveer 13 mensen bij Marbel. Marbel Software heeft inmiddels een ruime ervaring op het gebied van softwareontwikkeling. De projectinrichting van het bedrijf verschilt per project. Soms wordt er op detacheringbasis gewerkt en andere projecten worden op het kantoor in Noordwijkerhout uitgevoerd.
Het bedrijf is gespecialiseerd in het ontwikkelen en implementeren van maatwerk (software) systemen. Deze systemen kunnen variëren van administratief tot technische (datacommunicatie) systemen. De systemen worden tegenwoordig veel ontwikkeld in C#, een nieuwe programmeertaal die speciaal is ontwikkeld voor Microsoft's .NET-framework.
In het 18-jarige bestaan van Marbel heeft het gewerkt aan uiteenlopende projecten bij verschillende bedrijven.

- Bij de Rabobank International in Utrecht voert Marbel verschillende projecten uit. Ten behoeve van actuele koersinformatie voor het bank- en verzekeringsbedrijf, is er in samenwerking met de Rabobank een systeem opgezet welke de realtime opslag en distributie van koersinformatie naar interne en externe computersystemen van een financiële instelling verzorgt.
- Voor EuroCross International in Noordwijk, een bedrijf dat zich bezig houdt met de coördinatie van alle nationale en internationale hulpverlening in opdracht van onder andere alle ACHMEA verzekeringsmaatschappijen, heeft Marbel Software bijna alle programmatuur verzorgd die de alarmcentrale in haar hulpverlening ondersteunen.
- Voor het Centraal Bureau Rijvaardigheidsbewijzen (CBR) heeft Marbel Software onder andere het theorie-examensysteem ontwikkeld.

2.2. Plaats van de afstudeerder

Het afstudeerproject heb ik uitgevoerd op het kantoor van Marbel Software in Noordwijkerhout. Op de werkvloer heerst een informele werksfeer. Programmeurs en projectleiders werken in hetzelfde kantoor. De communicatielijnen zijn daardoor zeer kort. Ik werkte in hetzelfde kantoor dus ik kon tijdens het project altijd om advies vragen. De werknemers zijn allemaal minimaal afgestudeerd op HBO-niveau.

3. Totstandkoming van de opdracht

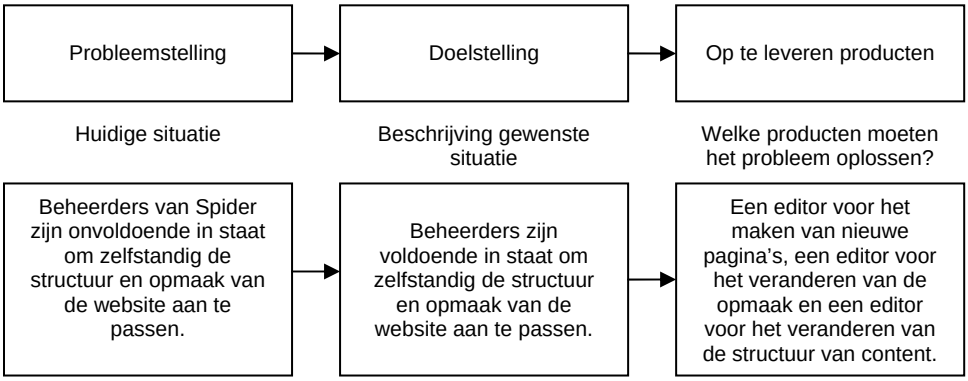
In dit hoofdstuk kunt u lezen hoe het proces van totstandkoming van de afstudeeropdracht is verlopen. Het hoofdstuk is verdeeld in een drietal paragrafen die elk een deel van het proces beschrijven. Voor het eindproduct van dit proces, de definitieve opdrachtomschrijving verwijs ik u naar bijlage A.

3.1. Aanleiding van de opdracht

Ik ben in aanraking met Marbel software gekomen toen ik voor het duaal-traject op zoek was naar een bedrijf. Via mijn vader ben ik in contact gekomen met Gert-Jan Scheeres, directeur van Marbel Consult. Bij Marbel Software zijn regelmatig afstudeerders werkzaam. Tijdens het duaal-traject werd duidelijk dat ik mijn afstudeeropdracht bij Marbel Software zou kunnen uitvoeren.
Samen met Gert-Jan Scheeres en andere werknemers heb ik toen de opdracht bepaald. Het project waar ik tijdens het afstudeertraject aan kon werken was Spider. Spider is een content management systeem (CMS) voor websites dat Marbel Software oorspronkelijk heeft ontwikkeld voor het Nederlands Dagblad (ND).
Met Spider is het mogelijk om redacteurs en moderators aan te wijzen die allen verantwoordelijk zijn voor hun eigen content. Verder kunnen bezoekers van de website reageren op artikelen en discussiëren over verschillende onderwerpen. Spider is volledig gebaseerd op XML, CSS en XSLT en is geschreven voor het ontwikkelplatform ASP.NET in de programmeertaal C#. Hoewel het systeem oorspronkelijk is ontwikkeld voor het Nederlands Dagblad wordt het nu ook ingezet bij andere bedrijven. Door de open structuur van het systeem kan het geheel aangepast worden aan de wensen van de klant.
Uit een usability-analyse die al eerder is gedaan, is gebleken dat gebruikers van Spider voldoende in staat zijn om eenvoudig artikelen toe te voegen. Op het gebied van beheer en het uitbreiden en aanpassen van de website bleek Spider de gebruiker echter onvoldoende te ondersteunen. Als afstudeeropdracht zou ik aan de slag kunnen gaan met het verbeteren daarvan.

3.2. Opstellen van de opdracht

Omdat ik al 3 maanden werkzaam was bij Marbel Software, heb ik ruim de tijd gehad om de opdrachtomschrijving van mijn afstudeerproject op te stellen samen met mijn bedrijfsmentor, Gert-Jan Scheeres. Vanuit de opleiding was Jaap van Beijnum als keurend docent aangewezen. In overleg met beiden is de uiteindelijke opdrachtomschrijving vastgesteld.
De kern van de opdrachtomschrijving zijn de onderdelen ‘probleemstelling’, ‘op te leveren producten’ en ‘doelstelling’. Vooral deze drie onderdelen moesten een aantal keren opnieuw worden geformuleerd zodat er een duidelijke relatie tussen deze zaken ontstond. In figuur 1breng ik deze relatie in beeld.



figuur 1: Relatie probleemstelling, doelstelling en op te leveren producten

Op de volgende pagina heb ik ter verduidelijking bovenstaande onderdelen uit de definitieve opdrachtomschrijving opgenomen.

Probleemstelling
Regelmatig komt het voor dat gebruikers (gebruikersgroep webmasters) van Spider naar Marbel Software bellen omdat ze de structuur en de opmaak van hun Website willen aanpassen. Zelf kunnen zij dit niet omdat ze niet overweg kunnen met de technieken XML en CSS. In Spider is het gebruik van deze technieken nodig om de structuur en de opmaak van een Website aan te passen.

Doelstelling
Het doel van de afstudeeropdracht is het verbeteren van Spider zodat gebruikers (gebruikersgroep webmasters) zelfstandig in staat zijn om de structuur en de opmaak van hun Website aan te passen zonder ervaring te hoeven hebben met de technieken XML en CSS.

Op te leveren producten
Om de doelstelling te verwezenlijken, zullen er GUI's worden ontwikkeld voor het ontwerpen en aanpassen van de opmaak en de structuur van de in Spider beheerde websites. De volgende producten zullen worden ontwikkeld, voortkomend uit de bovenstaande doelstelling:

- *Document-editor*
  - *De document-editor moet het mogelijk maken om Spider-documenten te genereren en te bewerken zonder dat de gebruiker kennis hoeft te hebben van XML.*
- *Template-editor*
  - *Met behulp van de template-editor moeten eenvoudig templates kunnen worden gegenereerd.*
- *CSS-editor*
  - *De CSS-editor wordt een editor voor CSS-sheets binnen Spider. Om opslag van CSS-sheets mogelijk te maken binnen het bestaande versiemanagement-systeem van Spider, wordt er een methode ontwikkeld om CSS op te slaan als XML.*

Naast de bovenstaande producten die vooral praktisch gericht zijn op het behalen van de doelstelling van de afstudeeropdracht, zullen er ook een aantal producten worden opgeleverd die vooral het proces van de opdracht ondersteunen. Hieronder deze categorie vallen de volgende producten:

- *Plan van Aanpak*
  - *In het plan van aanpak worden afspraken tussen de opdrachtgever en de opdrachtnemer opgenomen die ervoor moeten zorgen dat het proces goed verloopt. Ook wordt er in het plan van aanpak een tactiek gekozen voor het oplossen van het probleem zoals beschreven in paragraaf 3.2.*
- *Definitiestudie*
  - *Het rapport definitiestudie komt voor uit de in dit project gebruikte ontwikkelmethodiek, IAD. De belangrijkste onderdelen van de definitiestudie zijn het specificeren van de systeemeisen en het bepalen van het systeemconcept.*
- *Pilotontwikkelplannen*
  - *Per pilot (bouweenheden als onderdeel van het uiteindelijke resultaat) wordt een rapport opgesteld waarin een gedetailleerde specificatie van de te ontwikkelen pilot wordt gegeven. Ook deze producten worden gemaakt in het kader van de gebruikte ontwikkelmethodiek IAD.*
- *Testrapport*
  - *Als alle pilots zijn geïntegreerd en ingevoerd zal het eindresultaat worden getest. De resultaten van deze tests zullen worden beschreven in het testrapport.*

figuur 2: Kern opdrachtomschrijving (deel van de definitieve opdrachtomschrijving – Bijlage A)

In de eerste versie van de opdrachtomschrijving waren de te verrichten activiteiten zo geformuleerd dat er een 'universele' editor zou worden ontwikkeld voor gestructureerde content op basis van XML. Bij deze omschrijving werd te weinig benadrukt dat het om het verbeteren van Spider ging terwijl dat juist de bedoeling was. Daarom heb ik de opdrachtomschrijving zo aangepast dat in het gedeelte 'op te leveren producten' duidelijk de onderdelen van Spider, die ontwikkeld moesten worden, terugkwamen.

Verder heb ik het onderdeel ‘te verrichten activiteiten’ concreter gemaakt door de verschillende IAD-activiteiten meer te betrekken op mijn afstudeeropdracht.
De keuze voor IAD tijdens het opstellen van de opdracht was geen definitieve keuze. Deze werd gemaakt tijdens het opstellen van het plan van aanpak. Hierover kunt u meer lezen in het volgende hoofdstuk.
In week 9 van de afstudeerperiode vond het eerste gesprek plaats met mijn examinatoren, bedrijfsmentor en mijzelf. In dit gesprek is ondermeer de definitieve opdracht vastgesteld. De enige wijziging in mijn definitieve opdrachtomschrijving ten opzichte van de voorlopige was het schrappen van één van de op te leveren producten. De style-editor (XSL-T) kwam te vervallen na aanbevelingen van een collega. Dit omdat het volgens hem een bijna onmogelijke opgave zou zijn om een gebruikersinterface te ontwikkelen voor het bewerken van XSL-T documenten die door een beginnende gebruiker gebruikt kan worden en toch de mogelijkheid biedt tot het toevoegen van alle complexe functionaliteit die XSL-T te bieden heeft.

4. Bepalen van de aanpak

In het vorige hoofdstuk heeft u kunnen lezen hoe de definitieve opdrachtomschrijving tot stand is gekomen. De volgende stap in het proces was het opstellen van een plan van aanpak. Daarover kunt meer lezen in dit hoofdstuk. De belangrijkste keuzes die hierbij moesten worden gemaakt, worden toegelicht in dit hoofdstuk. Het volledige plan van aanpak is te vinden als externe bijlage (Bijlage B)

4.1. Definitieve keuze ontwikkelmethodiek

De keuze van een methode voor systeemontwikkeling was een belangrijke keuze die gemaakt moest worden bij het opstellen het plan van aanpak omdat de ontwikkelmethodiek als een ruggengraat moet dienen tijdens een ontwikkeltraject. In het vorige hoofdstuk heeft u kunnen lezen dat voor IAD gekozen is als voorlopige systeemmethodiek. Deze keuze is tijdens het opstellen van het plan van aanpak heroverwogen en hierover kunt u meer lezen in deze paragraaf.

Omdat het hier gaat om een ontwikkelingstraject dat in een relatief kort tijdbestek (18 weken) moest worden uitgevoerd en het voor de opdrachtgever belangrijk was dat er een compleet systeem opgeleverd zou worden, was de keuze voor een RAD-methode voor de hand liggend. Waar ik vooral benieuwd naar was, was of er een methode bestond die speciaal bedoeld was voor kleine trajecten waarbij geen compleet informatiesysteem ontwikkeld wordt.
Tijdens het zoeken hiernaar stuitte ik op een publicatie op internet waarin vier veelgebruikte methodes voor systeemontwikkeling tegen elkaar uit worden gezet[1]. De belangrijkste verschillen per methode heb ik geprobeerd om naast elkaar te zetten in de volgende tabel.

4 moderne ontwikkelmethodes vergeleken				
naam	**IAD**	**DSDM**	**RUP**	**XP**
betekenis	Iterative/Interactive/Incremental Application Development	Dynamic Systems Development Method	Relational Unified Process	eXtreme Programming
gebruikers-participatie	Optimaal	goed	Verantwoordelijk voor het opstellen van de usecase-diagrammen	goed
iteratief	ja	ja	Ja	ja
incrementeel	ja	ja	ja	ja
fasering	• definitiestudie • pilotontwikkeling • invoering	• haalbaarheids-onderzoek • bedrijfsonderzoek • ontwerp • bouw • implementatie	• aanvang • detaillering • constructie • transitie	• ontdekkingsfase • planningfase • ontwikkeling • oplevering • onderhoud
samenwerking	workshops, u-team, a-team	workshops	doet weinig uitspraken over de wijze waarop moet worden samengewerkt.	pair-programming, programmeren in tweetallen
prototyping	in alle fasen	evolutionair, prototypes leiden tot uiteindelijk resultaat	geen vast onderdeel	methode is product-gedreven, goed inzetbaar
testen	tijdens beoordeling- en testworkshop	integratie- en regressietests door zowel programmeurs als gebruikers	neemt toe gedurende het ontwikkelproces	testen is een van de bouwstenen van de methode
op te leveren documentatie	zelf te bepalen	gericht op kwaliteitscriteria	volgens verschillende modellen, niet eenduidig	zwaartepunt ligt op broncode, niet op documentatie

figuur 3: Vergelijking 4 moderne ontwikkelmethodes

De categorieën in de linker kolom zijn de verschillende aandachtspunten waarop gelet werd in het artikel. Opvallender dan de verschillen vind ik de overeenkomsten tussen de methodes. Alle bovenstaande methodes laten het toe om tijdens het ontwikkelproces de systeemeisen bij te stellen. Verder speelt gebruikersparticipatie een belangrijke rol bij deze methodes.

[1] http://www.serc.nl/resources/publicaties/artikelen/InformatieOntwikkelmodellen.pdf

Een andere optie zou zijn geweest om de standaardaanpak van Marbel Software te hanteren. Deze aanpak houdt in dat er eerst een functioneel ontwerp wordt gemaakt. Hierin staan de systeemeisen beschreven. Nadat het functioneel ontwerp is goedgekeurd door de opdrachtgever wordt er een technisch ontwerp gemaakt dat eveneens goedgekeurd dient te worden door de opdrachtgever. Daarna vindt de ontwikkeling van het systeem plaats en volgt er een test- en acceptatieperiode. Het systeem kan tijdens deze periode nog worden aangepast. Daarna wordt het systeem ingevoerd en wordt er ondersteuning geleverd.
De hierboven beschreven methode wordt alleen toegepast bij de ontwikkeling van maatwerk software. Hiervan is bij mijn afstudeeropdracht echter geen sprake. Uiteindelijk heb ik besloten om de methode IAD definitief te gaan gebruiken met als doorslaggevende redenen:

- Ik heb IAD al diverse malen heb toegepast bij verschillende ontwikkeltrajecten op school. Hierdoor hoef ik geen tijd de besteden aan het me 'eigen' maken van de methode en blijft er veel tijd over voor de daadwerkelijke ontwikkeling.
- IAD biedt genoeg vrijheid om haar aan te passen aan de eigenschappen van dit ontwikkeltraject.

Tijdens dit project heb ik geen gebruik gemaakt van workshops omdat de opdrachtgever geen onderdeel is van de organisatie waarin het product gebruikt wordt. Om er toch voor te zorgen dat er rekening wordt gehouden met de wensen van de eindgebruikers heb ik interviews bij de eindgebruikers afgenomen. Hierover kunt u meer lezen in het volgende hoofdstuk.

4.2. Keuze iteratiestrategie

Een belangrijke keuze die gemaakt moet worden bij het gebruik van IAD als systeemontwikkeling-methodiek is de te gebruiken iteratiestrategie. Deze strategie bepaalt op wat voor manier de verschillende IAD-fasen doorlopen worden.

figuur 4: Doorlopen van de IAD-fasen bij iteratiestrategie 'Incrementeel ontwikkelen'.

IAD hanteert bij het kiezen voor een iteratiestrategie een aantal voorwaarden per strategie. Ik heb gekozen voor de iteratiestrategie 'incrementeel ontwikkelen' omdat de gestelde voorwaarden bij die strategie overeenkwamen met de situatie van mijn afstudeerproject. Dit waren de volgende voorwaarden.

- Het systeem kan goed van tevoren worden gespecificeerd.
- De omvang van het systeem is te overzien.
- Stapsgewijze invoering van het systeem is niet mogelijk (opdrachtgever wil direct resultaat).

4.3. Opstellen planning

Een ander belangrijk onderdeel binnen het plan van aanpak is het vaststellen van de te verrichten activiteiten en de daarbijbehorende planning. Dit is belangrijk omdat een goede planning ervoor kan zorgen dat het gehele systeem ontwikkeld wordt. Hiervoor moeten dan wel prioriteiten aan de verschillende activiteiten worden toegekend. Eerst worden de activiteiten met de hoogste prioriteit uitgevoerd. Als dat gedaan is, staat er in principe al een systeem dat gebruikt kan worden. Indien er genoeg tijd over is kunnen dan de activiteiten met mindere prioriteit worden uitgevoerd. De activiteiten zijn ingedeeld in drie prioriteiten: basis, comfort en luxe. Deze planningstechniek wordt timeboxing genoemd en wordt veel gebruikt binnen IAD.
Eerst heb ik een opsomming gemaakt van alle activiteiten die verricht moesten worden in de verschillende fasen van IAD. Daarna heb ik een planningssheet ontworpen waarbij tot in detail (op dagniveau) gepland kan worden. Veel activiteiten in de fase pilotontwikkeling die samenhangen met

het ontwikkelen van de softwarebouweenheden zijn bij het opstellen van de planning opengelaten omdat de precieze pilot-indeling en de daarbij behorende activiteiten toen nog niet bekend waren. De planning is aangepast na het opstellen van het pilotplan. Voor het volledige planningssheet verwijs ik u naar bijlage C.

4.4. Kwaliteitsborging

Het laatste onderdeel van het plan van aanpak waarvan ik de totstandkoming in dit hoofdstuk wil bespreken is het onderdeel 'kwaliteitsborging'. IAD verstaat onder kwaliteitsborging het geheel van geplande en systematische activiteiten dat nodig is om in voldoende mate het vertrouwen te geven dat een resultaat voldoet aan de gestelde kwaliteitseisen.
Per product heb ik steeds bekeken wat belangrijke eisen zijn en vervolgens heb ik gekeken hoe ik ervoor kan zorgen dat er altijd aan deze eisen voldaan wordt. De uitkomsten hiervan worden weergegeven in de volgende tabel.

Product	Eis	Hoe te bereiken
Documenten	De opgeleverde documenten moeten duidelijk en volledig zijn.	Voordat de documenten worden opgeleverd, worden ze door een buitenstaander gelezen.
	De opgeleverde documenten moeten consistent zijn.	De standaarden uit het plan van aanpak gebruiken.
Ontwikkelde software	Bestaande gebruikers moeten zelfstandig overweg kunnen met nieuwe functionaliteit zonder eerst een handleiding te hoeven lezen.	Het zorgen voor grafische interfaces zodat gebruikers zich niet meer bezig hoeven te houden met codes.
	De code die geschreven gaat worden om de nieuwe functionaliteit te bereiken, dient correct te zijn en moet aansluiten bij de bestaande architectuur.	Voordat er ontwikkeld gaat worden, dient er overleg plaats te vinden met de ontwikkelaars om afspraken te maken over de applicatiearchitectuur.
	De code die geschreven gaat worden dient consistent te zijn.	De code dient zoveel mogelijk geschreven te worden volgens de Microsoft Internal Coding Guidelines[2].

figuur 5: Kwaliteitseisen uit plan van aanpak

De eerste twee eisen over de kwaliteit die gesteld wordt aan documenten heb ik hier opgenomen om mijzelf formeel te verplichten om duidelijk, volledige, en consistente documenten op te leveren. De derde eis komt overeen met een van de belangrijkste doelstellingen van de opdracht. Ik heb de eis hier opgenomen om ervoor te zorgen dat ik dit gedurende het hele project niet uit het oog verloor. De vierde eis heb ik opgenomen om ervoor te zorgen dat als ik klaar was met de afstudeeropdracht, de ontwikkelaars van Spider de nieuwe functionaliteit goed zouden kunnen onderhouden.
De laatste eis over de consistentie van de code is opgenomen op aanbeveling van een collega. De Microsoft Internal Coding Guidelines omvatten een geheel van afspraken over de naamgeving van objecten bij het schrijven van programmacode. Door me hier aan te houden, kunnen andere programmeurs eenvoudig verder ontwikkelen aan mijn programmacode.

[2] Deze richtlijnen zijn te lezen op http://blogs.msdn.com/brada/articles/361363.aspx

5. Opstellen van de definitiestudie

In het vorige hoofdstuk heeft u kunnen lezen hoe het plan van aanpak tot stand is gekomen en dat er definitief gekozen is voor IAD als ontwikkelmethodiek. De definitiestudie is de eerste fase binnen de IAD-ontwikkelcyclus. In dit hoofdstuk wordt het proces dat in deze fase is doorlopen, toegelicht. Allereerst zal ik beschrijven hoe het ontwikkelscenario werd vastgesteld. In de tweede paragraaf kunt u lezen hoe ik de uitgangssituatie bepaald heb. Op basis daarvan zijn de systeemeisen opgesteld. Hierover kunt u lezen in de derde paragraaf. Vervolgens wordt het in kaart brengen van de gewenste situatie beschreven in paragraaf 4. In de paragraaf daarna beschrijf ik hoe ik de technische structuur in kaart heb gebracht en tot slot kunt u lezen hoe ik het pilotplan heb opgesteld. Voor het volledige ‘rapport definitiestudie’ verwijs ik u naar bijlage D.

5.1. Bepalen van het ontwikkelscenario

Het belangrijkste doel van deze activiteit is volgens IAD het vaststellen van de reikwijdte en de context van het ontwikkelproject. De belangrijkste onderdelen van deze activiteit zal ik toelichten in de volgende paragrafen.

5.1.1. Doelstellingen en reikwijdte van het project

De doelstellingen van het project zijn al eerder vastgesteld in de opdrachtomschrijving en het plan van aanpak. De reikwijdte bepalen, betekent dat er vermeld moet worden wat er wel en wat er niet ontwikkeld gaan worden. Dit heb ik gedaan door duidelijk te vermelden aan welke programmaonderdelen er ontwikkeld ging worden, namelijk de documenteditor, de css-editor en de template-editor. Als deze programmaonderdelen ontwikkeld waren, zou de doelstelling zijn volbracht. Immers, Spider is dan in die mate verbeterd dat gebruikers zelfstandig in staat zijn om de opmaak en de structuur van hun website aan te passen.

5.1.2. Technische structuur

Normaal gesproken wordt in het ontwikkelscenario al een inschatting gemaakt van de technische structuur waarin ontwikkeld gaat worden. Ik heb besloten om dit op een later moment te doen, namelijk na het beschrijven van de gewenste situatie. Ik heb hiervoor gekozen omdat de technische structuur voor een groot deel bepaald wordt in een interview met de ontwikkelaars. Dit interview stond gepland binnen de activiteit ‘systeemconcept bepalen’.

5.1.3. Cruciale succesfactoren

In het onderdeel ‘cruciale succesfactoren’ moet er factoren genoteerd worden die van belang zijn voor een goed afloop van het project. Hierbij denk ik meteen aan eerdere projecten die ik gedaan heb. Bij eerdere projecten kwam ik wel eens in de problemen doordat de planning die ik maakte te krap was. Bij een te krappe planning kan je in de problemen komen doordat belangrijke geplande activiteiten niet meer uitgevoerd kunnen worden omdat andere activiteiten teveel tijd kosten. Het hanteren van een goede planning en het handhaven ervan heb ik dan ook opgenomen als cruciale succesfactor.
Een andere factor die ik heb opgenomen is ‘uitbreiding van de technische kennis van de afstudeerder’. De reden hiervoor is dat ik bij aanvang van de afstudeeropdracht een geringe programmeerervaring had en de afstudeeropdracht voor een groot deel programmeren was. Op zich hoefde dit natuurlijk geen probleem te zijn; eerder een uitdaging. Een cruciale succesfactor was het in ieder geval.

5.1.4. Kenmerken van het ontwikkelproces

Binnen dit onderdeel van het ontwikkelscenario moest worden vastgesteld hoe belangrijke aspecten van IAD ingevuld zouden gaan worden. Per aspect heb steeds bekeken hoe ik daar invulling aan kon geven tijdens het ontwikkeltraject. Ik zal hieronder de verschillende aspecten bespreken.

- Iteratie
  - In het plan van aanpak heb ik al gekozen voor de iteratiestrategie 'incrementeel ontwikkelen'. In de eerste versie van de definitiestudie dacht ik nog dat er in de praktijk niet echt sprake zou zijn van iteratieslagen omdat er geen sprake was van bijkomende functionaliteit. Ik ging er toen vanuit dat de fase pilotontwikkeling één maal doorlopen zou worden, waarna alle functionaliteit aanwezig zou zijn. In de tweede versie van de definitiestudie waarin het pilotplan was aangepast, bleek daar wel zeker sprake van te zijn. In de eerste iteratieslag wordt alle basis-functionaliteit ontwikkeld, in de tweede iteratieslag de comfort-functionaliteit, etc.
- Workshops
  - Een ander aspect dat ook al in het plan van aanpak besproken is, is het gebruik van workshops gedurende het ontwikkeltraject. Zoals eerder vermeld zullen er interviews worden afgenomen om ervoor te zorgen dat de wensen van de gebruikers mee worden genomen in het uiteindelijke systeem.
- Teams
  - Normaal gesproken wordt er bij IAD gewerkt in teams. De verschillende gebruikersgroepen worden dan ondergebracht in U-teams en de ontwikkelaars in A-teams. Omdat bij dit project niet uitvoerig wordt samengewerkt met de gebruikersorganisatie zal er geen gebruik worden gemaakt van teams.
- Time-boxing
  - Time-boxing is een projectmanagementinstrument dat bij IAD veel gebruikt wordt. Bij time-boxing moeten doelen bereikt worden binnen een van te voren vastgestelde datum. Om dit te bereiken worden de eisen geprioriteerd. In de vorige paragraaf 'cruciale succesfactoren' heb ik al besproken dat een goede planning van belang was voor een goede afloop van het project. Daarom heb ik besloten om dit instrument in te zetten voor dit project. In de laatste paragraaf van dit hoofdstuk, waarin wordt besproken hoe het pilotplan tot stand is gekomen, zal ik meer vertellen over hoe ik time-boxing heb toegepast tijdens dit project.
- Parallellisme
  - Parallellisme wil zeggen dat meerdere pilots of delen daarvan gedeeltelijk tegelijkertijd ontwikkeld worden. Bij het opstellen van de definitiestudie was ik aanvankelijk niet van plan om deze techniek in te zetten tijdens het ontwikkeltraject. Tijdens de fase pilotontwikkeling kwam ik erachter dat ik deze techniek min of meer onbewust toch inzette. De help-functionaliteit binnen de verschillende pilots is voor een groot deel parallel ontwikkeld. Meer hierover kunt u lezen in de hoofdstukken 6, 7 en 8.

5.1.5. Eerste indicatie pilot-strategie

Binnen dit onderdeel van het bepalen van het ontwikkelscenario wordt een eerste indicatie gegeven van hoe de pilot-indeling eruit komt te zien en hoe deze ontwikkeld gaan worden. In mijn eerste indicatie ging ik er nog van uit dat de volgende 5 pilots ontwikkeld zouden gaan worden.

- GUI voor de Document-editor
- GUI voor de Template-editor
- GUI voor de Style-editor
- GUI voor de CSS-editor
- Overige aanpassingen in Spider

Later heb ik dit bijgesteld naar drie pilots waarbij de GUI voor de Style-editor en de overige aanpassingen in Spider kwamen te vervallen. Ik heb hiertoe besloten omdat de opdracht anders te

breed zou worden. De kans zou dan bestaan dat ik niet alle pilots binnen de afstudeerperiode zou kunnen ontwikkelen.

5.1.6. Test- en acceptatiestrategie

Het bepalen van de test- en acceptatiestrategie was belangrijk omdat deze strategieën bepalen of het uiteindelijke resultaat van het project als succesvol mogen worden beschouwd. Ik heb ervoor gekozen om het grootste gedeelte van het testproces te laten verlopen tijdens de ontwikkeling. Hiervoor had ik twee redenen. Allereerst had ik weinig ervaring met programmeren en was ik tijdens de ontwikkeling niet altijd geheel zeker van het resultaat van mijn werk. Om te verifiëren of alle programmaonderdelen goed waren ontwikkeld, heb ik al deze onderdelen steeds getest. Ik ging dan pas verder met ontwikkelen als het voorgaande onderdeel werkte. Een andere reden waarom ik voor het grootste gedeelte wilde testen tijdens de ontwikkeling is dat grafische interfaces zich hier bij uitstek voor lenen. Elke vorm van gedrag van de GUI valt direct terug te leiden tot de actie die de gebruiker heeft uitgevoerd op de verschillende schermelementen in die GUI. Een ander argument om te kiezen voor deze teststrategie is dat de globale werking van Spider al in een eerder stadium uitgebreid getest is. Het testen van programmaonderdelen op een zo klein mogelijk niveau door programmeurs zelf wordt whitebox-testen genoemd. Verder heb ik besloten om behalve deze whitebox-tests ook nog te testen of het systeem voldoet aan de systeemeisen. Hierbij zal per systeemeis voor zover mogelijk worden geverifieerd of het systeem daaraan voldoet. Deze vorm van testen waarvoor in principe geen kennis van de achterliggende architectuur vereist is, wordt blackbox-testen genoemd.
Omdat de invoering van het systeem in één keer plaats vind, heb ik ervoor gekozen om ook de acceptatie in één keer per pilot plaats te laten vinden. Omdat Marbel Software de opdrachtgever is, zal het bedrijf ook de acceptatietest uitvoeren. Ik heb de volgende criteria opgesteld waarop gelet moest worden bij de acceptatie van het systeem.

- Komt de functionaliteit van de pilot overeen met de systeemeisen? (per pilot)
- Is voldaan aan de integriteits- en interface-eisen? (per pilot)
- Is de systeemdocumentatie op orde? (per pilot)
- Kan het systeem worden beheerd door Marbel Software?

Als deze vragen door de opdrachtgever positief beantwoord worden, zou het systeem kunnen worden ingevoerd.

5.2. Bepalen van de huidige situatie

In de vorige paragraaf heeft u kunnen lezen hoe ik als eerste activiteit binnen de definitiestudie het ontwikkelscenario heb bepaald. In deze paragraaf beschrijf ik hoe de tweede activiteit, het beschrijven van de huidige situatie, heb uitgevoerd. Het doel van deze activiteit was voor mij om de gebruikers en hun problemen en wensen in kaart te brengen. Dit diende voor mij als uitgangspunt om de systeemeisen op te stellen.

5.2.1. Gebruikersgroepen

Binnen Spider bestaat een voorziening om te werken met verschillende gebruikersgroepen. Aan elke gebruikersgroep kunnen specifieke rechten toegekend worden. Standaard zijn de volgende zeven gebruikersgroepen gedefinieerd:

- **Redacteurs**
- Hoofdredacteurs
- Marketing medewerkers
- **Moderators**
- Opmaak redacteurs
- **Site beheerders**
- Winkel beheerders

De vetgedrukte gebruikersgroepen zijn momenteel het belangrijkst. De overige groepen hebben allemaal overlap qua bevoegdheden of worden (nog) niet gebruikt in Spider en zullen hier niet besproken worden.
Nadat ik de taken van elke gebruikersgroep had genoteerd, kon ik een beschrijving per gebruikersgroep maken.

Gebruikersgroepen binnen Spider	
Redacteurs	Een redacteur schrijft artikelen die door een moderator geplaatst kunnen worden op de website. Bij het uitvoeren van zijn taak hoeft de redacteur geen specifieke technische kennis te hebben van HTML. Voor het toevoegen of wijzigen van artikelen kan een ingebouwde HTML-editor worden gebruikt.
Moderators	Een moderator keurt artikelen van redacteurs en/of reacties van bezoekers en beslist of een artikel of reactie geplaatst wordt op de website. Zonodig kan een moderator het artikel nog aanpassen voordat hij het plaatst. De moderatiefunctie kan ook worden uitgeschakeld.
Site beheerders	De beheerder van de website heeft alle rechten die de andere gebruikers hebben. Daarnaast kan de beheerder de vormgeving en/of structuur aanpassen en nieuwe pagina's toevoegen. Het aanpassen van de opmaak van de website gebeurt in de CSS-gedeelte. De beheerder dient hiervoor wel de nodige kennis van CSS te hebben omdat hij de gehele stylesheet zelf in moet typen. Het toevoegen van een nieuwe pagina met nieuwe functionaliteit zijn ook taken die onder de verantwoordelijkheid van de beheerder vallen. Op dit moment is het bewerken van een documentdefinitie of een contenttemplate echter nog een complex proces, aangezien de beheerder zelf XML-codes in moet voeren. Daarom schakelt een beheerder van een website nu meestal Marbel Software in als hier wijzigingen in moeten worden aangebracht.

figuur 6: De drie belangrijkste gebruikersgroepen binnen Spider

5.2.2. Persona's

In de eerste versie van de definitiestudie had ik in het hoofdstuk 'beschrijving huidige situatie' een gedeelte opgenomen over persona's als onderdeel van een doelgroepanalyse. Het maken van persona's is een hulpmiddel dat de auteur en software-ontwerper Alan Cooper heeft ontwikkeld op basis van een oude techniek uit de reclamewereld. Om hun producten beter aan de man te kunnen brengen gebruikten marketingmensen de demografische gegevens die ze hadden over hun doelgroep. Daarmee probeerden ze een bepaald stereotype neer te zetten die stond voor de doelgroep die ze wilden bereiken. Ze bouwden op die manier een identificatie op met hun doelgroep, waardoor ze beter konden zien wat aan zou sluiten bij de verwachtingen van de persoon die ze wilden bereiken.

Aanvankelijk leken de persona's me een goede techniek om rekening te kunnen houden met de doelgroep, vooral omdat er tijdens dit ontwikkeltraject geen workshops zijn waarbij de eindgebruiker betrokken wordt bij de totstandkoming van het eindresultaat. Hieronder één van de persona's die ik heb opgesteld.

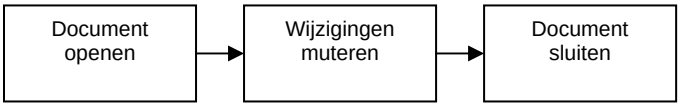
Henk van Lierop

Henk (39) is sinds 2 jaar eindredacteur van de internetredactie van een groot landelijk dagblad. Henk heeft een goede baan want sinds dat hij deze functie heeft, kon zijn vrouw Lia minder gaan werken om meer tijd met de kinderen door te brengen. Voorheen werkte Henk als freelance publicist. Hij verkocht zijn artikelen aan allerlei opiniebladen. Omdat Henk steeds vaker onderzoek moest verrichten voor de artikelen die hij schreef, raakte hij in aanraking met internet. Hij gebruikte internet toen vooral voor het zoeken naar specifieke informatie en het versturen van email. Omdat Henk meer zekerheid wilde, heeft hij gesolliciteerd naar een vaste aanstelling bij het dagblad waar hij nu werkt. Zelf schrijft Henk nu niet zoveel meer. Hij heeft veel meer een coördinerende rol en houdt zich bezig met het controleren van de artikelen van de redacteurs. Henk is van mening dat een goede krant meegaat met zijn tijd. Zowel in vorm als in inhoud. Ook vindt hij dat je lezers alleen kan binden als je vernieuwend blijft. Daarom experimenteert Henk regelmatig met de lay-out van de digitale krant.

Uiteindelijk heb ik besloten om het gedeelte van de persona's in een tweede versie van de definitiestudie te schrappen omdat in de praktijk bleek dat de techniek niets toevoegde aan het ontwerpproces. Dit kwam waarschijnlijk omdat ik bij het ontwikkelen van de GUI's meer rekening hield met reguliere GUI-designregels dan met de persona's. De persona's zijn niet echt tot leven gekomen in mijn hoofd. Ik heb toen besloten om de nadruk meer te leggen op de interviews om er toch voor te zorgen dat de wensen van zowel de gebruiker als ontwikkelaars terugkwamen in de te ontwikkelen oplossing.

5.2.3. Opstellen taakdiagrammen

Als onderdeel van de definitiestudie heb ik de huidige situatie onder andere in kaart proberen te brengen door middel van het opstellen van taakdiagrammen. Ik heb per programmaonderdeel (Redactionele gedeelte en beheergedeelte) taakdiagrammen opgesteld om te verifiëren of de knelpunten inderdaad het meest in het beheergedeelte voorkwamen. De taakdiagrammen heb ik opgesteld door zelf steeds stap voor stap te noteren welke acties ik uit moest voeren om een specifieke taak gedaan te krijgen.
Achteraf bleken deze taakdiagrammen weinig waarde toe te voegen omdat bij alle programma-onderdelen van het beheergedeelte in de oude situatie sprake was van een groot tekstvlak waarop de gebruiker tekst kon muteren. Hoewel de functie van deze tekstvlakken in de verschillende programmaonderdelen totaal verschillend waren, kwamen de handelingen voor het grootste deel met elkaar overeen. Zo kon eigenlijk voor elk programmaonderdeel in het beheergedeelte het volgende globale diagram worden aangehouden.



figuur 7: Globaal proces beheergedeelten

In het figuur hierboven valt op dat de eerste en de laatste stap behoorlijk duidelijk zijn. Bij het uitvoeren van deze stappen kan concreet worden verteld op welke knoppen gedrukt moest worden. De middelste stap laat echter een hoop vragen open: Wat voor wijzigingen moeten er bijvoorbeeld gemuteerd worden? En hoe moet dit dan gebeuren?
Hieruit heb ik de conclusie getrokken dat de verschillende onderdelen van het beheergedeelte de gebruiker niet genoeg ondersteunden zodat ze er zelfstandig mee zouden kunnen werken. Achteraf concludeer ik dat de hele stap van het maken van taakdiagrammen een erg omslachtig middel is voor het in kaart brengen van een relatief eenvoudig probleem.

5.2.4. Interviews

Omdat ik bij dit ontwikkeltraject geen gebruik maak van workshops om de wensen van de gebruiker te achterhalen, heb ik zoals eerder beschreven interviews bij de gebruikers van Spider afgenomen. Daarbij waren er een aantal aandachtspunten waar ik in geïnteresseerd was:

- Dagelijkse werkzaamheden van de gebruiker. Hierdoor kon ik inschatten wat voor soort computergebruiker de geïnterviewde was.
- De onderdelen waar ze Spider voor gebruiken. Hierdoor kon ik bepalen of ze een realistisch oordeel konden hebben over een bepaald programmaonderdeel. Iemand die alleen maar gebruik maakt van het redactionele gedeelte van Spider kan weinig zeggen over het beheergedeelte.
- De onderdelen die voor de gebruikers duidelijk zijn. Hierdoor weet je aan welke programmaonderdelen je vooral niet hoeft te sleutelen.
- De onderdelen die voor de gebruikers onduidelijk zijn. Dit zijn de onderdelen die moeten worden verbeterd.
- Overige ideeën van de gebruikers. Door de gebruikers zelf aan het woord te laten, kunnen er interessante ideeën aan het licht komen.

Op basis van bovenstaande aandachtspunten heb ik een lijst met vragen opgesteld waarbij ik erop gelet heb dat de vragen zoveel mogelijk open zijn, zodat de gebruiker uitgebreid zijn mening kwijt kan. Ik heb in de vragenlijst voldoende ruimte ‘opengelaten’ zodat de gebruiker al zijn opmerkingen kwijt kon. De vragenlijst heb ik verstuurd naar de volgende gebruikers:

- Linda Stelma, redacteur voor NDLite. (gebruikersgroep redacteurs)
- Wessel van Soest, directeur van het adviesbureau Argentum. (gebruikersgroep redacteurs/moderators/site-beheerders)
- Peter Brink, webmaster van het Nederlands Dagblad en NDLite (gebruikersgroep site-beheerders)

Van twee gebruikers heb ik de ingevulde vragenlijst teruggekregen. De uitkomst vond ik niet bepaald verassend. Hieronder een uitspraak van een gebruiker die goed illustreert dat vooral het uitvoeren van beheertaken door gebruikers als moeilijk wordt ervaren.

“Het doet voor mij wat het moet doen, maar ik vind de opbouw en structuur niet helemaal logisch. Als je het niet dagelijks gebruikt dan is het wel een zoektocht. Daarnaast ontbreekt een duidelijke handleiding. Het aanpassen van een stuk tekst is niet moeilijk. Wel het aanpassen van lettertype e.d. Het toevoegen van een pagina is lastiger en daarvoor ontbreekt een logisch stappenplan.”

Om deze en andere uitspraken te verifiëren en om meer te weten te komen van de technische structuur van Spider heb ik ook een gesprek met de ontwikkelaars van Spider gehad, waaruit naar voren kwam dat de meeste vragen van gebruikers gerelateerd waren aan het niet weten hoe bepaalde taken binnen Spider kunnen worden uitgevoerd. Op de vraag voor wie beheerfunctionaliteit zoals het aanpassen van de opmaak is bedoeld antwoordde Michael, een van de ontwikkelaars van Spider:

“Het is bedoeld voor de site administrators maar in veel gevallen kunnen ze er niet mee uit de voeten. Het zijn dingen die wij dan voor ze implementeren en daar genereren we dan ook uren voor. Maar als je bijvoorbeeld kijkt naar een CSS-editor waar we het al wel eens over gehad hebben, het zou mooi zijn als je via zo’n editor wat van de functionaliteit terug kan geven aan de gebruiker.
Bij veel vragen die we krijgen gaat het om hele eenvoudige taken waar wij eigenlijk helemaal geen trek in hebben zoals een documentje toevoegen, kleuren aanpassen en dat soort dingen. Als daar een oplossing voor wordt gemaakt, zou dat ons erg ontlasten.”

Ook deze uitspraak bevestigde mijn veronderstelling dat vooral het beheergedeelte niet duidelijk was voor de gebruiker. De complete vragenlijst met bijbehorende antwoorden van de gebruikers, alsmede het interview dat ik bij de ontwikkelaars van Spider heb afgenomen, is te lezen in paragraaf 3.4 en 3.5 van het rapport definitiestudie. (Bijlage D)

5.3. Opstellen van de systeemeisen

Een activiteit die centraal staat binnen de definitiestudie is het opstellen van de systeemeisen. De activiteit is belangrijk omdat in de systeemeisen in principe alle functionaliteit wordt beschreven die het eindproduct moet gaan bevatten op globaal niveau.
Een probleem waar ik tegenaan liep toen ik de systeemeisen ging opstellen was dat ik bij eerdere projecten IAD heb gebruikt als ontwikkelmethodiek voor een nieuw systeem. Bij dit project ging het om het verbeteren van een bestaand product. Het gevolg hiervan is dat ik zelf geen invloed kon uitoefenen op het bestaande systeem. Natuurlijk was het wel belangrijk om te kunnen formuleren aan welke eisen het eindresultaat van mijn opdracht moet voldoen. Daarom heb net als bij een normaal systeemontwikkelingtraject de systeemeisen opgesteld. De eisen hebben voor zover mogelijk alleen betrekking op datgene wat ik ga ontwikkelen. Zoals eerder vermeld gaat het bij dit project om het verbeteren van het gebruikersgemak van Spider. De eisen hebben dus geen betrekking op de technische structuur van Spider die al ontwikkeld is maar op de wijzigingen die zullen worden aangebracht aan bepaalde onderdelen binnen Spider. Eerst heb ik eisen opgesteld per categorie. De categorieën waren:

- Basissysteemeisen (BS)
- Interface-eisen (IF)
- Integriteitseisen (IG)
- Performance-eisen (PF)
- Operationele eisen (OP)

De systeemeisen zijn tot stand gekomen met behulp van de interviews die ik af heb genomen bij de gebruikers en ontwikkelaars. Verder hebben ook de taakdiagrammen tot inzichten geleid die hebben geholpen met het opstellen van de systeemeisen. Een voorbeeld hiervan:

BS.01 Gebruikers (gebruikersgroep site-beheerders) moeten wijzingen kunnen aanbrengen in de opmaak van een website zonder dat ze kennis hoeven te hebben van CSS.

Bovenstaande eis is tot stand gekomen nadat uit de taakdiagrammen was gebleken dat gebruikers onvoldoende ondersteuning krijgen bij het veranderen van de opmaak. Andere systeemeisen zijn tot stand gekomen doormiddel van bestaande design-regels. Ook hiervan een voorbeeld:

IF.04 De gebruikersinterfaces moeten worden uitgerust met ingebouwde helpfunctionaliteit.
IF.05 Ervaren gebruikers moeten de mogelijkheid hebben om de gebruikersinterfaces uit te kunnen schakelen.

Bovenstaande interface-eisen zijn afgeleid van de volgende twee usability-richtlijnen van een expert op het gebied van usability, Jakob Nielsen:

Help en documentatie:
Hoewel het beter is dat een systeem gebruikt kan worden zonder enige documentatie kan het toch nodig zijn om help en informatie te bieden. Zulke informatie moet makkelijk te doorzoeken zijn, gefocust zijn op de taken van de gebruiker en concrete stappen laten zien die uitgevoerd moeten worden.
Flexibiliteit en efficiëntie van gebruik:
Snelkoppelingen die niet gezien worden door een beginnende gebruiker kunnen de interactie met het systeem voor een ervaren gebruiker aanzienlijk versnellen.

figuur 8: Twee usability-richtlijnen van Jakob Nielsen

Ik zal verder op deze richtlijnen ingaan in hoofdstuk 6 waarin de ontwikkeling van de documenteditor wordt beschreven.

Andere systeemeisen kwamen tot stand door logisch na te denken of door middel van de kwaliteitseisen uit het plan van aanpak. Hiervan tot slot ook nog een voorbeeld:

OP.02 De nieuwe functionaliteit die ontwikkeld gaat worden, moet beheerd kunnen worden door Marbel Software.

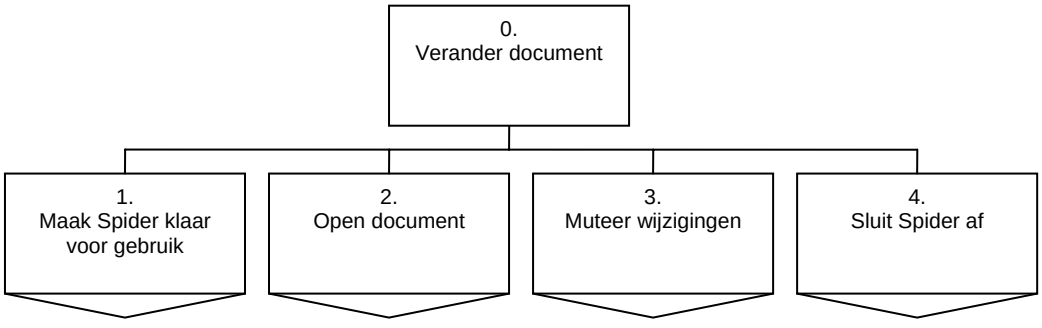
Om ervoor te zorgen dat er gemakkelijk verwezen kon worden naar de systeemeisen, heb ik ze genummerd per categorie. Vervolgens heb ik de eisen geconsolideerd tot één lijst. Die lijst heb ik vervolgens geprioriteerd zodat de belangrijkste eis bovenaan kwam te staan. Voor de complete lijst met systeemeisen verwijs ik u naar de definitiestudie. (Bijlage D)

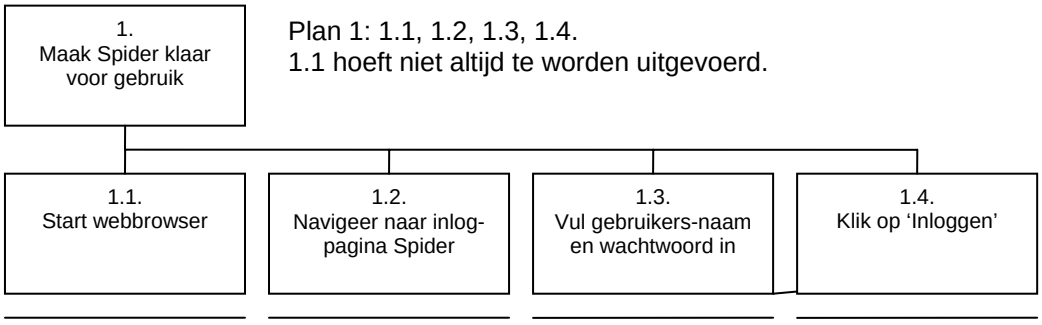
5.4. Bepalen van het systeemconcept

Toen de uitgangssituatie geanalyseerd was en de eisen voor de gewenste situatie opgesteld waren, kon er begonnen worden met het beschrijven van deze gewenste situatie, oftewel het bepalen van het systeemconcept. Het systeemconcept kan worden gezien als een globale beschrijving van het te ontwikkelen informatiesysteem.

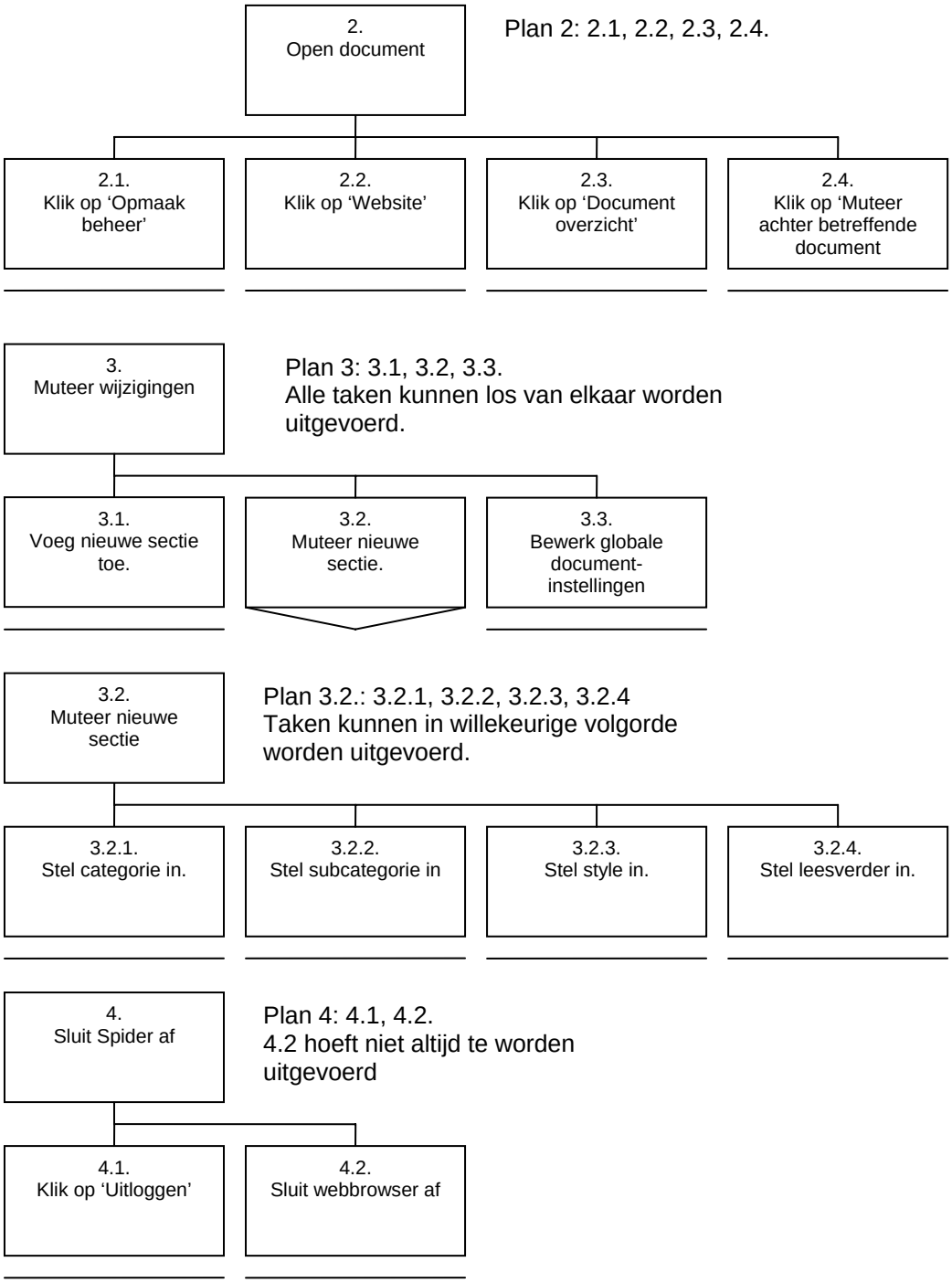
5.4.1. Taakdiagrammen gewenste situatie

Uit de tweede paragraaf van dit hoofdstuk is gebleken dat vooral in het beheergedeelte de functionaliteit erg onduidelijk was. In de derde paragraaf heb ik uitgelegd hoe ik de eisen heb opgesteld waar de te ontwikkelen oplossing aan moest voldoen. Deze twee aspecten moeten nu vertaald worden naar een globaal ontwerp voor een nieuwe gebruikersinterface. Om sitebeheerders te ondersteunen bij het beheren van een website moest er een gebruikersinterface komen die de gebruiker stuurt door het proces dat hij wil uitvoeren. Het grote tekstvlak uit de uitgangssituatie dat door de gebruiker gevuld moest worden met codes, wordt vervangen door een scherm waarbij de gebruiker door middel van het drukken op knoppen en het kiezen uit lijsten dezelfde taken uitvoert. Om een begin te maken met het ontwerpen van de nieuwe gebruikersinterface heb ik de taakdiagrammen uit de uitgangssituatie gedetailleerd zodat de kern hiervan (het daadwerkelijke muteren) meer taakgericht wordt Voor de gebruiker worden de taken makkelijker, omdat hij geen verstand hoeft te hebben van allerlei codes. (XML en CSS).
Hieronder een voorbeeld van een taakdiagram dat is gedetailleerd in de ‘gewenste’ situatie.





Plan 1: 1.1, 1.2, 1.3, 1.4.
1.1 hoeft niet altijd te worden uitgevoerd.

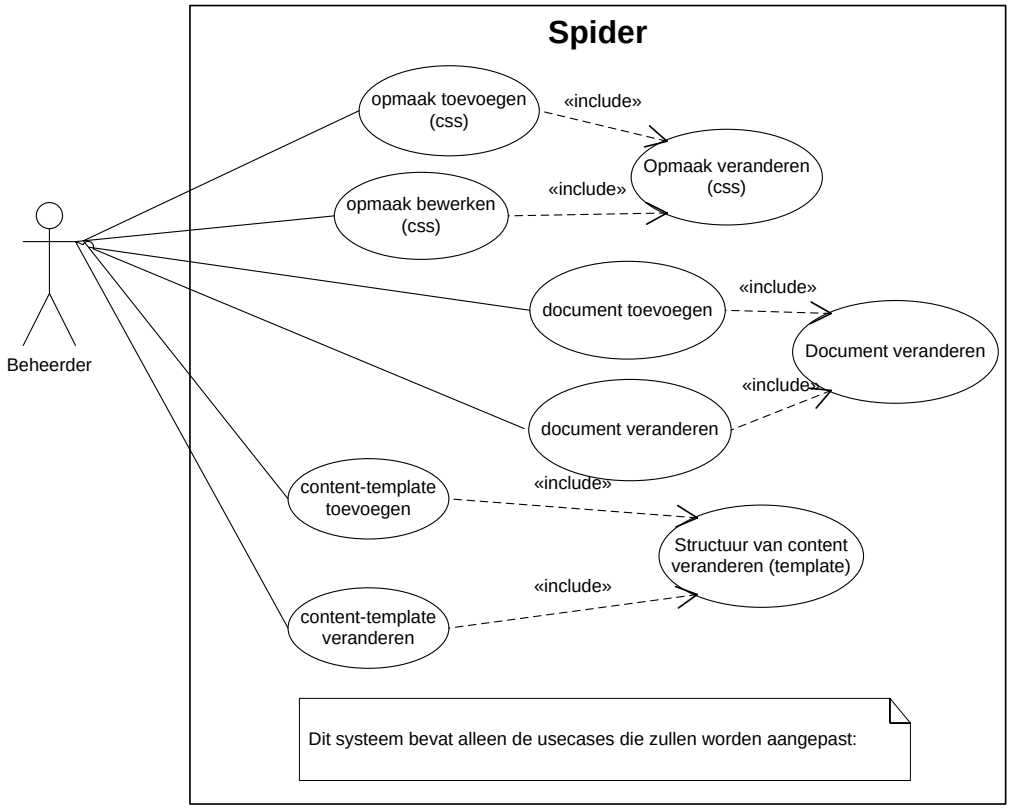


figuur 9:Taakdiagram gewenste situatie: document veranderen

De detaillering van de taken blijkt in dit figuur uit taak 3.2. De vier stappen die genomen moeten worden om die taak te volbrengen, moeten overeenkomen met het maken van keuzes (bijvoorbeeld het kiezen uit een dropdownlistbox of listbox) in de gebruikersinterface.

5.4.2. Objecten modeleren

In reguliere systeemontwikkelingtrajecten worden bij IAD in de activiteit ‘systeemconcept bepalen’ van de objecten globaal beschreven. Dit ontwikkeltraject maakt gebruik van objecten die al gemodelleerd en ontwikkeld zijn tijdens de initiële ontwikkeling van Spider. Daarom heb ik geen objecten gemodelleerd tijdens deze activiteit. Wel heb ik ervoor gekozen om een usecase-diagram te gebruiken om overzichtelijk te maken welke onderdelen van het systeem veranderd gingen worden. Hieronder het usecase-diagram met de usecase-beschrijving voor de document-editor dat ik gemaakt heb.



figuur 10: Use-case diagram

Binnen Spider wordt dezelfde pagina gebruikt om een onderdeel te veranderen en om een onderdeel toe te voegen. Dit heb ik in het usecase-diagram weergegeven door middel van include-afhankelijkheden.

Use case beschrijving document veranderen	
Naam	Document veranderen
Samenvatting	Een beheerder maakt een nieuw document aan of wijzigt een bestaand document en wijzigt de secties die worden weergegeven in dat document.
Actoren	Beheerder
Aannamen of precondities	• De gebruiker is ingelogd. • De gebruiker heeft beheerderrechten
Beschrijving scenario's	1. De gebruiker klikt in het menu op opmaak beheer. 2. De gebruiker klikt in het menu op website 3. De gebruiker klikt in het menu op document overzicht. 4. De gebruiker klikt in de lijst met documenten op 'Muteer' achter het te muteren document of voegt een nieuw document toe met de knop 'nieuw document'. 5. De gebruiker vult bovenaan de pagina de globale eigenschappen van het document in. 6. De gebruiker voegt een nieuwe sectie toe. 7. De gebruiker klikt in de lijst met gedefinieerde secties op wijzigen. 8. De gebruiker past de sectie-eigenschappen aan en klikt op 'opslaan'. Indien er syntaxfouten bevinden in de XML-code, treedt de uitzondering XMLSyntaxFout op.
Uitzonderingen	[XMLSyntaxFout]: Geeft een melding dat het document niet kan worden opgeslagen omdat er een fout zit in de XML-syntax.
Resultaat of postcondities	Document is aangepast of toegevoegd.

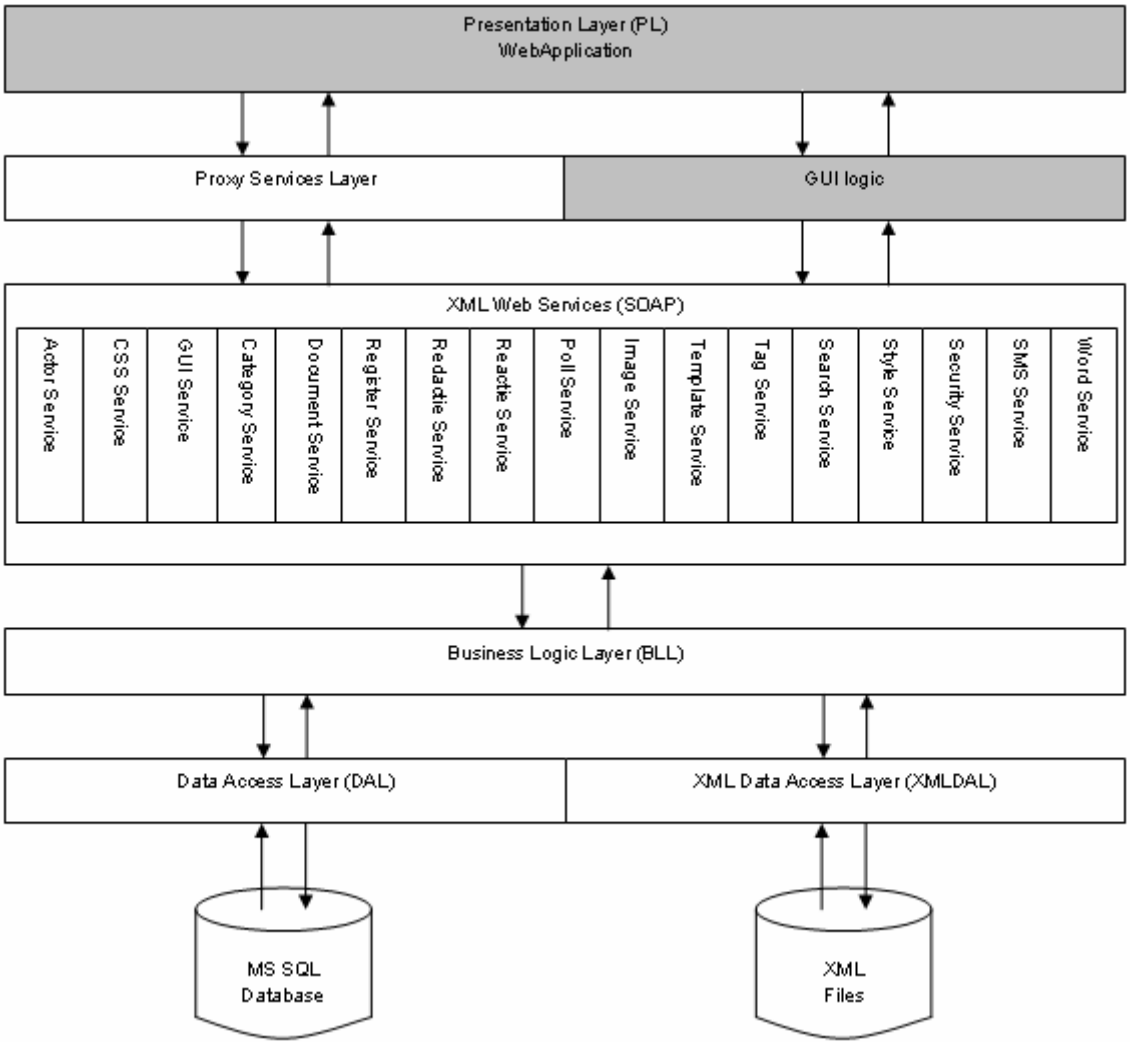
figuur 11: Usecase beschrijving gewenste situatie document veranderen

De usecase-beschrijving beschrijft het scenario dat doorlopen wordt bij het veranderen van het document.
Het systeemconcept dat ik heb gemaakt, is alleen een globale beschrijving van de te ontwikkelen oplossing. Een gedetailleerde beschrijving wordt gemaakt in de pilotplannen.

5.5. Beschouwen van de technische structuur

Bij reguliere systeemontwikkelingtrajecten waarbij IAD wordt gebruikt als methodiek, wordt in de activiteit 'beschouw technische structuur' een inschatting gemaakt van de bronnen (hard- en software) die nodig zijn voor de ontwikkeling en implementatie van het systeem. Verder wordt er in kaart gebracht welke (externe of eerder ontwikkelde) componenten herbruikbaar zijn binnen het project.
Omdat dit ontwikkeltraject een andere insteek heeft en het alleen draait om het verbeteren van de gebruikersinterface, heb ik ervoor gekozen om me tijdens deze activiteit te verdiepen in de technische structuur en architectuur van Spider. Ik heb deze keuze gemaakt omdat de gebruikersinterface die ik zou gaan ontwikkelen volledig moest aansluiten bij de bestaande architectuur van Spider.
Omdat er geen systeemdocumentatie beschikbaar was van Spider heb ik de technische structuur grotendeels in kaart gebracht door de broncode te bestuderen. Verder heb ik informatie verkregen door de ontwikkelaars van Spider te interviewen.
Spider is ontwikkeld met Microsoft Visual Studio .NET 2003 in de programmeertaal C# op het ASP.NET platform. Bij de ontwikkeling van toepassingen maakt Marbel Software gebruik van SourceGear Vault als versiebeheer- en samenwerkingssysteem. SourceGear Vault maakt gebruik van Microsoft SQL Server voor het opslaan van de broncode. In het kader van mijn afstudeerproject is er een aparte branch gemaakt waarin ik kon ontwikkelen. De branch was in dit geval een kopie van de laatste versie van Spider. Op deze manier kon ik volledig zelfstandig ontwikkelen en had ik in principe

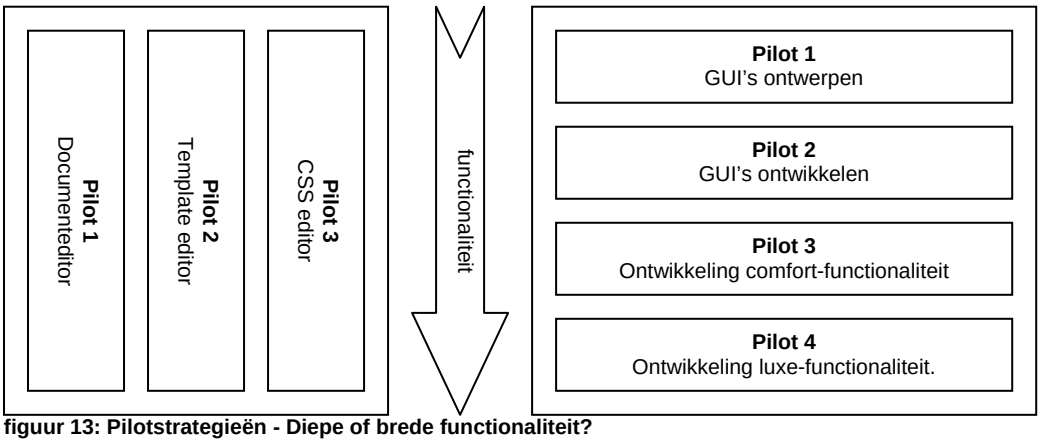
alle rechten om aanpassingen te maken. Bij de invoering hoefden dan alleen de gewijzigde componenten te worden gekopieerd naar de productie-branch. Visual Studio werkt met zogenaamde solutions die kunnen bestaan uit meerdere projecten. Om de technische structuur van Spider in kaart te brengen, heb ik de broncode van de verschillende projecten waar Spider uit bestaat, geanalyseerd om te kunnen zeggen wat precies de functie van elk project is en wat de relatie met andere projecten is. Tijdens het analyseren heb ik vooral gekeken naar hoe er in elk project gebruik gemaakt wordt van objecten uit andere projecten. Hierdoor werd me duidelijk hoe de onderlinge relaties tussen de projecten liggen. Deze relaties heb ik vervolgens zichtbaar gemaakt in het volgende figuur.



figuur 12: Schematische voorstelling technische structuur Spider

5.6. Opstellen van het pilotplan

Nu ik een duidelijk beeld had van de technische structuur van Spider, kon ik beginnen met het uitvoeren van de laatste activiteit binnen de IAD-fase definitiestudie, het opstellen van het pilotplan. In het pilotplan worden de pilots definitief gespecificeerd en wordt er een gedetailleerde planning gemaakt. Ik had al een voorlopige pilotindeling gemaakt bij het vaststellen van het ontwikkelscenario (beschreven in paragraaf 5.1.5). Deze indeling heb ik bij het opstellen van het pilotplan heroverwogen. Een pilot zie ik als een onderdeel van het systeem dat apart wordt ontwikkeld en gebruikt kan worden. Het pilotplan bevat een geprioriteerde lijst van pilots die sequentieel of parallel ontwikkeld en ingevoerd gaan worden. Verder wordt voor elke pilot wordt een inschatting gemaakt van de benodigde middelen en tijd.
Er waren twee duidelijke strategieën om de pilots in te delen waaruit ik moest kiezen. De eerste strategie gaat uit van pilots met een diepe functionaliteit en de tweede indeling gaat uit van een brede functionaliteit per pilot. Ik heb gekozen voor de strategie waarbij de pilots een diepe functionaliteit hebben. Voordeel hiervan is namelijk dat als een bepaalde pilot wegens tijdgebrek niet meer ontwikkeld kan worden, de andere pilots nog wel ingevoerd kunnen worden. Bij pilots met een brede functionaliteit kan het gebeuren dat als een pilot wegens tijdsgebrek niet meer kan worden ontwikkeld, er geen resultaat wordt behaald voor de opdrachtgever omdat de pilot niet kan worden ingevoerd.
De volgende afbeelding illustreert de twee strategieën. Ook de pilotdelen zijn op deze manier ingericht. Zelfs na de eerste iteratie is er al resultaat voor de opdrachtgever.



figuur 13: Pilotstrategieën - Diepe of brede functionaliteit?

De grote lijnen van de oorspronkelijke pilotindeling uit het ontwikkelscenario bleven dus intact.
Nu de verschillende pilots benoemd waren, moesten deze verder gespecificeerd worden en moest er een gedetailleerde planning per pilot opgesteld worden. Hiervoor heb ik eerst per pilot verschillende pilotdelen benoemd. Dit heb ik gedaan door in te schatten welke taken er verricht moesten worden. Deze heb ik vervolgens geclusterd. Daarna heb ik de benodigde tijd die ik nodig zou hebben om de pilot te ontwikkelen ingeschat. Dit vond ik erg moeilijk aangezien ik geen idee had of ik veel tegenslagen zou krijgen. De tijd heb ik uiteindelijk bepaald door naar de complexiteit van de XML-structuur te kijken samen met andere specifieke zaken die voor die pilot van toepassing waren. Zo kreeg ik een indicatie van welke pilot het meest complex zou zijn. Op basis van die complexiteit heb ik de tijd die ik tijdens het opstellen van het pilotplan nog beschikbaar had voor ontwikkeling verdeeld tussen de pilots.
De pilot 'style-editor' en 'overige aanpassingen in Spider' zijn tijdens de fase pilotontwikkeling komen te vervallen op advies van een collega omdat de opdracht anders te breed werd.
De volgende afbeelding laat een beschrijving van de pilot CSS-editor zien. Deze was opgenomen in de eerste versie van het pilotplan.

<table>
<tr><th colspan="2">Pilot 2: CSS editor</th></tr>
<tr><td>Omschrijving</td><td>De eerste pilot bevat alle functionaliteit voor de CSS-editor. Hiermee kan op eenvoudige wijze een stylesheet worden samengesteld. De stylesheet moet worden opgeslagen in XML-formaat. Daarvoor moet er een XSL-Transformatie plaatsvinden.</td></tr>
<tr><td>Pilotdelen</td><td>• CSS -> XML transformatie.
• Ontwerpen van de GUI
• Specificeren mogelijke acties
• Ontwikkelen Data Access Layer
• Ontwikkelen Business Objects
• Ontwikkelen GUI instructies</td></tr>
<tr><td>Schatting benodigde tijd</td><td>2 weken</td></tr>
<tr><td>Gewenst kwaliteitsniveau</td><td>Hoog, er is bij de gebruikers van Spider veel vraag naar functionaliteit om op een makkelijke manier de opmaak te veranderen. Dit moet mogelijk worden gemaakt door de CSS editor.</td></tr>
<tr><td>Tijdsplanning (timebox)</td><td>
<table>
<tr><th colspan="2">Datum</th><th>28-03-05</th><th>29-03-05</th><th>30-03-05</th><th>31-03-05</th><th>01-04-05</th><th>04-04-05</th><th>05-04-05</th><th>06-04-05</th><th>07-04-05</th><th>08-04-05</th></tr>
<tr><th>Activiteit</th><th>Prioriteit</th><th>M</th><th>D</th><th>W</th><th>D</th><th>V</th><th>M</th><th>D</th><th>W</th><th>D</th><th>V</th></tr>
<tr><td>CSS <-> XML transitie</td><td>Basis</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>Ontwerpen GUI</td><td>Basis</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>Acties specificeren</td><td>Basis</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>Ontwikkeling DAL</td><td>Basis</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>Ontwikkeling BO</td><td>Basis</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>Ontwikkeling GUI</td><td>Basis</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
</td></tr>
</table>

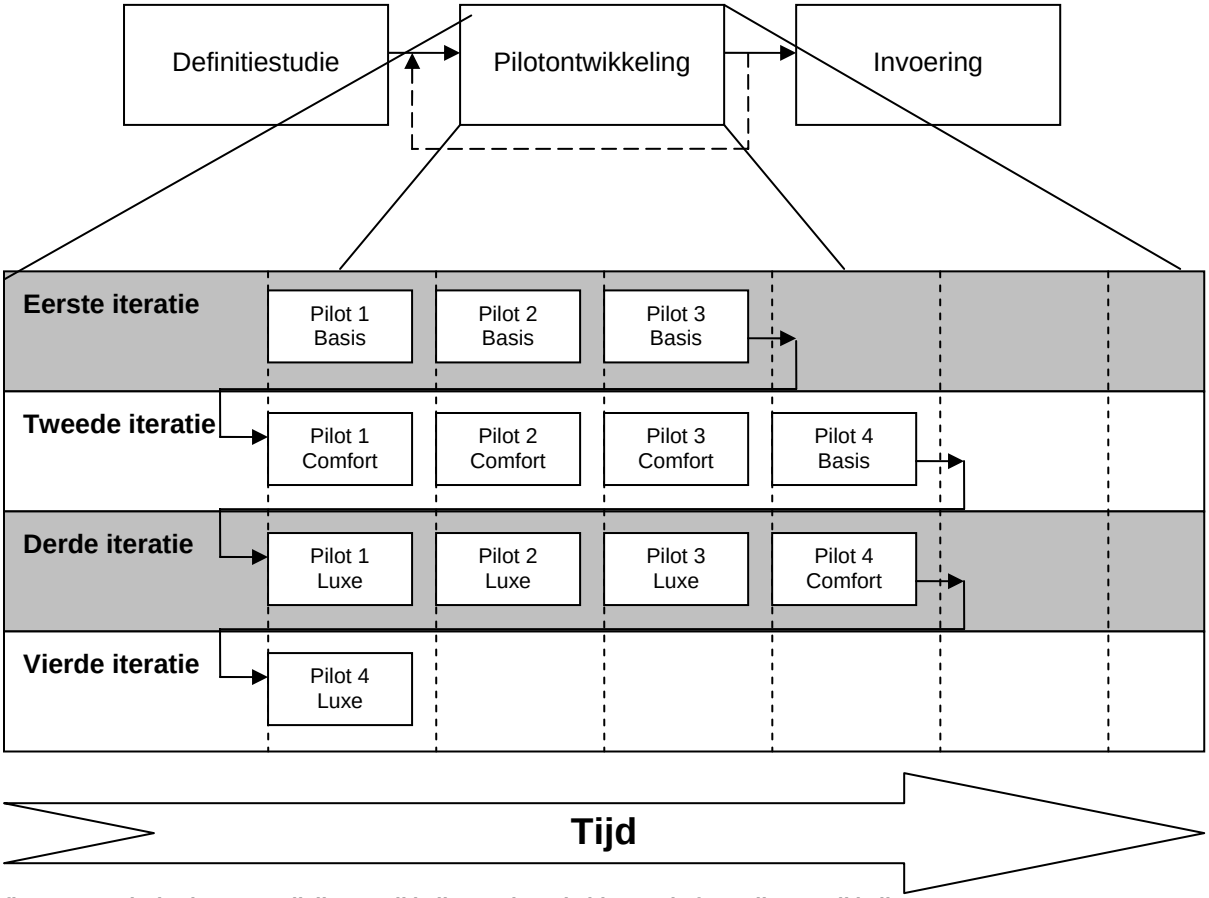
figuur 14: Beschrijving van pilot 2 (CSS-editor) uit definitiestudie versie 1.0

Toen ik begon aan de fase pilotontwikkeling kwam ik er meteen achter dat de planning en gespecificeerde pilotdelen niet voldeden. Ik kwam erachter dat ik helemaal niets aan de DAL of Business objects hoefde te wijzigen. Het gedeelte wat ik zou ontwikkelen, speelt zich voor het grootste gedeelte af binnen de presentatie-laag (zoals bechreven in figuur 12) en voor een klein gedeelte in de GUI logic-laag. Toen ik hierachter kwam, heb ik de pilotdelen van alle pilots opnieuw gedefinieerd. In nog een andere versie van de definitiestudie heb ik ook nog eens de volgorde van de pilots aangepast waardoor de beschrijving er als volgt uit kwam te zien.

Pilot 3: CSS editor	
Omschrijving	De eerste pilot bevat alle functionaliteit voor de CSS-editor. Hiermee kan op eenvoudige wijze een stylesheet worden samengesteld. De stylesheet moet worden opgeslagen in XML-formaat. Daarvoor moet er een XSL-Transformatie plaatsvinden.
Pilotdelen	• Ontwerpen XML-model voor CSS (basis) • XML -> CSS transformatie (basis) • Ontwerpen van de GUI (basis) • Ontwikkeling van de GUI (basis) • Code opschonen / herstructureren / commentarieren (comfort) • XML DAL gereedmaken voor opslag CSS-XML (luxe)
Schatting benodigde tijd	2 weken
Gewenst kwaliteitsniveau	Hoog, er is bij de gebruikers van Spider veel vraag naar functionaliteit om op een makkelijke manier de opmaak te veranderen. Dit moet mogelijk worden gemaakt door de CSS editor.
Planning	Eerst moet er een XML-model ontworpen worden om CSS in op te kunnen slaan. Als dit gebeurd is, kan de transformatie van XML naar CSS via XSL-T ontwikkeld worden. Vervolgens kan de GUI ontworpen en ontwikkeld worden. In de tweede iteratie wordt de code opgeschoond en becommentarieerd. Verder wordt de reeds ontwikkelde XML Data Access Layer aangepast om XML op te kunnen slaan.

figuur 15: Beschrijving van pilot 3 (CSS-editor) uit definitiestudie versie 1.2

Bij een andere aanpassing van het pilotplan heb ik de dagplanning verwijderd uit de pilotbeschrijving. Ik heb dit gedaan omdat in praktijk niet veel van deze planning terecht kwam. Sommige activiteiten namen extra veel tijd in beslag en andere activiteiten gingen onverwacht voorspoedig. Ik denk dat dit kwam doordat ik nog weinig ervaring had met de gebruikte technieken. De schatting van de benodigde tijd vond ik niet concreet genoeg om als echte planning te hanteren. Daarom heb ik het pilotplan uitgebreid met een meer gedetailleerde iteratiestrategie die wordt weergegeven in het volgende figuur. Op het model is goed te zien in welke volgorde de verschillende pilots worden ontwikkeld en komt het verband naar voren tussen de prioritering van de verschillende pilotdelen en de iteraties. Ik heb gekozen voor de onderstaande strategie omdat de prioriteit van de verschillende pilots onderling ook verschilt. Eerst worden de belangrijkste delen van de belangrijkste pilots ontwikkeld, Daarna de minder belangrijke delen van die pilots, die toch nog belangrijker zijn dan de belangrijkste delen van pilot 4. Een voordeel van deze tactiek is dat er in ieder geval resultaat wordt bereikt voor de opdrachtgever.

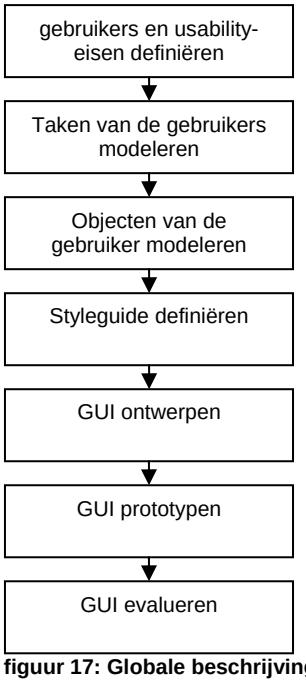


figuur 16: Prioritering – Parallelle ontwikkeling en iteratie binnen de fase pilotontwikkeling

Het opstellen van het pilotplan was de laatste activiteit die ik verricht heb in het kader van de definitiestudie.

6. Ontwikkelen van de documenteditor

In het vorige hoofdstuk heb ik uitgelegd hoe de IAD-fase definitiestudie is doorlopen. De volgende fase in de IAD-ontwikkelcyclus is 'pilotontwikkeling'. In deze fase heb ik per pilot steeds eerst een ontwerp gemaakt waarna de ontwikkeling kon plaatsvinden. Ik heb ervoor gekozen om deze fase te beschrijven in drie hoofdstukken. In elk hoofdstuk komt de ontwikkeling van een pilot eerste drie aan bod. Ik heb gekozen voor deze hoofdstukindeling omdat het hele proces op deze manier chronologisch beschreven wordt.
De eerste pilot die ik ontwikkeld heb, is de documenteditor. De documenteditor moest de mogelijkheid bieden tot het aanmaken en aanpassen van Spider-documenten zonder kennis te hoeven hebben van XML. Dit hoofdstuk beschrijft het proces dat ik doorlopen heb om de documenteditor te ontwikkelen. Voor het complete pilotontwikkelplan van deze pilot verwijs ik u naar de externe bijlagen (Bijlage E). Ik heb ervoor gekozen om het ontwikkelen van de GUI niet specifiek volgens een bepaalde methode te doen (zoals GUIDE) omdat ik bang was dat de balast van het gebruik van de methode groter zou zijn dan het uiteindelijke voordeel dat het op zou leveren. Het proces dat ik gevolgd heb, vertoont wel een aantal overeenkomsten met GUIDE.



figuur 17: Globale beschrijving GUIDE methode

Bij GUIDE worden eerst de gebruikers en usability-eisen bepaald en de taken gemodelleerd. Dit heb ik al gedaan in de fase definitiestudie. Het modelleren van de objecten doe ik hier niet. In plaats daarvan heb ik de broncode bekeken en de XML-structuur geanalyseerd. De styleguide hoefde niet gemaakt te worden, omdat die alvast lag in een CSS-bestand waar de interface van Spider zich op baseert. Verder heb ik ook een stap meegemaakt waarin de GUI is ontworpen en ontwikkeld. De hele totstandkoming van de GUI zal ik beschrijven in dit hoofdstuk.

6.1. Ontwerpen van de GUI

Bij het ontwerpen van de verschillende GUI's ben ik uitgegaan van de usability-richtlijnen van Jakob Nielsen en Donald Norman. Deze richtlijnen worden vaak gebruikt bij heuristic evaluations, een techniek waarbij een usability-expert bepaalde software of een website toetst aan de hand van verschillende vooraf opgestelde criteria. Ik heb bij het ontwerpen van de GUI zoveel mogelijk rekening gehouden met deze richtlijnen. Ik heb de richtlijnen weergegeven in figuur 18 op de volgende pagina.

Norman:

A *Goede zichtbaarheid*
- Kan de gebruiker duidelijk zien wat de mogelijke acties zijn en hoe deze acties moeten worden uitgevoerd?

B *Goed conceptueel model*
- Is de gebruiker in staat om de uitkomst van zijn eigen acties te voorspellen doordat hij een mentaal beeld van het systeem heeft dat overeenkomt met de werkelijkheid?

C *Duidelijke mapping*
- Is er sprake van een directe link tussen de vereiste actie van de gebruiker en de daadwerkelijke uitkomst daarvan.

D *Duidelijke feedback*
- Krijgt de gebruiker duidelijke feedback van het systeem? Hoe zit het met foutmeldingen?

Nielsen:

1 *Zichtbaarheid van de status van het systeem*
- Is het op elk moment duidelijk waar het systeem mee bezig is?

2 *Overeenkomst tussen het systeem en de echte wereld*
- Spreekt het systeem de taal van de gebruiker?

3 *Vrijheid van de gebruiker*
- Is de gebruiker in staat om gemaakte fouten ongedaan te maken?

4 *Consistentie en standaarden*
- Houdt het systeem zich aan platformgerelateerde afspraken en conventies?

5 *Voorkomen van fouten*
- Voorkomt het systeem fouten van de gebruiker door duidelijkheid te bieden?

6 *Herkennen in plaats van herstellen*
- Zijn de objecten, acties en opties duidelijk? De gebruiker moet geen informatie hoeven te onthouden van een bepaald deel van een dialoogvenster naar een ander deel. Instructies voor het gebruik van het systeem dienen duidelijk zichtbaar te zijn of makkelijk opvraagbaar wanneer dat gewenst is.

7 *Flexibiliteit en efficiëntie van gebruik*
- Snelkoppelingen die niet gezien worden door een beginnende gebruiker kunnen de interactie met het systeem voor een ervaren gebruiker aanzienlijk versnellen.

8 *Mooi en minimalistisch ontwerp*
- Dialoogvensters moeten geen informatie bevatten die irrelevant of nauwelijks nodig is.

9 *Help gebruikers bij het herkennen en herstellen van fouten*
- Foutmeldingen moeten uitgedrukt zijn in normale taal (geen codes), moeten het probleem precies omschrijven en een mogelijke oplossing bieden.

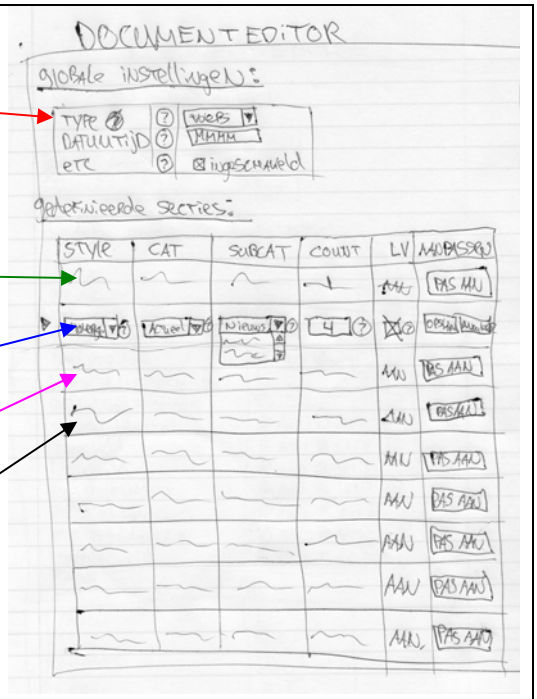
10 *Help en documentatie*
- Hoewel het beter is dat een systeem gebruikt kan worden zonder enige documentatie kan het toch nodig zijn om help en informatie te bieden. Zulke informatie moet makkelijk te doorzoeken zijn, gefocust zijn op de taken van de gebruiker en concrete stappen laten zien die uitgevoerd moeten worden.

figuur 18: Usability-richtlijnen van Norman en Nielsen

Nielsen zegt in regel 2 dat het systeem overeenkomst dient te hebben met de echte wereld. Donald Norman beschrijft een vergelijkbaar principe (mapping) in regel C. Dit principe heb ik toegepast bij het ontwerpen van de GUI. Ik zal dit proberen uit te leggen aan de hand van een voorbeeld. Het is vergelijkbaar met een soort mal: Als een kind een zandkasteel bouwt met allerlei vormpjes en hij wil een kegelvormige punt op een van zijn torens zet, weet hij precies welk vormpje hij daarvoor nodig heeft: het kegelvormige vormpje. Van dit principe ben ik uitgegaan bij het ontwerpen van de GUI. XML in een bepaald formaat (het zand in een bepaalde vorm) is hetgeen dat gemaakt moest kunnen worden met de documenteditor (mal in een bepaalde vorm). Daarom heb ik eerst het XML-formaat dat de documenteditor moest kunnen genereren en bewerken, geanalyseerd. Dit heb ik gedaan door verschillende documentdefinities van Spider te openen en te bekijken. Door goed naar de structuur te kijken en terugkerende patronen in kaart te brengen, kreeg ik een beeld van de onderdelen waar de documenteditor uit moest bestaan.

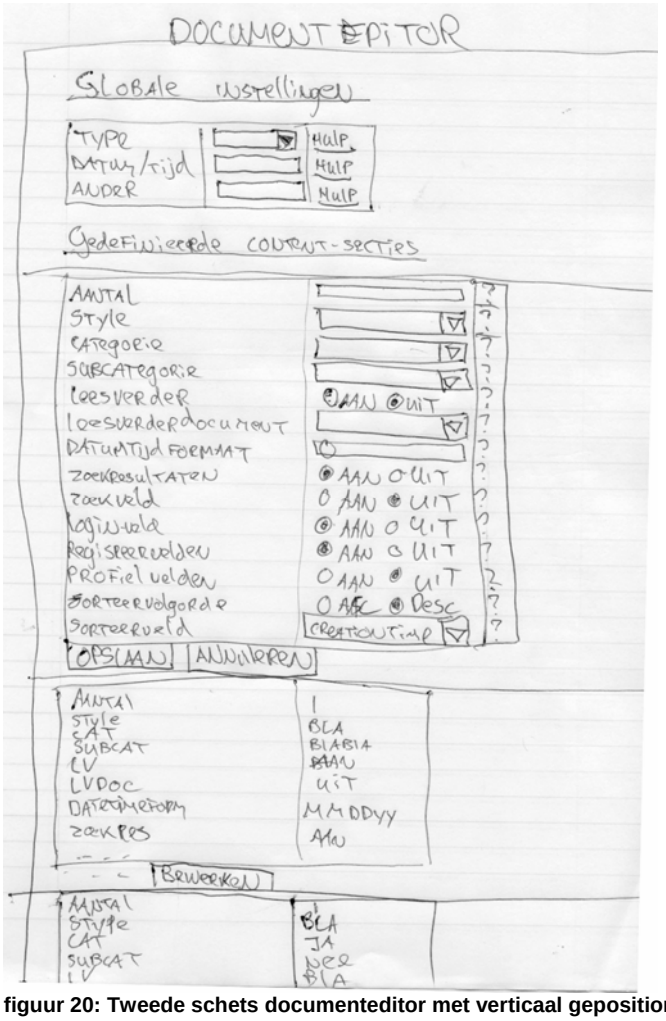
Als je een documentdefinitie opent, kun je globaal gezien twee delen herkennen. Een algemeen gedeelte waarin globale instellingen van dat document worden opgeslagen en een gedeelte waarin een aantal secties worden gedefinieerd die elk steeds terugkerende eigenschappen hebben.
Deze eigenschappen heb ik vervolgens vertaald naar een eerste schets. Dit 'vertaalproces' (in het voorgaande voorbeeld vergelijkbaar met het maken van een afgietsel van gips in het zand) heb ik in het volgende figuur proberen weer te geven.

```
<document type="web" dateTimeFormat=""
currentTimeFormat="dddd d MMMM">
   <head>
      <title>
         NDLite.nl - Voorpagina
      </title>
      <custom> <![CDATA[]]> </custom>
   </head>
   <sections>
      <section>
         <content style="Actueel" category="Actueel"
         subcategory="Nieuws" count="6"
         leesVerderDocument="leesverderactueel" />
      </section>
      <section>
         <content count="1" style="Column"
         category="Column" subcategory="Nieuws" />
      </section>
      <section>
         <content style="Home" category="Home"
         subcategory="Nieuws" count="6"
         leesVerderDocument="leesverderhome" />
      </section>
      <section>
         <content style="Layout" category="Layout"
         subcategory="EindMidden" />
      </section>
   </sections>
</document>
```



figuur 19: links: spider documentdefinitie, Rechts: Eerste schets documenteditor

In deze eerste schets bestaat de documenteditor (net als de XML documentdefinitie) uit twee delen. Bovenaan de globale instellingen zoals de titel van het document en de datumtijd-instellingen en daaronder een tabel met daarin de verschillende content-secties. Achter elke rij staat een 'pas aan'-knop. Als daar op gedrukt wordt verandert de rij in een aanpasbare rij. Afhankelijk van het soort veld dat aangepast moet worden, wordt er een ander schermelement geboden om een waarde te kiezen. Voor het aanpassen van vrije text is er een textbox, voor meerkeuze-opties een dropdownlistbox en voor aan/uit-opties een checkbox.
Achter elke eigenschap die gewijzigd kan worden, bevindt zich een vraagteken waarop geklikt kan worden. Er verschijnt dan helptekst waarin de betreffende eigenschap wordt uitgelegd. Dit is overeenkomstig met regel 10 uit figuur 18.
Toen ik een schermontwerp van dit ontwerp aan een collega liet zien kwam ik er echter achter dat er een aantal nadelen kleefden aan dit ontwerp. Een belangrijk nadeel is namelijk dat er maar ruimte is voor een beperkt aantal kolommen en dus ook maar voor een beperkt aantal eigenschappen per content-sectie van de documentdefinitie. Hij wees me erop dat er veel meer dan 5 eigenschappen mogelijk waren. De exacte eigenschappen heb ik later in de broncode van Spider (in het business logic project in het gedeelte waar de documentdefinities gelezen en verwerkt worden) teruggevonden. Het ging in totaal om 14 eigenschappen. Omdat 14 eigenschappen niet als kolommen op het scherm passen en het door internetgebruikers als storend wordt ervaren om naar rechts te scrollen, heb ik een nieuw ontwerp gemaakt.



figuur 20: Tweede schets documenteditor met verticaal gepositioneerde eigenschappen

De tweede schets (figuur 20) gaat uit van hetzelfde principe als de eerste schets. De tabel waar in elke kolom een eigenschap kon worden aangepast, heb ik in deze schets vervangen door meerdere tabellen onder elkaar. De verschillende eigenschappen worden hier als rijen in een tabel weergegeven zodat er in geen geval meer verticaal hoeft te worden gescrold. Als er een regel moet worden aangepast, klikt de gebruiker net als bij de eerste schets op de ‘bewerken’-knop. In de tweede kolom van het betreffende item worden de waarden vervangen door schermelementen om de waarden aan te passen. Verder is het principe van deze schets hetzelfde als de eerste. Ik zal nu in het volgende figuur mijn keuze voor de gebruikte schermelementen per eigenschap toelichten.

eigenschap	motivatie schermelement
aantal	Als de eigenschap ‘aantal’ is gedefinieerd, moet het een numerieke waarde bevatten. Ik heb gekozen voor een tekstvlak waar de gebruiker zelf een getal in kan tikken omdat het snelste werkt. Om te voorkomen dat de gebruiker onjuiste waarden invoert (zoals letters, spaties en speciale tekens), zal de invoer moeten worden gevalideerd voor dat die kan worden geaccepteerd.
stijl	De eigenschap stijl bevat een naam van het stijldocument (XSL-T) dat gehanteerd wordt om de betreffende sectie weer te geven. Omdat hier alleen een keuze moet worden gemaakt uit bestaande stijlen, heb ik ervoor gekozen om een dropdownlistbox te gebruiken. De verschillende namen van stijlen moeten door de GUI worden opgehaald door middel van de style-webservice. Andere mogelijkheden om een item uit een lijst te kiezen zijn de listbox of radiobuttonlist. Maar omdat deze in de hoogte veel ruimte innemen heb ik toch gekozen voor de dropdownlistbox.
leesverderdocument	Voor het kiezen van een document heb ik gekozen voor een dropdownlistbox

eigenschap	motivatie schermelement
	met dezelfde motivatie als bij 'stijl'. Het enige verschil is dat de waarden waaruit gekozen kan worden, verkregen moeten worden door middel van de document-webservice.
(sub)categorie	Aanvankelijk had ik voor het kiezen van een categorie of subcategorie een dropdownlistbox in gedachte met dezelfde redenering als bij de stijl en leesverderdocument. Toen ik er door een collega op gewezen werd dat er meerdere categorieën geselecteerd moesten kunnen worden, heb ik dit echter bijgesteld naar twee listboxen met twee knoppen ertussen: Een knop om een item uit de linker listbox te verwijderen en een knop om een item uit de rechter listbox in de linker listbox te plaatsen. Ik heb hier gekozen voor listboxen omdat de gebruiker zo meteen het resultaat van zijn actie kan zien. Als hij een item heeft toegevoegd, ziet hij een nieuw item in de listbox staan.
sorteerveld	Bij het sorteerveld moet een statische waarde worden gekozen uit 4 mogelijkheden. Om zo min mogelijk ruimte in te nemen op het beeldscherm heb ik hier een dropdownlistbox gebruikt.
overige eigenschappen	De overige eigenschappen zijn allemaal opties waarbij steeds gekozen kan worden tussen twee waarden: aan en uit. Daarom zou het voor de hand liggend zijn geweest om hiervoor checkbox-elementen te gebruiken. Ik heb hier echter gekozen voor radiobuttonlists omdat er hier technisch gezien ook een verschil bestaat tussen de optie 'uit' en 'niet gedefinieerd'. Als geen van de radiobuttons is aangevinkt moet dit resulteren in een xml-regel in de document-definitie waar het attribuut dat correspondeert met de betreffende eigenschap niet is gedefinieerd.

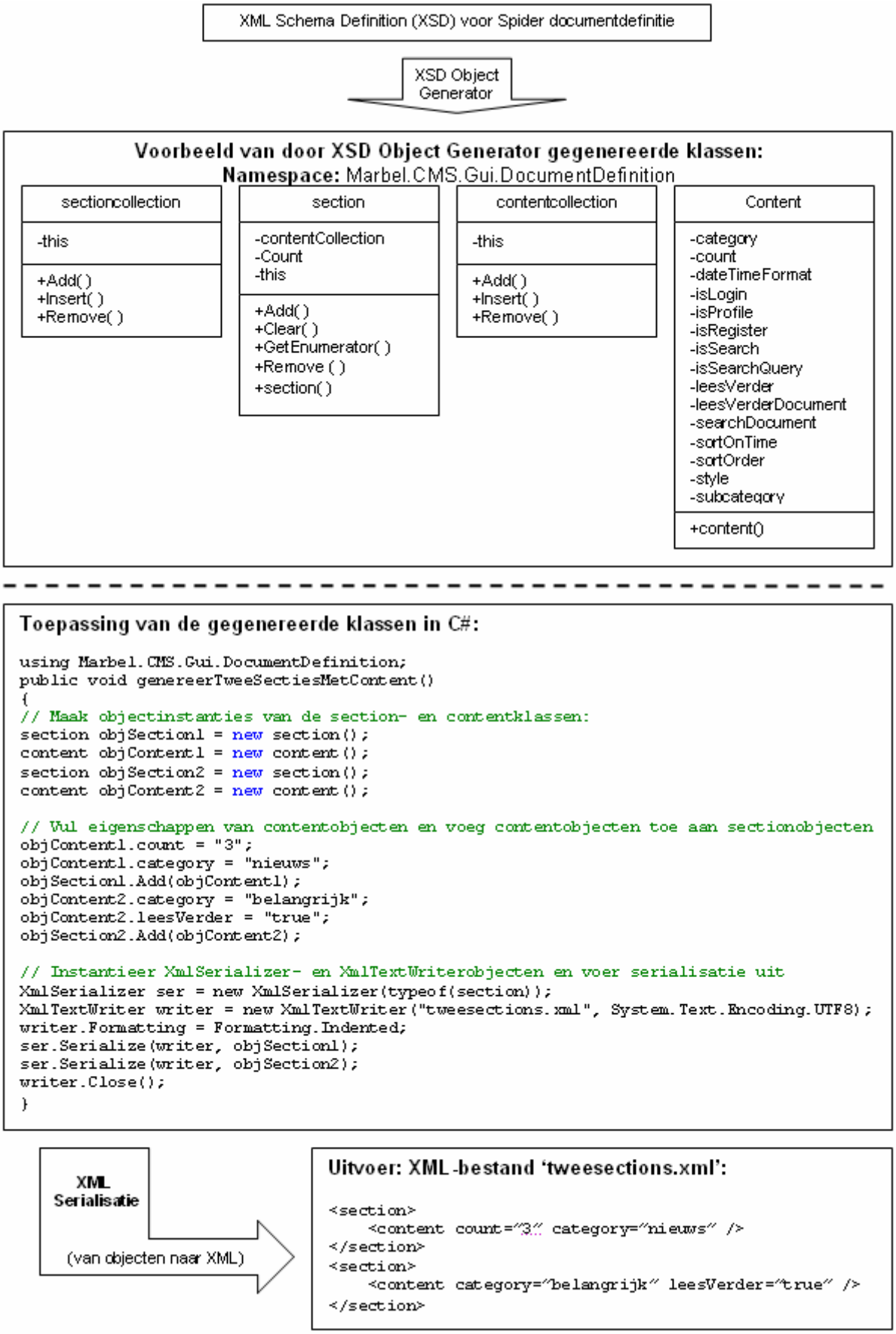
figuur 21: Motivatie keuze schermelementen voor documenteditor

6.2. Kiezen van een geschikte techniek

Om het ontwerp uit figuur 20 om te kunnen zetten naar een werkende interface, moest ik op zoek naar een techniek die het volgende kon bereiken:

- Objecten dynamisch op het scherm tonen, zodat ze ook nog gewijzigd kunnen worden.
- De XML documentdefinitie vertalen naar objecten.

Het eerste aspect van de techniek had ik al snel gevonden in de vorm van de datagrid-control. De datagrid-control zit standaard inbegrepen bij het Microsoft .NET Framework. Als de datagrid-control op een ASP.NET pagina wordt geplaatst, kan het dataverzamelingen uit verschillende bronnen weergeven in tabelvorm. Als de pagina wordt opgevraagd door een webbrowser wordt de datagrid-control gerenderd als een normale HTML TABLE. Behalve de mogelijkheid om datavelden als kolommen te gebruiken, kunnen er ook template-kolommen worden gedefinieerd. Door middel van het gebruik van template-kolommen heeft de ontwikkelaar volledige controle over de representatie van de gegevens. De lay-out van een datagrid-control kan worden vastgelegd in een stylesheet (CSS). Op deze manier is het eenvoudig om alle datagrids binnen een webapplicatie dezelfde lay-out te geven.
Het tweede aspect van de techniek (het vertalen van de XML document-definitie naar objecten) was iets ingewikkelder.
Een collega wees me op de XSD object generator, een gratis tool die Microsoft beschikbaar stelt om een XSD-schema te vertalen naar klassen (geprogrammeerd in C#). Ik heb een paar dagen met deze techniek geëxperimenteerd om te bekijken hoe ik het kon toepassen binnen mijn document-editor.
De XSD object generator is gebaseerd op het principe van XML-serialisatie. Bet behulp van XML-serialisatie en deserialisatie kunnen objecten geconverteerd worden naar XML-data en andersom. Het figuur op de volgende bladzijde laat het proces van XML-serialisatie ten behoeve van de documenteditor zien waarbij de klassen worden gegenereerd door de XSD object generator.



figuur 22: Toepassing van de XSD Object Generator en het serialiseren van XML

Tijdens het uitproberen van deze techniek, kwam ik er echter achter dat het veel te omslachtig is voor gebruik binnen een grafische interface. Het inzetten van deze techniek zou betekenen dat als Spider aangepast zou worden waarbij er nieuwe mogelijkheden worden toegevoegd voor de document-definitie, er onnodig veel werk nodig is om de document-editor aan te passen om de nieuwe eigenschap te kunnen verwerken. Een andere reden waarom ik uiteindelijk afgezien heb van het gebruiken van XML-serialisatie is dat ik de klassen die gegenereerd werden niet goed kon koppelen aan de datagrid-control. Daarom heb ik er uiteindelijk voor gekozen om het hele serialisatie-principe te laten vallen en een andere oplossing te zoeken.
Toen ik op internet opzoek was naar alternatieven stuitte ik een artikel waarin de datagrid rechtstreeks gebonden werd aan het XmlNodeList-object. Dit object kan eenvoudig geïnitieerd worden met een XPath-query op een XmlDocument. Met dit principe ben ik gaan experimenteren. Toen ik een XmlNodeList-object met daarin alle secties van een document-definitie bondt aan datagrid-control, kreeg ik de volgende uitvoer.

Prefix	HasChildNodes	Value	HasAttributes	NamespaceURI	InnerText	InnerXml	Name	OuterXml	BaseURI	LocalName	IsReadOnly	IsEmpty
	True		False				section			section	False	False
	True		False				section			section	False	False
	True		False				section			section	False	False
	True		False				section			section	False	False
	True		False				section			section	False	False
	True		False				section			section	False	False
	True		False				section			section	False	False

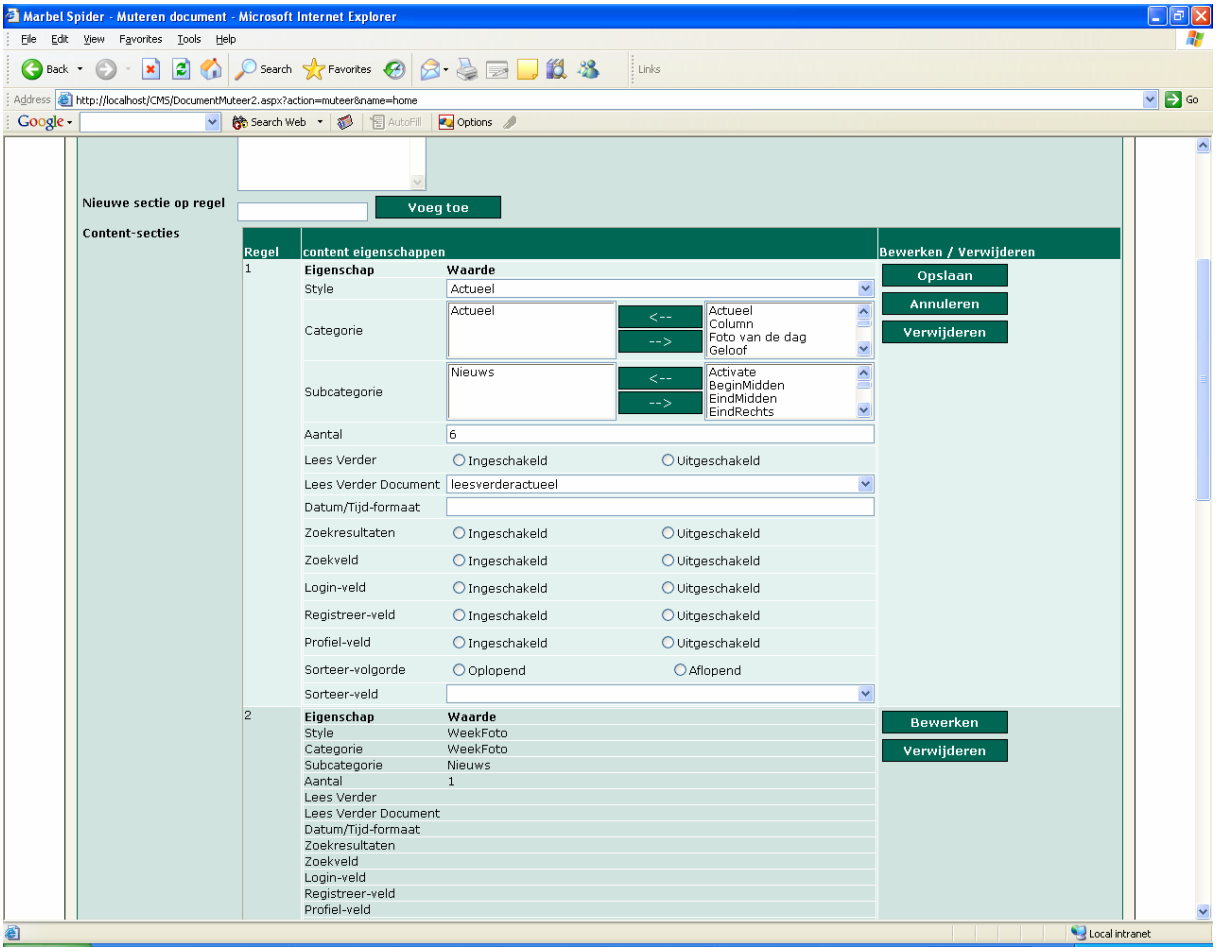
figuur 23: Datagrid gebonden aan een XmlNodeList-object van document-definitie

In het bovenstaande voorbeeld is te zien dat er 7 section-regels worden weergegeven. De regels bevatten echter geen bruikbare informatie. Standaard is de datagrid-control ingesteld met de optie AutoGenerateColumns. Hierdoor wordt aan de hand van de datasource bepaald welke kolommen er worden weergegeven (De velden in de uitvoer zijn eigenschappen van de XmlNode-objecten die samen het XmlNodeList-object vormen). Om de datagrid-control toch de juiste eigenschappen (namelijk de waarden van attributen van de content-elementen) te laten weergeven, moest ik de manier waarop de datagrid zich bond, verder specificeren. Dit heb ik bereikt door een klasse te schrijven die de ITemplate-interface implementeert.
De ITemplate-interface is onderdeel van het .NET-framework en wordt gebruikt door de datagrid, repeater en datalist-control om door middel van programmacode (in de code-behind) template-kolommen te specificeren. Een andere methode om template-kolommen te specificeren is via de GUI van Visual Studio. In dat geval worden de template-kolommen statisch gedefinieerd op de pagina. Voordeel hiervan is dat je meteen in Visual Studio ziet hoe de kolommen eruit komen te zien. Het nadeel van deze methode is dat het erg gecompliceerd is om de events die de datagrid genereert correct af te handelen. Daarom heb ik er uiteindelijk voor de methode gekozen waarbij de template-kolommen programmatisch worden gedefinieerd.
Een vereenvoudiging van de programmacode voor de template-kolommen die ik heb geschreven, heb ik opgenomen als externe bijlage (Bijlage H). Commentaar in de code lichten de verschillende stappen toe.

6.3. Ontwikkelen van de editor

Nu ik een ontwerp van het scherm gemaakt had en de technieken die ik ging gebruiken had geselecteerd en geprobeerd, kon ik beginnen met de daadwerkelijke ontwikkeling van de editor. Allereerst heb ik de broncode van de bestaande documenteditor (het grote tekstvlak) geopend in Visual Studio en heb ik de pagina opgeslagen onder een andere naam. In de kopie die zo ontstond ben ik gaan ontwikkelen. Op deze manier kon ik gebruik maken van de functionaliteit die al aanwezig was voor het ophalen en opslaan van de data. Toen heb ik de schermelementen op de pagina aangebracht zoals in het ontwerp en in het geval van de datagrid ervoor gezorgd dat deze werd gebonden aan de data. In de oorspronkelijke schets was ik vergeten om functionaliteit in te bouwen om nieuwe secties toe te voegen. Deze functionaliteit heb ik tijdens het ontwikkelen toegevoegd door middel van een tekstvlak met een knop ‘toevoegen’. Als de gebruiker op de knop ‘toevoegen’ drukt, wordt er een sectie boven in het document toegevoegd. In het tekstvlak kan de gebruiker eventueel

aangeven op welke regel de sectie wordt toegevoegd. Het volgende figuur is een schermafbeelding van de uiteindelijke documenteditor.



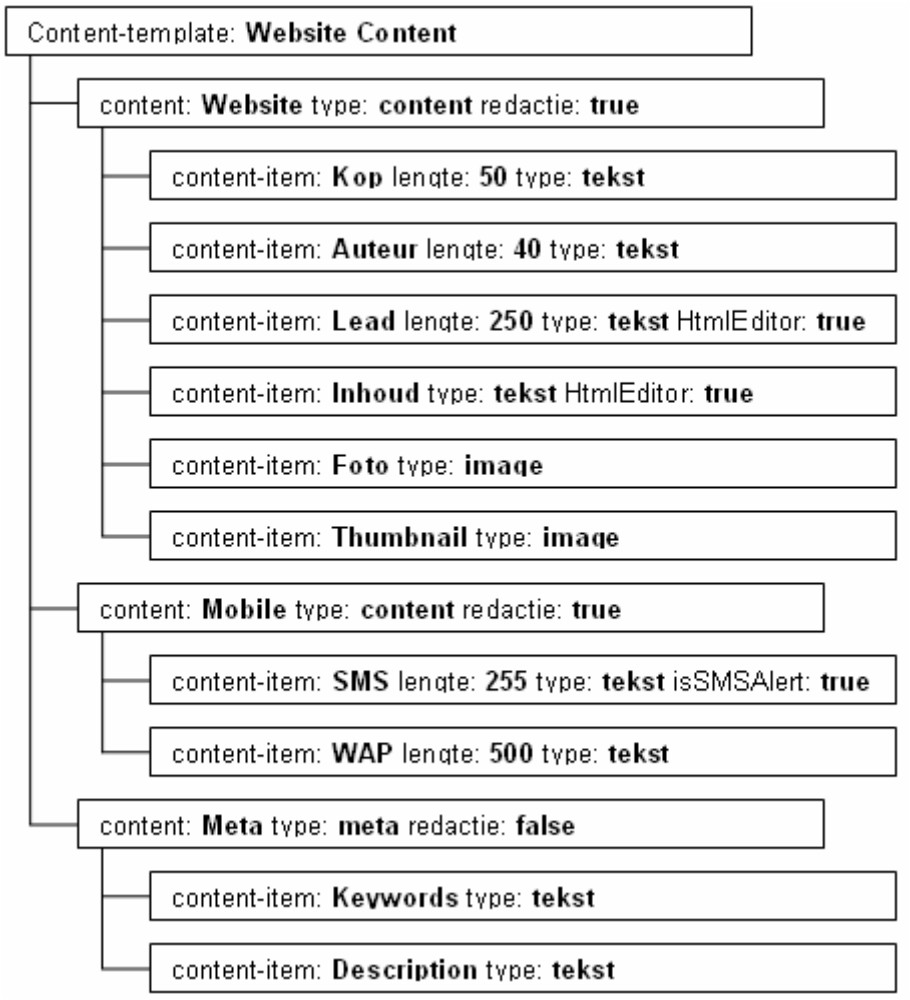
figuur 24: Schermafbeelding documenteditor

7. Ontwikkelen van de template-editor

In het vorige hoofdstuk heb ik beschreven hoe ik de eerste pilot, de documenteditor, heb ontwikkeld. De tweede pilot die ik moest ontwikkelen, was de template-editor. Door middel van de template-editor moeten gebruikers content-templates kunnen genereren zonder dat ze daarvoor XML hoeven te gebruiken. Bij de ontwikkeling van de template-editor heb ik precies dezelfde werkwijze gehanteerd als bij de document-editor: Eerst heb ik de structuur van het XML-formaat bestudeerd om te bepalen aan welke eisen de GUI moest voldoen. Vervolgens heb ik schetsen gemaakt en geëxperimenteerd met technieken waarmee het ontwerp mogelijk kon worden gerealiseerd. Daarna heb ik de meest geschikte techniek uitgekozen en ben ik gaan ontwikkelen. Dit hele proces beschrijf ik in dit hoofdstuk. Voor het pilotontwikkelplan van deze pilot verwijs ik u naar externe bijlage F.

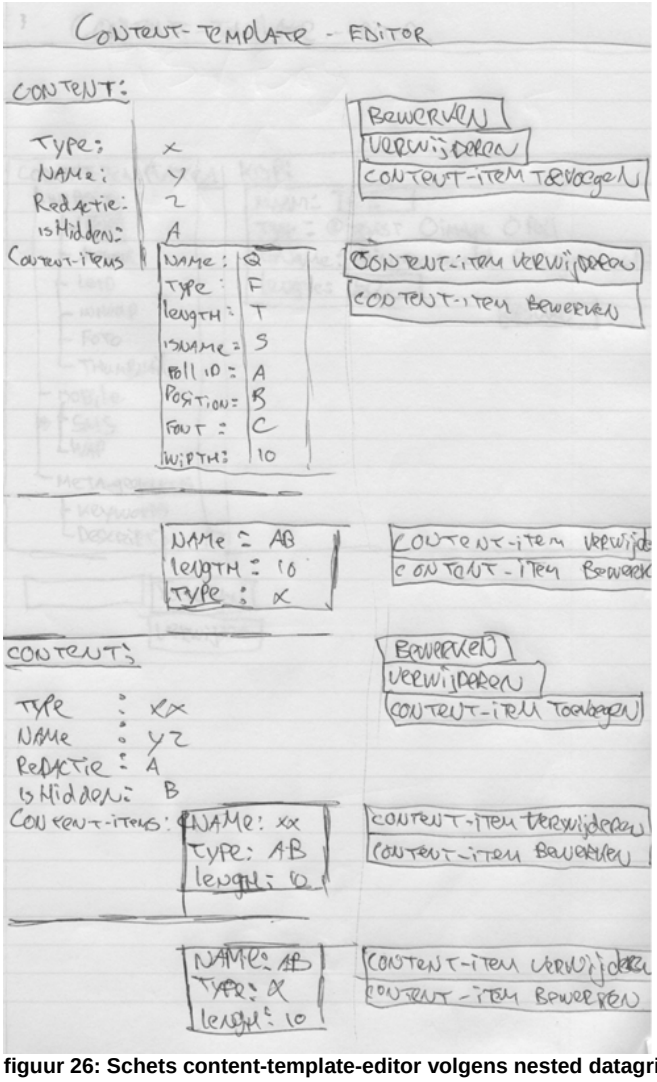
7.1. Ontwerpen van de GUI

Voor deze pilot heb ik hetzelfde mapping-principe gebruikt als bij de documenteditor.
Daarom moest ik weer eerst de XML-structuur van content-templates in kaart brengen. Bij de document-editor heb ik een fout gemaakt door eerst alleen te kijken naar xml-bestanden terwijl de exacte structuur beter kon worden achterhaald door de broncode te raadplegen. Daarom heb ik deze keer ook meteen naar de broncode gekeken. Daardoor kwam ik erachter dat een content-template uit meerdere soorten content bestaat die op hun beurt weer kunnen bestaan uit meerdere content-items. De structuur van een content-template heb ik hieronder in beeld gebracht door middel van een voorbeeld.



figuur 25: Schematische weergave content-template

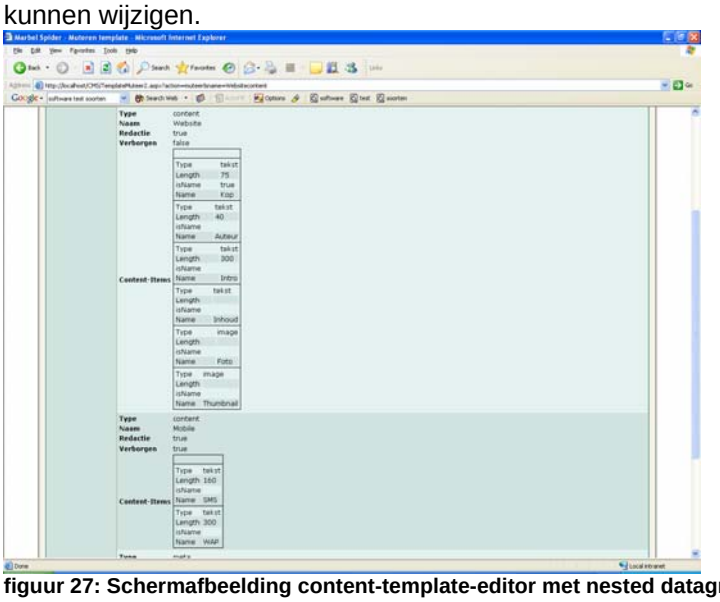
Zoals te zien is in figuur 25 hebben content-templates een totaal andere structuur dan de document-definities uit het vorige hoofdstuk. Daarbij waren er steeds terugkerende elementen op één niveau. Hier is sprake van een soort boomstructuur die bestaat uit een variabel aantal terugkerende elementen op twee niveaus: content-elementen en contentitem-elementen.
Deze boomstructuur leverde meteen een probleem op bij het ontwerpen van de GUI omdat ik de interface van de document-editor (met de datagrid) terug wilde laten komen in de template-editor. Toen ik begon te schetsen kwam ik er echter achter dat dit niet zou lukken, vanwege de twee niveaus. Toen ik op internet ging zoeken naar oplossingen op dit probleem kwam ik terecht bij een website waar het principe van de nested datagrid beschreven werd. Bij een nested datagrid wordt een datagrid binnen een andere datagrid geplaatst. Het voorbeeld ging uit van databinding met een database. De SQL-query die de datasource van de binnenste datagrid bepaalde, werd bepaald door variabelen uit de buitenste datagrid.
Ik bedacht me dat ditzelfde principe ook kon worden toegepast bij databinding aan een XmlNode-object. Wanneer een datagrid namelijk gebonden wordt aan een XmlNode worden de onderliggende XmlNodes (childnodes) weergegeven als regel in de datagrid. (zelfde principe als in figuur 23). Per regel van een datagrid kan het DataGridItem-object worden geïnstantieerd. Daaruit kan door middel van type-casting een XmlNode-object worden afgeleid. Dit XmlNode-object kan dan weer als datasource dienen voor een andere datagrid. Dit principe heb ik gebruikt in onderstaande schets.



figuur 26: Schets content-template-editor volgens nested datagrid principe

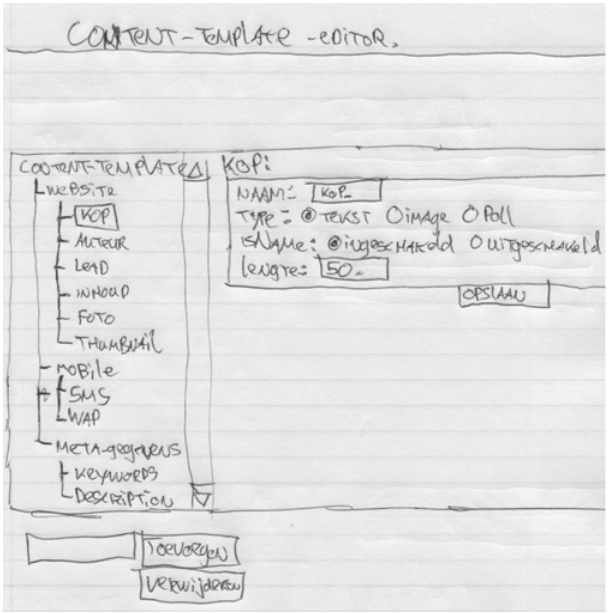
De eindresultaat geeft door de verschillende niveaus een onoverzichtelijke indruk. Ook de grote hoeveelheid knoppen kunnen het voor de gebruiker verwarrend maken. Omdat ik dacht dat ik deze problemen wel kon verhelpen door te zorgen voor duidelijke omschrijvingen en een overzichtelijke

opmaak, ben ik begonnen met het maken van een prototype van de template-editor volgens dit ontwerp. Ik slaagde erin om de nested datagrid te binden aan een content-template (figuur 27). De volgende uitdaging was het vinden van een mogelijkheid om de gegevens in de binnenste datagrid te kunnen wijzigen.



figuur 27: Schermafbeelding content-template-editor met nested datagrid

Het lukte me namelijk niet om de events van controls in de binnenste datagrids af te handelen. Dit is echter wel van belang omdat deze knoppen ervoor moeten zorgen dat een bepaalde regel van een van de binnenste datagrids in de edit-modus wordt weergegeven. Toen ik op internet zocht naar een oplossing voor dit probleem ontdekte ik dat ik niet de enige was die dit probleem had. Uiteindelijk stuitte ik op een artikel waarin de binnenste datagrid gebonden werd door middel van een user-control. Omdat ik hier nog nooit mee gewerkt had en ik me niet blind wilde staren op één oplossing, besloot ik het hele nested datagrid principe overboord te zetten. Een andere overweging om te zoeken naar een andere oplossing was dat er in het geval van de content-template uit figuur 25, welke in principe niet vreselijk uitgebreid is, wel 14 x 2 = 28 knoppen op het scherm zouden komen. Rekening houdend met de usability-richtlijnen uit figuur 18, zou dit regel 7 en 8 over de flexibiliteit en simplistisch ontwerpen natuurlijk niet ten goede komen.
Geïnspireerd door websites zoals MSDN, heb ik toen een tweede schets gemaakt met daarin aan de linkerkant een boomstructuur die de structuur van een content-template representeert.



figuur 28: Tweede schets content-template-editor: boomstructuur als uitgangspunt

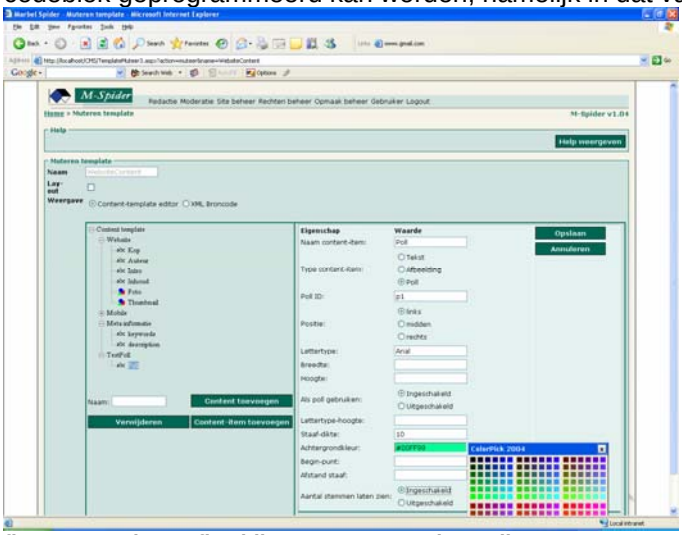
De boom is op dezelfde manier opgebouwd als in figuur 25. Behalve dat de boom een schematische voorstelling geeft van de content-template, dient het ook als navigatiemethode. Door te klikken op de content- of contentitem-elementen in de boom verschijnen in het rechtergedeelte van de editor de eigenschappen ervan. Ik heb hiervoor gekozen omdat op deze manier veel ruimte op de pagina wordt bespaard.

7.2. Kiezen van een geschikt component

Nadat ik het tweede ontwerp had gemaakt, moest ik ook hierbij weer op zoek naar een techniek om het ontwerp te verwezenlijken. De boomnavigatie is een methode die op internet veel toe wordt gepast in online bibliotheken (zoals MSDN) en andere naslagwerken. Windowsgebruikers kennen de boomstructuur uit Verkenner als methode om het bestandssysteem weer te geven. Op internet zijn tientallen treeview-componentent voor zo'n beetje alle client- en serverside scriptingtalen beschikbaar dus ook voor ASP.NET-platform moest wel zo'n treeview zijn ontwikkeld. Ik heb met verschillende producten geëxperimenteerd en ben tot de conclusie gekomen dat het prijskaartje niets zegt over de kwaliteit en flexibiliteit. Zo heb ik componenten geprobeerd die meer dan $ 200,- kosten. Soms kon ik deze echter niet gebruiken omdat het niet mogelijk was om ze door middel van programmacode op te bouwen. Deze componenten moesten dan vaak worden opgebouwd door middel van een Xml-bestand. Hier kon ik niets mee, omdat ik niet de beschikking had over fysieke Xml-bestanden maar slechts over 'in mermory'-Xml-fragmenten. Ik heb uiteindelijk gekozen voor de TreeView-control, onderdeel van de Internet Explorer Web Controls-set omdat hij gratis te downloaden is en op verschillende manieren kan worden geconfigureerd. Hij kan worden gebonden aan een xml-bestand dat aan een bepaalde indeling moet voldoen of hij kan programmatisch worden opgebouwd. Ik zelf wilde de TreeView-control programmatisch opbouwen omdat ik er zo het meest flexibel gebruik van kon maken. Nadat ik me in de eigenschappen en de methodes van het component had verdiept, heb ik een functie geschreven die de treeview-control opbouwt op basis van XML content-template. De programmacode met toelichting heb ik opgenomen in externe bijlage H.

7.3. Ontwikkelen van de editor

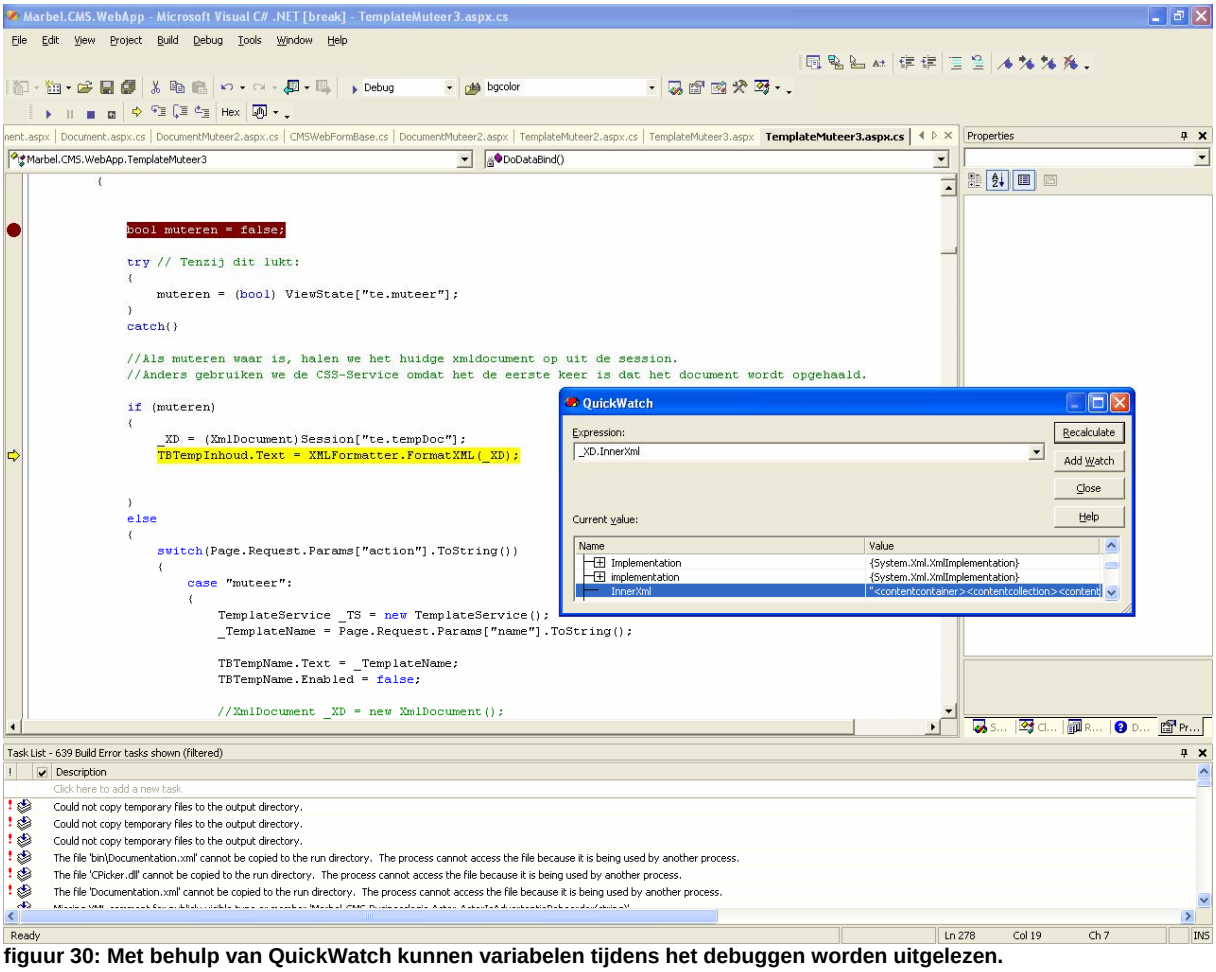
Om de content-template-editor verder te kunnen ontwikkelen, heb ik eerst uitgezocht welke attributen content- en contentitem-elementen kunnen hebben. Hiervoor heb ik de broncode van Spider (Businesslogic) geraadpleegd. Toen ik wist wat de exacte attributen van beide elementen waren, kon ik beginnen met ontwikkelen. De GUI maakt afgezien van de TreeView-control gebruik van principes die ook al gebruikt waren bij de document-editor (pilot 1). Hoewel er steeds maar één xml-element aan data bewerkt hoeft te worden (dankzij het gebruik van de TreeView-control voor selectie van dat element), heb ik ook hier een datagrid gebruikt. Het verschil met de document-editor is dus dat de content-template-editor maar 1 DataGridItem (regel) per keer hoeft te laten zien. De reden om toch een datagrid te gebruiken is dat het bijwerken van álle attributen van het content(item)-element in één codeblok geprogrammeerd kan worden, namelijk in dat van het DataGrid.UpdateCommand-event.



figuur 29: Schermafbeelding Content-template-editor

7.4. Oplossen van fouten tijdens het programmeren

Tijdens het programmeren ben ik me een mentaal beeld gaan vormen van hoe de programmeeromgeving werkt. Naarmate ik meer ervaring kreeg met een bepaalde taal en techniek, ging ik daar steeds meer op vertrouwen. Toch kwam het voor dat er fouten ontstonden op het moment dat de code werd uitgevoerd. Deze runtime-errors zijn vaak moeilijker op te lossen dan fouten die ontstaan tijdens het schrijven van de code, omdat de herkomst vaak moeilijk te achterhalen is. Ook kan het gebeuren dat de applicatie wel normaal compileert en geen runtime-error geeft, maar toch onverwacht gedrag vertoont.
Tijdens het ontwikkelen heb ik me meerdere malen blindgestaard op onverwacht gedrag totdat een collega mij erop wees om meer gebruik te maken van de debugger. Met behulp van de debugger kan je een programma bij een bepaalde programmaregel laten pauzeren en kan je alle variabelen die op dat moment gelden, uitlezen. Zo kun je stap voor stap de waarden van variabelen volgen en kun je precies zien of er iets mis gaat en hoe dit komt. Bij het programmeren maakte ik regelmatig fouten doordat ik een onjuist beeld had van de volgorde waarin opdrachten worden uitgevoerd bij het (her)laden van een asp.net-pagina. Met behulp van debugging heb ik dat beeld verhelderd.



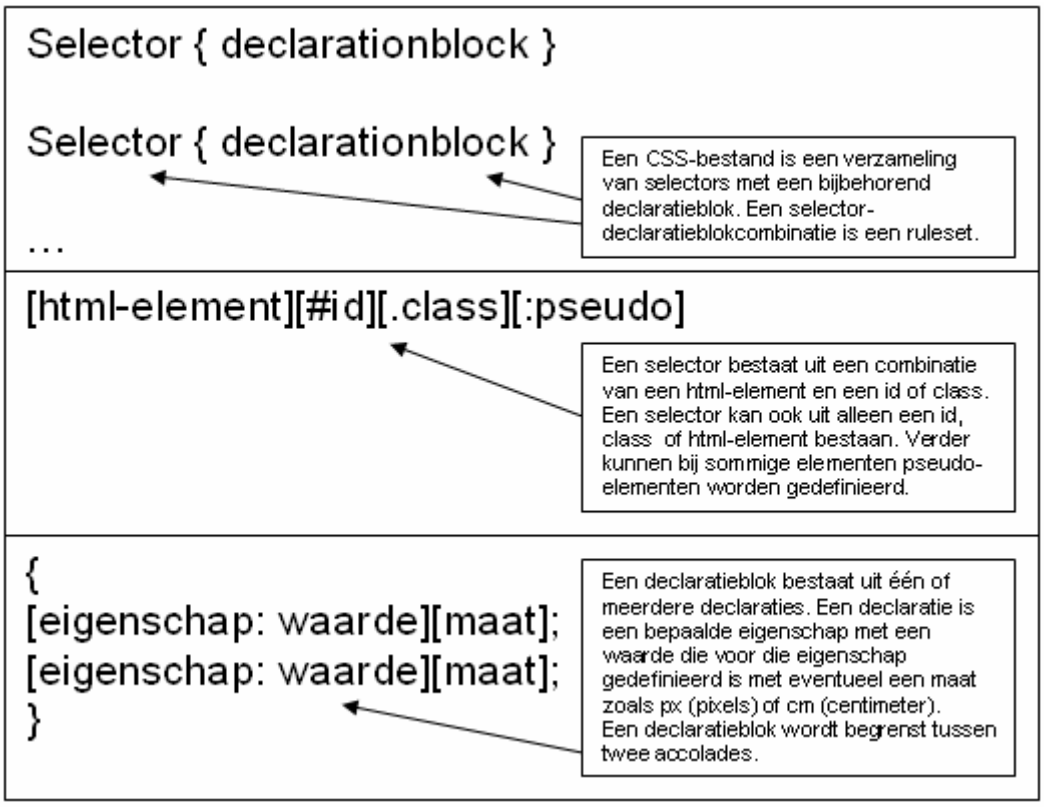
figuur 30: Met behulp van QuickWatch kunnen variabelen tijdens het debuggen worden uitgelezen.

8. Ontwikkelen van de CSS-editor

In de vorige twee hoofdstukken heeft u kunnen lezen hoe ik de document- en de template-editor heb ontwikkeld. De laatste pilot die ik moest ontwikkelen, was de CSS-editor. De totstandkoming daarvan beschrijf ik in dit hoofdstuk. Deze pilot verschilt ten opzichte van voorgaande pilots doordat CSS-bestanden platte tekst zijn en documentdefinities en content-templates XML-bestanden zijn.
In een vroeg stadium van mijn afstudeeropdracht had ik al eens met een collega gesproken over de mogelijkheid om CSS-bestanden op te slaan in een zelf te ontwerpen XML-formaat. Op deze manier zijn de bestanden veel gemakkelijker te muteren dan bij platte tekst. Dit komt vooral omdat voor XML het Document Object Model (DOM) is ontwikkeld. Hierdoor kan informatie in een XML document gemakkelijk worden gevonden en gemuteerd.
Bij de ontwikkeling van deze pilot heb ik weer ongeveer dezelfde werkwijze gehanteerd als bij de voorgaande pilots. Eerst heb ik de structuur van het formaat (CSS) in kaart gebracht en een XML-structuur voor het vastleggen van CSS bedacht. Vervolgens heb ik conversieroutines ontwikkeld om CSS naar XML te vertalen en andersom. Daarna heb ik de GUI voor de CSS-editor ontworpen en ontwikkeld. Het pilotontwikkelplan dat voor deze pilot is opgesteld, is opgenomen in de externe bijlagen (bijlage G)

8.1. Bepalen van de structuur van CSS en ontwikkelen XML-structuur

Voordat ik een XML-formaat kon ontwerpen voor het vastleggen van een CSS-bestand, heb ik eerst gekeken of het principe van het gestructureerd opslaan van CSS door middel van XML ergens al eerder was toegepast. Als dit het geval zou zijn, zou ik wellicht het concept kunnen overnemen zodat ik niet opnieuw het wiel zou hoeven uit te vinden. Na wat zoekacties op internet bleek dat dit toen niet het geval was. Het leek dus een zeer origineel idee te zijn. Voordat ik kon beginnen met het opstellen van het XML-formaat, moest ik me eerst verdiepen in de structuur van CSS omdat de editor immers bestanden met deze structuur moet kunnen genereren. Ik heb dit gedaan door op internet diverse websites te raadplegen waar de CSS-specificatie wordt uitgelegd. Aan de hand van de informatie die ik daar heb gevonden, heb ik toen voor mezelf een schematische weergave van een CSS document gemaakt. (figuur 31)



figuur 31: Vereenvoudigde structuur CSS

Bovenstaande syntax heb ik als uitgangspunt gehanteerd voor het opstellen van de XML-structuur. Bij het opstellen van deze XML-structuur ben ik als volgt te werk gegaan. De elementen die ik heb benoemd in figuur 31 heb ik in een leeg XML-document genoteerd in de vorm van XML-tags. De relaties tussen de verschillende CSS-elementen heb ik naar de XML-structuur vertaald door in het XML-document de genoteerde (lege) XML-tags in de juiste volgorde en in het juiste niveau te plaatsen. Toen ik hiermee klaar was, heb ik dit XML-document voorgelegd aan een van de ontwikkelaars van Spider. Ik twijfelde namelijk of ik geavanceerde CSS-functionaliteit zoals @-blocks (het aanroepen van een ander CSS-bestand vanuit een CSS-bestand), child-element selectors en adjacent sibling element-selectors (conditionele selector voor elementen op vergelijkbaar niveau) ook mee moest nemen in de XML-structuur. Nadat ik de structuur had voorgelegd, hebben we besloten om de geavanceerde functionaliteit niet mee te nemen omdat deze toch niet gebruikt zou gaan worden door de gebruikers van Spider. Bovendien zou een te complexe structuur ook betekenen dat de GUI van de editor ingewikkelder wordt. Het volgende XML-bestand is een voorbeeld van de uiteindelijke structuur van het CSS-XML-formaat.

```
<stylesheet>
   <rulesets>
      <ruleset>
         <selectors>
            <selector>
               <element>a</element>
               <class>menu</class>
               <id />
               <pseudo>hover</pseudo>
            </selector>
         </selectors>
         <declarationblock>
            <declaration>
               <property>font-family</property>
               <value>Verdana, Arial, Helvetica, sans-serif</value>
               <unit />
            </declaration>
            <declaration>
               <property>text-decoration</property>
               <value>none</value>
               <unit />
            </declaration>
         </declarationblock>
      </ruleset>
   </rulesets>
</stylesheet>
```

figuur 32: Voorbeeld van een ruleset in het CSS-XML-bestand

8.2. Ontwikkelen conversieroutines

Nu de structuur van CSS in kaart was gebracht en het XML-equivalent daarvan was ontwikkeld, moest ik eenmalig twee routines maken om tussen beide formaten te kunnen converteren. Binnen de CSS-editor moet namelijk net als bij de voorgaande pilots de mogelijkheid komen om de stylesheets aan te passen zonder de GUI te hoeven gebruiken. Dit is om te kunnen voldoen aan richtlijn 7 uit figuur 18. Daarin staat dat ervaren gebruikers de mogelijkheid moeten hebben om bepaalde stappen over te slaan. Gebruikers die ervaring hebben met CSS kunnen op deze manier gewoon de stylesheet zelf invullen. En wanneer ze toch een bepaalde eigenschap niet weten, kunnen ze op elk gewenst moment overschakelen naar de GUI. Een andere reden waarom de conversieroutines moesten worden ontwikkeld, is omdat de stylesheets uiteindelijk toch als CSS doorgegeven moeten worden aan de webbrowser. Bij het ontwikkelen van deze routines heb ik twee verschillende technieken gebruikt. Het converteren van CSS naar XML gebeurt door middel van het parsen van tekst. De conversie van XML naar CSS wordt verzorgd door middel van XSL-T. Ik heb hiervoor gekozen omdat deze techniek sneller werkt dan het parsen van XML met het XML-DOM. XSL-T is alleen te gebruiken voor conversie met XML als bronformaat. Ik zal de werking van beide routines kort beschrijven. Programmacode van beide routines is opgenomen in externe bijlage H.

8.2.1. Van CSS naar XML

Voor deze routine is de invoer een string met daarin het CSS-document. Uitvoer van deze routine is een string van het XML-bestand. Uit het CSS-document moest ik eerst het commentaar verwijderen om te voorkomen dat er problemen zouden ontstaan tijdens het parsen. Toen ik op internet een zocht naar een functie waarmee dit bewerkstelligd kon worden, kwam ik op een website over 'regular expressions' terecht. Dit is een techniek om te zoeken naar complexe patronen in tekst zodat daar bewerkingen mee uitgevoerd kunnen worden. Op de website stond een RegEx-expressie voor het zoeken naar commentaar in diverse programmeertalen. Ik kwam erachter dat het .NET-framework ook uit is gerust met een RegEx-engine. Die heb ik toen gebruikt voor het verwijderen van commentaar (zie figuur 33) met de opbouw */* commentaar */*.

```
cssstring = System.Text.RegularExpressions.Regex.Replace(cssstring, "/\\*.*?\\*/", "");
```

figuur 33: Commentaar verwijderen d.m.v. regular expressions

Vervolgens heb ik een loop gemaakt om voor iedere ruleset (het grootst mogelijke terugkerende element binnen een CSS-bestand, zie figuur 32) in een CSS-bestand een aantal acties uit te voeren. Ik kwam er later achter dat deze aanpak top-down-parsing wordt genoemd. De loop heb ik gemaakt door de CSS-string op te splitsen bij iedere keer dat er een accolade '}' werd gevonden. Het resultaat is een string-array met in elke string een ruleset. Vervolgens heb ik op dezelfde manier in iedere loop een nieuwe loop gemaakt om de verschillende declaraties te vinden. Dit gebeurde op basis van de puntcomma. Elke declaratie werd gesplitst in een eigenschap en een waarde door middel van de dubbele punt. Op deze manier werd bij iedere keer dat er werd geloopt of gesplitst het XML-document verder opgebouwd en de CSS-string in kleinere stukken gestript.

8.2.2. Van XML naar CSS

Zoals eerder vermeld heb ik XSL-T voor deze routine gebruikt omdat deze techniek veel sneller is dan het programmatisch parsen van XML. Het transformeren van XML naar CSS gebeurt doormiddel van een XSL-T document. Dit document is opgebouwd uit een aantal templates die worden toegepast als een opgegeven gedeelte in het XML-DOM (opgegeven met een XPath-expressie) wordt gevonden. Hierdoor is het vrij eenvoudige en overzichtelijke techniek om te gebruiken. Ter illustratie: Deze routine neemt in totaal (de programmacode om de transformatie uit te voeren en het XSL-T document) ongeveer 35 regels in beslag tegenover zo'n 150 regels bij de CSS naar XML conversie. Het XSL-T document is te zien in figuur 34.

```
<?xml version="1.0"?>
   <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
   <xsl:output method="text" indent="no"/>
   <xsl:template match="rulesets">
      <xsl:apply-templates select="ruleset"/>
   </xsl:template>
   <xsl:template match="selector">
      <xsl:if test="element != ''"><xsl:value-of select="element"/></xsl:if>
      <xsl:if test="class != ''">.<xsl:value-of select="class"/></xsl:if>
      <xsl:if test="id != ''">#<xsl:value-of select="id"/></xsl:if>
      <xsl:if test="pseudo != ''">:<xsl:value-of select="pseudo"/></xsl:if>
      <xsl:if test="position() &lt; last()">, </xsl:if>
   </xsl:template>
   <xsl:template match="declarationblock"> {<xsl:text>
      </xsl:text>
      <xsl:apply-templates select="declaration"/><xsl:text>}
      </xsl:text>
   </xsl:template>
   <xsl:template match="declaration">
      <xsl:if test="property"><xsl:value-of select="property"/></xsl:if>
      <xsl:if test="value"><xsl:if test="unit">: <xsl:value-of
      select="value"/><xsl:value-of select="unit"/>;<xsl:text>
      </xsl:text>
      </xsl:if></xsl:if>
      <xsl:if test="value"><xsl:if test="boolean(unit) = false">: <xsl:value-of
      select="value"/>;<xsl:text>
      </xsl:text>
      </xsl:if></xsl:if></xsl:template></xsl:stylesheet>
```

Als deze attributen niet leeg zijn, geef de waardes dan weer met het corresponderende symbool ervoor.

Een declaratieblok begint en eindigt met een accolade.

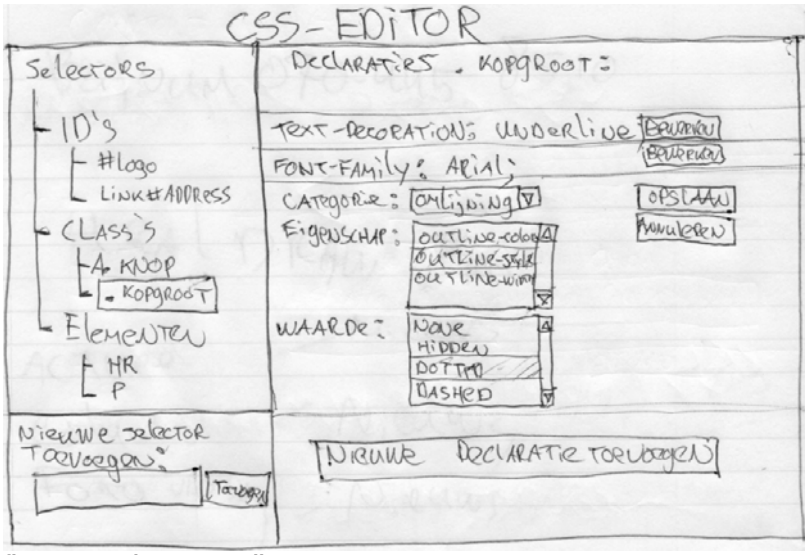
daartussen de declaration-template toepassen.

Declaratieregels opbouwen, indien unit is opgegeven, weergeven.

figuur 34: XSL-T document voor tranformatie XML naar CSS

8.3. Ontwerpen van de GUI

In de voorgaande twee paragrafen heb ik uitgelegd hoe ik de structuur van CSS en van het CSS-XML-formaat heb bepaald. Daarna heb ik conversieroutines gemaakt om over te kunnen schakelen tussen beide formaten. Nu kon ik een ontwerp maken van de GUI van de CSS-editor.

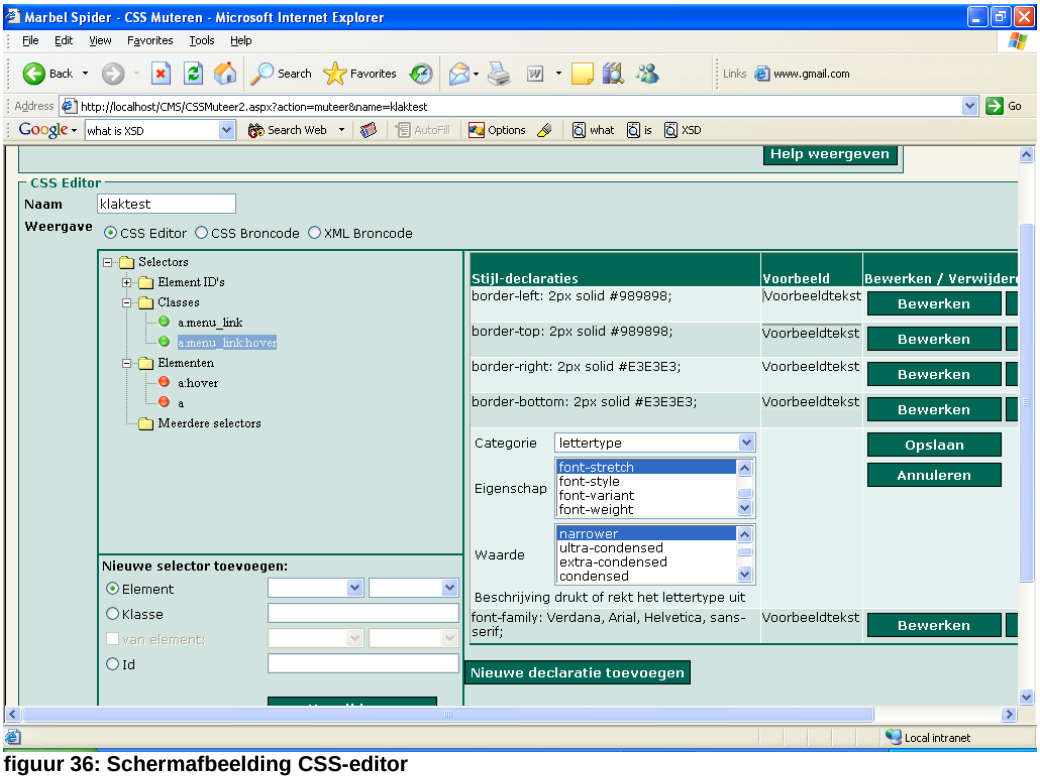


figuur 35: Schets CSS-editor

Veel elementen in de bovenstaande schets zijn afgeleid van de document- en template-editor. Voor de selectie van selectoren heb ik net als bij de template-editor een TreeView-control gebruikt. Hierin kunnen de verschillende selectoren gesorteerd worden weergegeven waardoor de gebruiker ze gemakkelijk kan terugvinden. Het muteren van het declaratieblok gebeurt weer door middel van een datagrid omdat daarmee de XML-data gemakkelijk gemuteerd kan worden. Elke regel van de datagrid stelt een combinatie van een eigenschap, waarde en maat voor. Voor het kiezen uit de lijst met categorieën wordt een dropdownlistbox gebruikt omdat er sprake is van een beperkt aantal items waaruit 1 item moet kunnen worden gekozen. Als de categorie gekozen is, verschijnen daaronder de bijbehorende eigenschappen. Die kan worden gekozen door middel van een listbox. De eigenschappen binnen een bepaalde categorie zijn zo voor een lange tijd gedeeltelijk tegelijkertijd in beeld. Hierdoor heeft de gebruiker zo veel mogelijk tijd om te kiezen en kan hij gemakkelijk zijn keuze weer herzien. Ditzelfde geldt voor het kiezen van een waarde. Ook hier heb ik een listbox voor gebruikt. Niet alle waarden kunnen echter gekozen worden door middel van een listbox omdat de mogelijkheden niet bij elke eigenschap vastliggen. Waarden waarbij kleuren of afstanden moeten worden gespecificeerd, gebruiken andere schermelementen. Voor het kiezen van kleuren heb ik een ColorPickercontrol gebruikt. Dit is een extern gratis beschikbaar component waarmee eenvoudig een kleur kan worden geselecteerd. Het kiezen van lengten gebeurt met een tekstbox en een dropdownlistbox. In de tekstbox kunnen numerieke waarden worden ingevoerd en in de dropdownlistbox kan een maat worden geselecteerd. Op deze manier vang ik een eventuele foutieve invoer van de gebruiker af.

8.4. Ontwikkelen van de editor

Zoals hierboven vermeld, heb ik alle CSS-eigenschappen verdeeld in categorieën. Hierdoor hoeven de gebruikers niet door een lijst van meer dan 100 eigenschappen te scrollen. De categorieën met bijbehorende eigenschappen en mogelijke waarden heb ik vastgelegd in een XML-bestand. Op deze manier kunnen er gemakkelijk eigenschappen worden toegevoegd in het geval van een nieuwe versie van CSS. Het XML-bestand bevat tevens een beschrijving van elke CSS-eigenschap. Hierdoor kon ik gemakkelijk helpfunctionaliteit integreren in de editor. Het XML-bestand is tot stand gekomen door de CSS-reference van de website w3schools.net om te zetten naar een eigen XML-formaat.



figuur 36: Schermafbeelding CSS-editor

De eigenlijke ontwikkeling van de CSS-editor heeft op dezelfde manier plaatsgevonden als bij de voorgaande pilots. Eerst heb ik de schermelementen op de pagina geplaatst. Vervolgens heb ik de acties geprogrammeerd die moesten worden uitgevoerd bij de events van de verschillende schermelementen.

9. Testen, acceptatie en invoering van de pilots

In de vorige drie hoofdstukken heeft u kunnen lezen hoe de ontwikkeling van de verschillende pilots heeft plaatsgevonden. Tijdens de ontwikkeling hebben er bij alle pilots whitebox-testen plaatsgevonden. Toen de pilots ontwikkeld waren, heb ik vervolgens nog een functionele test uitgevoerd om te verifiëren dat het systeem aan de vastgestelde specificaties voldeed. Het gebruik van beide testsoorten in dit project wordt toegelicht in de eerste twee paragrafen van dit hoofdstuk. In de derde paragraaf kunt u lezen hoe de acceptatie bij dit project tot stand is gekomen. Tot slot beschrijft de laatste paragraaf het proces van invoering van de pilots. De uitkomsten van de functionele test en acceptatie vindt u op het 'test- en acceptatieformulier', opgenomen als externe bijlage bij dit verslag (bijlage I). Dit formulier heb ik opgesteld om de activiteiten die bij reguliere IAD-trajecten tijdens de 'beoordeel- en testworkshop' en de 'acceptatieworkshop' aan bod komen formeel te kunnen vastleggen.

9.1. Whitebox testen tijdens de ontwikkeling

Tijdens de ontwikkeling van de verschillende pilots heb ik volgens de whitebox-techniek getest. Dit hield in mijn geval in dat ik alle functionaliteit per pilot structureel heb getest. In de praktijk kwam het erop neer dat ik bij het ontwikkelen een lijstje bijhield met punten die nog ontwikkeld moesten worden. Ik streepte deze punten pas af als ik ze onder alle omstandigheden had getest. Deze manier van testen heb ik aangehouden vanwege mijn natuurlijke werkwijze: Omdat ik nog niet veel ervaring had met programmeren, zag ik elk ontwikkeld detail als een kleine overwinning. Omdat ik graag zeker wilde zijn van het resultaat, heb ik al deze details steeds uitvoerig getest. Een voordeel van deze manier van testen is dat je de applicatie structureel test en dus nooit een onderdeel zal vergeten. Een nadeel van whitebox-testen zou kunnen zijn dat fouten die betrekking hebben op de relatie tussen verschillende componenten onopgemerkt zouden kunnen blijven. Op zich gaat dit nadeel maar voor een klein deel op bij dit project aangezien het om GUI's gaat die elk slechts één pagina bestrijken. De relatie tussen de verschillende componenten is hierdoor goed te overzien. Uit al deze verschillende testen zijn geen specifieke producten voortgekomen zoals een testplan of testrapport. Het enige tastbare wat ik aan deze testen heb overgehouden, zijn wat krabbeltjes en aantekeningen op mijn kladblok. Mijn motivatie om geen vaste testmethodiek voor whitebox-testen te hanteren is dat ik het ontwikkelproces zo flexibel mogelijk wilde houden.
Om te verifiëren of de ontwikkelde pilots voldeden aan de gestelde systeemeisen heb ik na de ontwikkeling een functionele test gedaan. Hierbij heb ik de verschillende pilots in zijn geheel getest. Deze test wordt beschreven in de volgende paragraaf.

9.2. Functioneel testen na de ontwikkeling

Zoals hierboven vermeld heb ik naast de structurele tests tijdens de ontwikkeling, na de ontwikkeling ook nog getest. Het doel van deze functionele test was voor mij om na te gaan of de verschillende pilots in zijn geheel voldeden aan de systeemeisen. De systeemeisen zijn onderverdeeld in 5 categorieën: functionele- of basissysteemeisen, interface-eisen, integriteitzeisen, performance-eisen en operationele eisen. Normaal gesproken worden bij functionele testen (ook wel blackbox testen genoemd) alleen de functionele eisen (hier: basissysteemeisen) getest. Voor dit project waren er echter maar drie basissysteemeisen gesteld. Daarbij resulteerde elke systeemeis in een pilot. (zie figuur 37)

BS.01	*Gebruikers (gebruikersgroep site-beheerders) moeten wijzingen kunnen aanbrengen in de opmaak van een website zonder dat ze kennis hoeven te hebben van CSS.*
BS.02	*Gebruikers (gebruikersgroep site-beheerders) moeten nieuwe pagina's kunnen toevoegen aan een website zonder dat ze kennis hoeven te hebben van XML dat gebruikt wordt binnen Spider om pagina's te definiëren.*
BS.03	*Gebruikers (gebruikersgroep site-beheerders) moeten content-templates kunnen toevoegen aan een website zonder dat ze kennis te hoeven hebben van XML dat gebruikt wordt om nieuwe templates te definiëren binnen Spider.*

figuur 37: Basissysteemeisen uit definitiestudie

Als ik deze eisen puur inhoudelijk zou moeten verifiëren door middel van blackbox-testen zijn er maar 3 aspecten om op te testen. Namelijk of er nog CSS of XML gebruikt moet worden bij het gebruik van de ontwikkelde editors. Hoewel dit feitelijk wel geverifieerd zou kunnen worden (bv: een CSS-editor voldoet als het geldige CSS kan genereren), vind ik ze achteraf wat kort door de bocht. Daarom heb ik besloten om alle systeemeisen mee te nemen in de test. Daarvoor heb ik een formulier gemaakt waarbij elke eis geverifieerd en afgetekend moest worden. Eventuele opmerkingen konden hierbij ook per systeemeis genoteerd worden.
Voor de eigenlijke test heb ik geen specifieke testscenario's opgesteld. De tactiek die ik bij het testen van iedere pilot gehanteerd heb, was het maken van een CSS-, template- en documentdefinitie waarbij alle mogelijke functionaliteit van de editor gebruikt werd. Ik heb hiervoor gekozen omdat op deze manier de tests zo volledig mogelijk zouden zijn en omdat ik in dit stadium van de afstudeeropdracht niet veel tijd meer over had om extra documentatie zoals een gedetailleerd testplan en testrapport te schrijven, hoewel dit wel gepland stond.. Nadat ik via het test- en acceptatieformulier voor elke systeemeis had nagegaan of de verschillende pilots eraan voldeden, kon de opdrachtgever (in het geval van mijn opdracht was dat mijn bedrijfsmentor, Gert-Jan Scheeres) overgaan tot een definitief acceptatieoordeel.

9.3. Acceptatie van de pilots

IAD verstaat onder acceptatie het aanvaarden van een eindresultaat door de opdrachtgever. Dit aanvaarden gebeurt op basis van de acceptatiecriteria die zijn opgesteld in de definitiestudie. De volgende criteria heb ik in die fase opgesteld:

- Komt de functionaliteit van de pilot overeen met de systeemeisen? (per pilot)
- Is voldaan aan de integriteits- en interface-eisen? (per pilot)
- Is de systeemdocumentatie op orde? (per pilot)
- Kan het systeem worden beheerd door Marbel Software?

De eerste twee punten waren vrij concreet. Deze heb ik geverifieerd in de functionele test die beschreven is in de vorige paragraaf. De overige punten zijn minder concreet. Ik heb het test- en acceptatieformulier aan de opdrachtgever voorgelegd zodat hij hierover kon oordelen. Nadat de opdrachtgever de verschillende pilots had beoordeeld, kon worden overgaan tot invoering.

9.4. Invoering van het systeem

Nu de pilots formeel waren beoordeeld door de opdrachtgever, kon de invoering plaatsvinden. Volgens IAD is het doel van de activiteit invoering het installeren en invoeren van de pilot in de breedste zin van het woord. Omdat het in het geval van mijn ontwikkeltraject slechts ging om het aanpassen van de GUI's van het systeem, was de procedure om het systeem in te voeren niet erg complex: De enige actie die genomen moest worden, was het overplaatsen van de door mij aangepaste GUI's vanuit de ontwikkelomgeving die was ingericht speciaal voor mijn afstudeerproject naar de normale ontwikkelomgeving van Spider. Wanneer Marbel Software dan in de toekomst een nieuwe release van Spider aanbiedt aan de verschillende klanten, worden de door mij ontwikkelde editors automatisch meegenomen. Dit heb ik helaas niet meer mee kunnen maken tijdens mijn afstudeerperiode.

10. Evaluatie

In de voorgaande hoofdstukken heeft u kunnen lezen hoe mijn afstudeerproject is verlopen, welke keuzes ik heb gemaakt en wat mijn motivatie voor die keuzes was. In dit hoofdstuk zal ik afstudeerperiode evalueren. Dat doe ik aan de hand van twee paragrafen. In de eerste paragraaf evalueer ik het proces dat ik gevolgd heb bij het uitvoeren van de afstudeeropdracht. In de tweede paragraaf evalueer ik de opgeleverde producten die voortgekomen zijn uit dat proces.

10.1. Procesevaluatie

In deze paragraaf evalueer ik het proces dat ik doorlopen heb. Daarbij gebruik ik de onderverdeling in fasen zoals ik die in het plan van aanpak heb gedefinieerd:

- Fase oriëntatie
- Fase definitiestudie
- Fase pilotontwikkeling
- Fase invoering

Fase oriëntatie

De fase oriëntatie besloeg de periode van een maand voor aanvang van afstudeerperiode tot en met het eind van de eerste afstudeerweek. In deze periode heb ik me bezig gehouden met het opstellen van de opdrachtomschrijving en het plan van aanpak. Op zich ben ik tevreden met het verloop van deze fase. Ik wist wat er van me verwacht werd ten aanzien van inlevermomenten op school en heb de gestelde deadlines kunnen halen. Als ik deze fase over zou mogen doen, zou ik wel een aantal zaken anders aanpakken. Ik zou dan bijvoorbeeld meer tijd hebben gereserveerd voor het opstellen van het plan van aanpak omdat hierbij misschien wel de belangrijkste keuzes van de hele afstudeerperiode moesten worden gemaakt. Hierbij doel ik onder andere op het kiezen van een ontwikkelmethodiek en het maken van een detailplanning. Als ik toen de ervaring bezat die ik tijdens het afstuderen heb opgedaan, zou ik er misschien voor hebben gekozen om geen specifieke ontwikkelmethodiek te gebruiken maar om zelf een methodiek op te stellen waarin ik verschillende technieken aan elkaar koppel. Hierdoor is de kans groot dat de methodiek beter bij het proces past. Tijdens het doorlopen van dat proces stuit je dan veel minder op vraagstukken over hoe je een bepaalde activiteit moet inpassen binnen je eigen opdracht. Aan de andere kant zou het zelf verzinnen van een methodiek ook kunnen betekenen dat je veel keuzes die je moet maken tijdens het proces overslaat omdat je die als vanzelfsprekend ziet.
Wat betreft het opstellen van de detailplanning had ik zaken ook anders aangepakt als ik het over mocht doen. Ik heb nu in week 1 een detailplanning gemaakt waarbij de activiteiten op dagniveau gepland waren. De praktijk wees uit dat het hanteren van deze planning een onbegonnen taak was. Ik ben hier dan ook vaak van afgeweken. De grote lijnen van deze planning heb ik wel gewoon gevolgd. Op deze twee punten na ben ik tevreden met het verloop van deze eerste fase.

Fase definitiestudie

Op zich ben ik wel tevreden over het verloop van deze tweede fase van mijn afstudeerproject. Er zijn wel een aantal zaken die ik anders zou hebben aangepakt als ik de opdracht opnieuw zou moeten doen. Zo zou ik het opstellen van de systeemeisen zou ik anders aangepakt hebben. Daar kwam ik achter bij het ontwikkelen en testen. Het probleem zit hem er vooral in dat ik in de categorie basissysteemeisen maar 3 eisen gedefinieerd had. Deze 3 eisen stonden ongeveer gelijk aan de doelstelling van de afstudeeropdracht. Tijdens het ontwikkelen had ik hier nog niet zoveel last van omdat het een solo-opdracht was en ik toch wel precies wist aan welke eisen het systeem moest voldoen. Een probleem deed zich hierbij echter meer voor bij het verifiëren van deze systeemeisen. Het is natuurlijk raar om te zeggen dat een bepaalde eis behaald is door te zeggen dat er een pilot ontwikkeld is die daarin voorziet. Dit was echter wel het geval bij mijn systeemeisen. Een ander punt dat ik anders zou hebben aangepakt met betrekking tot de systeemeisen is dat de interface-eisen wat beknopt zijn. Als ik hier meer (of alle) generieke usability-regels had genoteerd, zoals de heuristics van Jakob Nielsen of de designregels van Donald Norman, had ik achteraf beter kunnen verantwoorden waarom ik een bepaalde keuze gemaakt heb. Ik had dan simpelweg kunnen verwijzen naar de systeemeisen. Als je dit soort zaken niet in de systeemeisen opneemt, verricht je uiteindelijk

meer werk dan dat je formeel vooraf hebt afgesproken. Kort samengevat: Achteraf gezien zou ik meer zaken in de systeemeisen opnemen die ik als vanzelfsprekend zie maar anderen misschien niet. Een andere activiteit in de fase definitiestudie die ik achteraf anders aan had willen pakken is het bepalen van het systeemconcept. Daarbij heb ik taakdiagrammen van de gewenste situatie gemaakt. Deze taakdiagrammen voegen achteraf weinig aan waarde toe, omdat het ontwerpen van de gebruikersinterfaces pas in een later stadium aan bod kwam. Gelukkig heb ik hier tijdens de ontwikkeling van de pilots niet te stug aan vast gehouden. Het opstellen van het pilotplan was een activiteit die ik meerdere malen overnieuw heb gedaan omdat ik steeds tot andere inzichten kwam. In de eerste versie had ik de pilots verkeerd opgesplitst in delen. Hier kwam ik achter toen ik al in de fase pilotontwikkeling zat. In een tweede versie van het rapport definitiestudie heb ik deze indeling aangepast. Later heb ik ook nog de planning van deze pilots aangepast zodat de gehele pilot 'style-editor' de minste prioriteit kreeg. Het steeds weer aanpassen van deze zaken heb ik als zeer nuttig ervaren. Hierdoor heb ik voorkomen dat ik me in allerlei bochten zou wringen om maar vast te houden aan datgene wat ik van te voren vast heb gelegd. Bovendien onderkent ook IAD dat het pilotplan binnen een ontwikkelingstraject sterk aan verandering onderhevig kan zijn.

Fase pilotontwikkeling

Als ik terugkijk op de fase pilotontwikkeling ben ik tevreden met mijn algehele aanpak. Per pilot bestond het eerste deel van deze fase steeds uit het opstellen van het pilotontwikkelplan. Daarbij heb ik besloten om naar eigen inzicht te bepalen welke aspecten aan bod moesten komen. In het pilotontwikkelplan van de template-editor bijvoorbeeld heb ik een XSD-document gebruikt om uit te beelden wat het exacte formaat is voor document-templates. Als ik hier een UML-techniek voor had gebruikt, had ik het nooit zo precies kunnen doen. Daar ben ik dus wel tevreden mee. Verder ben ik tevreden over mijn keuze om het pilotontwikkelplan niet te gedetailleerd uit te werken. Bij grote systeemontwikkelingtrajecten waarbij een groot aantal ontwikkelaars aan hetzelfde systeem werkt, is een tot in detail uitgewerkt pilotontwikkelplan met uitgebreide objectmodellen natuurlijk een vereiste om ervoor te zorgen dat de verschillende pilots (en pilotdelen) tijdens de integratie goed 'in elkaar passen'. Ik was in dit traject de enige ontwikkelaar en vanwege mijn geringe ervaring op het gebied van programmeren wist ik na het opstellen van het pilotontwikkelplan lang niet altijd hoe ik bepaalde technische kwesties op moest lossen. Door in het pilotontwikkelplan bepaalde onderdelen 'open' te laten, kreeg ik meer vrijheid bij het ontwikkelen, wat naar mijn mening tot een beter eindresultaat heeft geleid.
Als ik terugkijk naar het eigenlijke programmeren, moet ik toegeven dat er een hoop dingen zijn die ik als ik het over mocht doen anders aangepakt zou hebben. Een eenvoudig voorbeeld is dat ik in het begin veel gebruik maakte van een grote rij met if-statements met dezelfde testwaarde. Later kwam ik erachter dat ik hiervoor beter een switch-statement kon gebruiken. Wat ik wel meteen vanaf het begin goed aangepakt heb, zo vind ik, is het maken van functies als ik op meerdere plaatsen gebruik moest maken van dezelfde functionaliteit. Dit heeft de onderhoudbaarheid van de code die ik geschreven heb aanzienlijk verhoogd. Het eerste voorbeeld geeft al aan dat ik een behoorlijke ontwikkeling heb meegemaakt tijdens het ontwikkelen. Vanuit de tactiek van 'drie regeltjes code schrijven en afwachten of het werkt' ben ik gegroeid naar een werkwijze waarbij ik steeds zekerder werd van wat ik deed en zo hele codeblokken kon schrijven zonder dat ik geneigd was om steeds maar halve functionaliteit te testen.
Ik heb gekozen voor deze (voor mijn doen) technische afstudeeropdracht omdat ik het een uitdaging vond om in aanraking te komen met allerlei nieuwe technieken en mogelijkheden om zo mijn brede ICT-kennis wat te verdiepen. Terugkijkend op de afgelopen 4 maanden kan ik zeggen dat ik daar zeker in geslaagd ben.

Fase invoering

De fase invoering besloeg volgens mijn planning de weken 13 tot en met 15. Het schrijven van mijn afstudeerverslag had ik gepland in de weken 16 tot en met 18. De werkelijkheid verliep echter anders. Doordat de invoering nog eenvoudiger was dan ik had verwacht, heb ik besloten om het schrijven van mijn afstudeerverslag meer prioriteit te geven. Dit is achteraf een goede keuze geweest aangezien ik meer moeite had met het schrijven van mijn afstudeerverslag dan ik had verwacht. De daadwerkelijke invoering van het systeem heeft nog niet plaatsgevonden op het moment dat ik dit schrijf maar alle benodigde acties zijn al wel verricht zodat het gemakkelijk ingevoerd kan worden. Daarom ben ik uiteindelijk toch tevreden met het verloop van deze fase.

10.2. Productevaluatie

In deze paragraaf evalueer ik de producten die ik opgeleverd heb tijdens dit afstudeerproject in chronologische volgorde. De volgende producten zijn opgeleverd.

- Plan van aanpak
- Rapport definitiestudie
- Pilotontwikkelplannen
- Pilot 1: documenteditor
- Pilot 2: template-editor
- Pilot 3: CSS-editor
- Test- en acceptatieformulier

Plan van aanpak

Mijn algehele oordeel over het plan van aanpak is dat ik het een nuttig document vind waar ik veel aan heb gehad tijdens de opdracht. Uiteraard zijn er wel een aantal punten die beter zouden kunnen. Vaak kwam dat doordat ik bij aanvang van de opdracht nog niet over de informatie beschikte die ik later wel had. Het stukje 'te gebruiken technieken' bijvoorbeeld, daar staat in het plan van aanpak dat ik een navigatieschema zou gebruiken. Deze techniek heb ik niet gebruikt aangezien er per te ontwikkelen pilot maar één scherm was. Dit gaat niet op voor het onderdeel 'standaarden, richtlijnen en procedures'. Daarbij stel je van te voren een bepaalde aanpak vast, in dit geval over de opmaak van documenten. Hierbij is er niets dat anders uit kan pakken. Als je je er aan houdt, krijg je consistente documenten. Anders niet. De detailplanning is ook onderdeel van het plan van aanpak. Deze heb ik als zeer nuttig ervaren omdat ik me ermee dwong om op een bepaald pad te blijven. Hierdoor voorkwam ik dat ik te lang bleef hangen in een bepaalde activiteit. Als de tijd verstreken was en de activiteit geen basisprioriteit had, ging ik door naar de volgende activiteit. Wat minder goed was aan de planning is dat ik te weinig tijd had gereserveerd voor het schrijven van mijn afstudeerverslag. Door de planning hier aan te passen voorkwam ik dat ik in tijdnood raakte.

Rapport definitiestudie

Over het algemeen ben ik tevreden over het rapport definitiestudie. Ik heb er veel aangehad in de fase pilotontwikkeling. Het interview dat ik bij de ontwikkelaars van Spider heb afgenomen, is voor mij belangrijk geweest om vast te stellen waar de prioriteiten lagen bij het verbeteren van de gebruiksvriendelijkheid. Het interview met de gebruikers leverde, hoe opmerkelijk ook, minder bruikbare informatie op. Dit wijdt ik vooral aan het feit dat de belangrijkste gebruiker van de gebruikersgroep 'site administrators' niet gereageerd heeft. Een ander punt zou kunnen zijn, dat door het gebruik van de vragenlijsten de gebruikers belemmerd werden in het kunnen spuien van hun kritiek. In de definitiestudie had ik de techniek persona's gebruikt om meer te weten te komen over de gebruikers. Deze techniek heb ik echter halverwege deze fase laten vallen omdat dat gewoon niet werkte binnen mijn opdracht.

Pilotontwikkelplannen

Als ik nu achteraf de verschillende pilotontwikkelplannen bekijk, zie ik ze als een soort proeftuin waarin ik verschillende mogelijkheden onderzoek om in te zetten bij de ontwikkeling. Dit komt omdat ik op het moment van schrijven nog niet volledig kon bepalen of ik een bepaalde techniek in de praktijk ook echt kon toepassen vanwege mijn technische kennis. Hoewel de pilotontwikkelplannen normaal gesproken dus veel gedetailleerder zullen zijn en een minder experimenteel karakter hebben, heb ik er tijdens dit ontwikkeltraject geen hinder van ondervonden. In die zin ben ik wel tevreden over het resultaat.

Pilot 1: documenteditor

Door middel van de documenteditor kunnen nu vrij eenvoudig documenten worden toegevoegd waarbij alle functionaliteit gebruikt kan worden zonder ook maar een regel XML te hoeven schrijven. Dit was ook het aanvankelijke doel van deze pilot dus daar ben ik zeer tevreden over. Een ander punt waar ik tevreden over ben is dat gebruikers nu praktisch geen fouten kunnen maken bij het muteren van documenten. Er vindt validatie plaats bij alle tekst-invoervelden. In het verleden traden dikwijls

fouten op bij het maken of bewerken van een document. Een XML-document kan namelijk niet worden opgeslagen als de syntax ervan onjuist is. Bij het gebruik van de editor kunnen deze problemen niet meer optreden. Een ander punt waar ik tevreden over ben is het gemak waarmee nieuwe functionaliteit in Spider in de documenteditor kan worden toegevoegd: Als er nieuwe mogelijkheden beschikbaar komen die moeten worden gedefinieerd in een document, kan er met een paar regels code voor gezorgd worden dat deze nieuwe mogelijkheden vanuit de documenteditor kan worden toegevoegd. Een punt waar ik minder tevreden over ben is het feit dat er naar beneden gescrold moet worden bij het bewerken van een document. Hierdoor zijn de knoppen om het document op te slaan of te annuleren niet altijd in beeld. Dit zou opgelost kunnen worden door de datagrid in een layer te plaatsen waarin gescrold kan worden.

Pilot 2: template-editor

De template-editor die ik ontwikkeld heb, zorgt ervoor dat er nu templates kunnen worden aangemaakt, of bewerkt zonder dat daarvoor XML hoeft te worden ingetikt. In tegenstelling tot de documenteditor hoeft er hier niet horizontaal te worden gescrold om de verschillende elementen te kunnen zien. Dit komt doordat ik een boomstructuur heb gebruikt als navigatiemethode. Ik ben erg tevreden over deze boomstructuur. Naast het feit dat het een navigatiemethode is, zorgt het er ook voor dat de gebruiker zich een mentaal beeld vormt dat overeenkomt met de werkelijkheid. Hierdoor is hij beter in staat om het gedrag van het systeem te kunnen begrijpen en te voorspellen. Een punt dat wellicht verbeterd had gekund, is de helptekst die nu bij de template-editor is opgenomen. Hierin wordt nu eigenlijk onvoldoende uitgelegd wat voor plek templates innemen binnen Spider. In het algemeen ben ik in ieder geval zeer tevreden over deze pilot.

Pilot 3: CSS-editor

De laatste pilot die ik ontwikkeld heb, is de CSS-editor. Over het algemeen ben ik zeer tevreden over deze pilot. Ook hier heb ik de boomstructuur gebruikt om te voorkomen dat gebruikers verticaal moeten scrollen. Verder worden de verschillende soorten selectors in de boomstructuur gecategoriseerd weergegeven. Dit versterkt het mentale beeld bij de gebruiker dat er duidelijke verschillen bestaan tussen deze soorten. De soorten worden uitgelegd in de ingebouwde helptekst. Een ander punt waar ik tevreden over ben is het CSS-XML-formaat dat ik ontwikkeld heb. Toen ik het idee bedacht had samen met een van de ontwikkelaars van Spider, heb ik op internet gezocht of er al een soortgelijk concept bestond. Dat was toen niet het geval. Nu, aan het einde van mijn afstudeerperiode heb ik nog een keer gezocht en kwam ik op een website[3] terecht waar een soortgelijk idee beschreven werd. Origineel of niet origineel, het formaat zorgt er in ieder geval voor dat een stylesheet tot in detail te parsen is. Een ander punt waar ik tevreden over ben is de ingebouwde previewmogelijkheid. Hierdoor kan per selector worden bekeken hoe een layer gevuld met tekst eruit ziet als de selector daarvoor gebruikt wordt. Verder ben ik blij dat gebruikers kleuren kunnen kiezen zonder dat ze daar codes voor te hoeven gebruiken. Over het hele proces van het kiezen van een opmaakeigenschap ben ik voor het grootste gedeelte ook tevreden. Een nadeel daarbij is echter dat de eigenschappen nu nog in het engels worden weergegeven.

Test- en acceptatieformulier

Dit formulier heb ik gemaakt omdat ik graag de IAD-ontwikkelcyclus compleet wilde afronden. Voorgaande ontwikkeltrajecten die ik tijdens mijn studie heb meegemaakt, hielden meestal op na de fase pilotontwikkeling. Omdat ik de cirkel graag formeel een keer rond wilde maken, heb ik besloten om het test- en acceptatieformulier te maken. In die zin ben ik tevreden dat ik het gemaakt heb. Minder tevreden ben ik echter over de kwaliteit en het uiteindelijk bereikte effect van het formulier. Het geeft een beetje een knullige indruk aangezien ik weet dat de test- en acceptatieprocedure bij grotere ontwikkeltrajecten veel omvangrijker is.

[3] http://www.cubicgarden.com/blojsom/blog/cubicgarden/?month=2&day=2&year=2005

Figurenlijst

Geraadpleegde bronnen

Boeken

Duthie, A.G. e.a., *Programmeercursus Microsoft ASP.NET*. 2e druk, Academic Service, Schoonhoven, 2002

Archer, T., *Microsoft Handboek C# - Programmeren met C#.* 2e editie, Academic Service, Schoonhoven, 2003

Tolido, R.J.H., *IAD - het evolutionair ontwikkelen van informatiesystemen*. 1e druk, Academic Service, Schoonhoven, 2000

Internet

D. Greefhorst en M. van Elswijk, *Eigenschappen van moderne ontwikkelmodellen - Vier modellen vergeleken*,
http://www.serc.nl/

Ståle Refsnes, *W3Schools CSS tutorial + reference*,
http://www.w3schools.com/css/default.asp

Jan Egil Refsnes, *W3Schools XML tutorial*,
http://www.w3schools.com/xml/default.asp

Jan Egil Refsnes, *W3Schools XPath tutorial*,
http://www.w3schools.com/xpath/default.asp

Hans de Jong, *Handleiding CSS,*
http://www.handleidinghtml.nl/css/

Microsoft, *Creating Templates Programatically in the DataGrid Control,*
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vbcon/html/vbtskcreatingtemplates programmaticallyindatagridcontrol.asp

J. Bipin, *Creating DataGrid Templated Columns Dynamically - Part II,*
http://www.dotnetbips.com/4A2A1004-FAFC-4064-BE59-2D92D6099860.aspx?articleid=85

Nielsen, J., *Heuristics for User Interface Design,*
http://www.useit.com/papers/heuristic/heuristic_list.html

Microsoft, *.NET Framework Class Library,*
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/frlrfsystemwindowsformstextboxclasstopic.asp

Gebruikte afkortingen en woordenlijst

.NET Platform Programmeerinfrastructuur ontwikkeld door Microsoft om programma's en service die gebruik maken van .NET technologie te ontwikkelen en uit te voeren.

Array Een array bestaat uit verschillende componenten van hetzelfde type. Elk component is uniek te onderscheiden doormiddel van een index (voor een 1-dimensionale array) of een serie indexen bij multi-dimensionale arrays. Bij bijna alle programmeertalen worden arrays gebruikt om verzamelingen objecten in op te slaan.

ASP.NET Een serie .NET-klassen die gebruikt worden om webapplicaties te maken.

Binden Zie databinding.

Blackbox testen Blackbox testen (ook wel aangeduid met functioneel testen) is een software testtechniek waarbij een uitvoerige kennis van de interne werking van het geteste item niet vereist is. De tester hoeft alleen de invoer en de verwachte uitvoer te weten en niet wat het programma ermee doet. Voordeel van deze manier van testen is dat de programmeur en de tester onafhankelijk van elkaar zijn. Verder wordt de test uitgevoerd vanuit het oogpunt van de gebruiker en kunnen testgevallen opgezet worden zodra de specificatie zijn vastgesteld. Nadelen van deze manier van testen kunnen zijn dat tests onnodig dubbel kunnen worden uitgevoerd als de programmeur de test ook al heeft uitgevoerd. Verder zijn de testgevallen moeilijk te ontwikkelen en is het onrealistisch om elke mogelijke invoer te kunnen testen. Daardoor blijven er vele mogelijkheden over die niet getest worden. Voor een volledige softwaretest zijn zowel whitebox- als blackboxtests vereist.

BLL Business Logic Layer. Laag van een applicatie-architectuur waarbinnen de applicatielogica en objecten gemodelleerd zijn.

C# C# (uit te spreken als C-Sharp) is een objectgeoriënteerde programmeertaal die ontwikkeld is door Microsoft speciaal voor het .NET-platform. C# wordt gezien als de natuurlijke opvolger van de programmeertalen C en C++ .

Categorie Categorieën en subcategorieën zijn binnen Spider onderwerpen binnen een website waarin content geplaatst kunnen worden. Enkele voorbeelden van categorieën en subcategorieën binnen een website zijn:

- Nieuws
  - Binnenlands nieuws
  - Buitenlands nieuws
- Links
  - Verwante links
  - Overige links

Checkbox Schermelement binnen (web)applicaties waarbij meerdere keuzes uit een lijst kunnen worden geselecteerd door items uit die lijst aan te vinken.

Class Binnen het .NET-platform zijn klassen (classes) containers waarin een type wordt gedefinieerd. Klassen beschrijven de eigenschappen en het gedrag van objecten.

Content(-collection) Content is in Spider een verzameling 'content-items' over hetzelfde onderwerp. Als content 'geplaatst' wordt in een bepaalde (sub-)categorie, is het zichtbaar op de website.

Content-item Een content-item is in Spider een stukje content dat deel uitmaakt van een contentverzameling (content-collection). Een content-item onder andere zijn:

- De titel van een artikel
- De broodtekst bij een artikel
- De inhoud van een artikel
- Een afbeelding behorend bij een artikel
- Meta-informatie over het artikel

CSS CSS staat voor cascading stylesheets. Het is een techniek waarmee de opmaak van websites centraal kan worden geregeld. Een CSS-document is een verzameling van opmaakregels die betrekking hebben op één of meerdere HTML-tags of klassen van HTML-tags. Binnen Spider kan onder

	andere verwezen worden naar deze opmaakregels vanuit een Style of vanuit content.
CSS-XML document	XML-formaat die ik in het kader van de afstudeeropdracht heb ontwikkeld om CSS-documenten vast te leggen in XML zodat deze beter te interpreteren zijn door een CSS-editor.
DAL	Data Access Layer. Laag van de applicatie-architectuur waardoor objecten (gedefinieerd in de BLL) gekoppeld worden aan de fysieke opslag (database of XML)
Databinding	Databinding is bij ASP.NET het proces waarbij een server-component, zoals een textbox of een datagrid-control een waarde weergeven die afkomstig is van een datasource
DataGrid(control)	Klasse binnen het (ASP).NET-platform die het mogelijk maakt om snel dataverzamelingen in tabelvorm weer te geven en te wijzigen.
Datasource	Een datasource is een gegevensverzameling die als bron dient voor een ander component. Dat kan bijvoorbeeld een selectie op een database, een XML-bestand of een array zijn.
Debugging	Het verwijderen van fouten uit software.
Declaratie	Binnen het CSS-formaat is een declaratie een combinatie van een eigenschap met een daarbij ingevulde waarde.
Definitiestudie	Eerste fase binnen de IAD-ontwikkelcyclus waarbinnen ondermeer de doelen van het systeem worden geanalyseerd. Het eindproduct van deze fase is onder andere een rapport. Voor dit rapport wordt ook de term ‘definitiestudie’ gebruikt.
Documentdefinitie	Een documentdefinitie is binnen Spider de benaming voor een XML-fragment waarmee een document gedefinieerd wordt. Zie ook ‘Spider-Document’.
Dropdownlistbox	Schermelement binnen (web)applicaties waarmee een item uit een lijst kan worden geselecteerd. De lijst klapt naar beneden als een gebruiker op het schermelement klikt.
Event	Een bericht dat door een programma wordt gegenereerd als er iets is gebeurd (bijvoorbeeld het drukken op een knop). Als antwoord op een event kan een event-handler worden aangeroepen om te specificeren wat voor actie er moet worden ondernomen.
GUI	Graphical User Interface. Grafische schil van een computerprogramma dat grafische elementen vertaalt naar programma-opdrachten en andersom.
GUIDE	Graphical User Interface Design and Evaluation. Methode voor het systematisch ontwerpen van GUI’s.
Heuristic Evaluation	Een heuristic evaluation is een populaire methode om usability bij software te meten. Bij een heuristic evaluation beoordelen de beoordelaars de software aan de hand van usability-principes.
HTML	HyperText Markup Language. Opmaak-taal waarmee documenten kunnen worden gedefinieerd die kunnen worden weergegeven in een webbrowser. HTML-pagina’s kunnen verwijzingen naar plaatjes, ander HTML-pagina’s of andere elementen bevatten. Des te meer beoordelaars er mee doen aan de test, des te meer usabilityproblemen er gevonden kunnen worden.
IAD	Iterative / Interactive / Incremental Application Design. Methode om gestructureerd applicaties te ontwikkelen op een Iteratieve wijze. Dat wil zeggen dat er steeds een vaste cyclus wordt doorlopen waarbij er elke keer nieuwe functionaliteit beschikbaar komt.
Klasse	Zie Class.
Listbox	Schermelement binnen (web)applicaties waarbij een of meer keuzes kan worden gemaakt uit een lijst.
Microsoft Internal Coding Guidelines	Van oorsprong interne afspraken van Microsoft die zijn opgesteld over de opmaak en structuur van programmacode. De afspraken worden nu ook gehanteerd door andere ontwikkelaars.
MSDN	Microsoft Developers Network. Portal voor ontwikkelaars van/aan microsoft-technologieën.
Parsing	Parsing is het proces van het opsplitsen van een ononderbroken stroom van

	karakters (die bijvoorbeeld uit een bestand worden gelezen) in betekenisvolle delen om daar een boomstructuur (parse-tree) van op te bouwen. Er zijn twee belangrijke manieren om een data-stroom te ontleden: top-down-parsing, waarbij de parse-tree wordt opgebouwd beginnend bij de grootste elementen waarna deze worden opgesplitst in kleinere delen en bottom-up-parsing waarbij wordt begonnen met parsen bij de kleinste elementen. Een parser is een computerprogramma die dit proces uitvoert.
Persona's	Het maken van persona's is een hulpmiddel dat de auteur en software-ontwerper Alan Cooper heeft ontwikkeld op basis van een oude techniek uit de reclamewereld. Om hun producten beter aan de man te kunnen brengen gebruikten marketingmensen de demografische gegevens die ze hadden over hun doelgroep. Daarmee probeerden ze een bepaald stereotype neer te zetten die stond voor de doelgroep die ze wilden bereiken. Ze bouwden op die manier een identificatie op met hun doelgroep, waardoor ze beter konden zien wat aan zou sluiten bij de verwachtingen van de persoon die ze wilden bereiken.
Pilot	Een samenhangende subset van een te ontwikkelen informatiesysteem die als zelfstandige, bruikbare eenheid kan worden ingevoerd in de organisatie.
Pilotontwikkeling	Tweede fase van de IAD-ontwikkelcyclus waarbinnen een pilot wordt ontwikkeld.
RAD	Er wordt gesproken van RAD (Rapid Application Development) wanneer de traditionele onbuigzaamheid van de levenscyclus systeemontwikkeling omzeild wordt door snel een bruikbaar programma af te leveren. Men wil de wijziging van de externe omstandigheden als het ware vóór zijn: hoe korter de ontwikkelingstijd, hoe kleiner de kans dat de behoeften van de opdrachtgever ondertussen wijzigen.
Radiobutton	Schermelement binnen (web)applicaties waarbij één keuze kan worden gemaakt uit meerdere opties.
Regular expressions	Regular expressions, vaak aangeduid met 'RegEx' is een serie standaard lettercombinaties die gebruikt worden om bepaalde patronen in tekst te kunnen vinden.
SOAP	Simple Object Access Protocol. Protocol voor de berichtafhandeling van op web services.
Spider	Content Management Systeem voor Websites ontwikkeld door Marbel Software.
Spider-document	Een document is binnen Spider een XML-bestand waarin wordt gespecificeerd wat voor en hoeveel content er moet worden weergegeven op een pagina die opgevraagd kan worden door een browser en hoe deze moet worden gepresenteerd. Een document kan verwijzingen bevatten naar de 'categorie', subcategorie', 'style' en naar andere documenten.
Stijl	Een stijl is binnen Spider de manier hoe content wordt weergegeven in een gedeelte binnen een bepaalde pagina. De stijl wordt gedefinieerd door middel van XSL-T documenten. De stijl dient niet verward te worden met CSS. Een stijl regelt de inhoud van content die moet worden weergegeven. Ook kan in een stijl worden gedefinieerd volgens welk CSS-element de content moet worden weergegeven. De daadwerkelijke opmaak wordt vastgelegd in een 'CSS-document'. Binnen een stijl kunnen logica en condities worden meegegeven die de representatie van content kunnen beïnvloeden.
String	
Style	Een Style is binnen Spider een XSL-T document waarin de representatie van content wordt geconfigureerd.
Styleguide	Een styleguide is een verzameling van afspraken omtrent de vormgeving van documenten, programma's of webpagina's.
Subcategorie	Zie categorie.
Template	Een template is binnen Spider een definitie van uit welke content-items een content-verzameling bestaat. Een template voor een kranten-artikel bevat dus bijvoorbeeld de volgende elementen (content-items):

- Titel (van het type tekst)
- Lead (van het type tekst)
- Afbeelding (als verwijzing naar een afbeelding in de database).

UML UML (Unified Modelling Language) is een standaard taal om aspecten van software systemen te specificeren, visualiseren, construeren en te documenteren voor business modellering en andere niet-software systemen. De UML presenteert een verzameling zogenaamde best practices dat haar succes bewezen heeft in het modelleren van grote en complexe systemen.

Usability De mate waarin een interactief systeem (website, software e.d.) de gebruiker in staat stelt om effectief, efficiënt en comfortabel in een gegeven omgeving zijn taak te voltooien.

Visual Studio (.NET) Visual Studio .NET is een complete set ontwikkelgereedschappen om webapplicaties, webservices, desktop applicaties en mobiele applicaties mee te ontwikkelen. De programmeertalen Visual Basic .NET, Visual C++, Visual C# en Visual J# maken allen gebruik van dezelfde ontwikkelomgeving.

W3C Het W3C (World Wide Web Consortium) is een organisatie die als doel heeft om gemeenschappelijke protocollen te ontwikkelen die de groei van het Web bevorderen en interoperabiliteit garanderen. Het W3C heeft onder andere de standaarden voor HTML, CSS en XML bepaald.

Web Service Een Web Service bestaat uit software die zichzelf toegankelijk maakt via internet en gebruikmaakt van een gestandaardiseerd XML-berichtensysteem. Een Web Service kan een publieke interface hebben die de methodes (oftewel functies) beschrijft die de Web Service kan uitvoeren. Een Web Service moet de mogelijkheid hebben om deze beschikbaar te stellen aan mogelijke gebruikers via een standaard methode.

Whitebox testen Whitebox testen (ook wel aangeduid met glassbox, structural of clearbox testen) is een software-testtechniek waarbij een uitvoerige kennis van de interne werking van het geteste item vereist is. In tegenstelling tot blackbox testen vereist whitebox testen specifieke kennis van programmacode om resultaten te bestuderen. De test is alleen accuraat als de tester weet wat het programma hoort te doen. Hij of zij kan dan bepalen of het gewenste gedrag van het programma afwijkt van het feitelijke (gemeten) gedrag.

XML Extensible Markup Language (XML) is een markuptaal die, net als HTML, is ontwikkeld uit de SGML-standaard. Bij XML gaat alles om markup: een manier om in data informatie te stoppen die beschrijft wat die data is. In data worden tags gezet die meer informatie geven over de data, die dus semantiek toevoegen aan de data. Door aan je data tags toe te voegen, creëer je data-containers. Elke tag geeft het begin of einde van een nieuwe data-container aan.

XML DOM Het XML Document Object Model (XML DOM) definieert een standaard manier om toegang te krijgen tot informatie in XML documenten en deze te kunnen manipuleren. Het model presenteert een XML document als een boomstructuur en geeft toegang tot die structuur via een serie objecten. Deze objecten worden toegepast binnen verschillende (ontwikkel) omgevingen.

XML-(de)serialisatie XML serialisatie geeft programma's de mogelijkheid om de status en waarden van objecten op te slaan in XML-bestanden. Deserialisatie is het terugvertalen van de XML bestanden naar objecten.

XmlNode Een XMLNode is een bepaald punt binnen het DOM van een XML document. Binnen het .NET framework is XmlNode de basisklasse in de implementatie van het DOM. XmlDocument is een uitbreiding van die klasse. Daarbij wordt functionaliteit toegevoegd om XML-data te openen en op te slaan.

XmlNodeList XmlNodeList is een object binnen het .NET framework dat een geordende lijst van XmlNode-objecten representeert.

XPath XPath is een manier om een bepaalde plek binnen een XML-document te kunnen specificeren. Een XPath-expressie is een route die gevolgd moet

	worden in een XML-document om bepaalde informatie te vinden. XPath is samen met XSL-T en XSL-FO één van de drie technieken die onderdeel zijn van XSL.
XSD	XML Schema Definition (XSD) is een taal waarmee de structuur van een XML document beschreven kan worden. Als de structuur van een XML document eenmaal is vastgelegd in een XML Schema wordt het mogelijk om XML documenten te kunnen controleren op deze structuur.
XSL	XSL is de verzamelnaam voor drie aan XML gerelateerde technieken: • XPath • XSL-T • XSL-FO
XSL-T	Extensible Stylesheet Language Transformations (XSL-T) is een taal waarmee XML documenten kunnen worden getransformeerd naar XML documenten met een andere structuur, HTML of tekst.

Definitieve omschrijving

Afstudeeropdracht

student:	Dhr. M. Verheul
keurend docent:	Dhr. J.J. van Beijnum
afstudeerbedrijf:	Marbel Software B.V. te Noordwijkerhout
bedrijfsmentor:	Dhr. Ing. G. Scheeres
datum:	30 maart 2005
afstudeerrichting:	Informatica en Informatiekunde: VIA
afstudeerblok	2005-1.1
versie:	1

Afstudeerblok:	2005-1.1
Studentnummer:	20010124
Afstudeerrichting:	Informatica en Informatiekunde: VIA

Achternaam:	Verheul
Voorletters:	M.
Roepnaam:	Michiel
Adres:	Dorsersstraat 13
Postcode:	2151 CE
Woonplaats:	Nieuw-Vennep
Telefoon:	0252-689425

Naam bedrijf:	Marbel Software B.V.
Afdeling bedrijf:	-
Bezoekadres:	Gieterij 32
PC+Plaats bezoekadres:	2211 XK Noordwijkerhout
Postbusnummer:	Postbus 111
Postcode postbusnummer:	2215 ZJ Voorhout
Telefoon bedrijf:	0252-241210
Telefax bedrijf:	0252-241218

Titel afstudeeropdracht: “Het verbeteren van de usability van Spider-CMS”

Inleiding (organisatorische omgeving, kader, historie):

Marbel Software, opgericht in november 1990, is gespecialiseerd in het ontwikkelen en implementeren van maatwerk (software) systemen, zowel binnen Local Area Networks als Wide Area Networks. Deze systemen kunnen variëren van administratief tot en met technische (datacommunicatie) systemen. Bij het ontwikkelen van deze systemen past Marbel Software moderne technieken zoals XML en SOAP toe.
Spider is een content management systeem (CMS) voor Websites dat Marbel Software oorspronkelijk heeft ontwikkeld voor NDLite, de jongerensite van het Nederlands Dagblad. Met Spider is het mogelijk om redacteurs en moderators aan te wijzen die allen verantwoordelijk zijn voor hun eigen content. Verder kunnen bezoekers van de Website reageren op berichten en kunnen ze een sms-bericht ontvangen als er belangrijke nieuwe content is geplaatst op de Website. Spider is volledig gebaseerd op XML, CSS en XSL-T en is geschreven voor het ontwikkelplatform ASP.NET in de programmeertaal C#. Spider wordt nu ook ingezet als CMS bij kleinere Websites. De afstudeeropdracht draait om het verbeteren van Spider.

Probleemstelling:

Regelmatig komt het voor dat gebruikers (gebruikersgroep webmasters) van Spider naar Marbel Software bellen omdat ze de structuur en de opmaak van hun Website willen aanpassen. Zelf kunnen zij dit niet omdat ze niet overweg kunnen met de technieken XML en CSS. In Spider is het gebruik van deze technieken nodig om de structuur en de opmaak van een Website aan te passen.

Doelstelling van de opdracht:

Het doel van de afstudeeropdracht is het verbeteren van Spider zodat gebruikers (gebruikersgroep webmasters) zelfstandig in staat zijn om de structuur en de opmaak van hun Website aan te passen zonder ervaring te hoeven hebben met de technieken XML en CSS.

Uitgangssituatie
Benodigde software:

- Microsoft Windows XP
- Microsoft Office XP
- Microsoft Visual Studio 2003 .NET
- Adobe Photoshop

Benodigde hardware:

Een modern werkstation met voldoende processorcapaciteit en geheugen (2GHz/512MB)

Beschikbare rapporten:

Bij aanvang van de afstudeeropdracht is er een rapport van een usability-analyse beschikbaar. Hierin staat een lijst met probleempunten op het gebied van de usability van Spider.

Aanwezige ideeën:

Het ontwikkelen van een front-end dat er voor zorgt dat gebruikers zonder het intikken van XML of CSS codes de vormgeving en de structuur van hun Website kunnen aanpassen.

Concrete werkzaamheden
Uit te voeren activiteiten:

1. Opstellen plan van aanpak
2. Definiëren ontwikkelscenario
3. Definiëren systeemeisen
    - Interviewen van de ontwikkelaars om een beeld te krijgen van de vragen die gebruikers stellen
    - Analyseren van de doelgroep (gebruikers)
    - Interviewen van de gebruikers om hun wensen in kaart te brengen
4. Bepalen systeemconcept
    - Globaal beschrijven van de beoogde oplossing op basis van de systeemeisen
5. Beschouwen technische structuur
    - Onderzoeken van de haalbaarheid door middel van experimentele proto-typing
6. Opstellen pilotplan
7. Specificeren globaal-functionele structuur pilot
    - Detailleren van het systeemconcept
    - Maken van schermontwerpen
    - Vaststellen events
8. Specificeren globaal-technische structuur pilot
    - Interviewen van de ontwikkelaars om alle opties van Spider in kaart te brengen
9. Specificeren globaal-organisatorische inrichting
10. Opstellen pilotontwikkelplan
11. Ontwerpen en bouwen software-bouweenheden
12. Samenstellen handleidingen
13. Invoeren pilot (integreren binnen Spider)
14. Verzamelen feedback

Te hanteren methodieken:

- IAD

Te gebruiken technieken:

- UML
- XML
- XSL-T
- DTD
- ASP.NET
- C#
- GUIDE

Te vermelden nadrukken:

De nadruk ligt bij deze afstudeeropdracht op het verbeteren van het product Spider zodat gebruikers zelfstandig de structuur en de opmaak van hun Website kunnen aanpassen.

Eventueel aanvullende opmerkingen:

De student is bij aanvang van het afstudeertraject nog niet bekend met alle te gebruiken technieken.

Resultaten voor de opdrachtgever (op te leveren producten):

De volgende producten zullen opgeleverd worden:

- Plan van aanpak.
- Definitiestudie.
- Pilotontwikkelplannen.
- Volledig werkende GUI's voor het ontwerpen/aanpassen van de opmaak en de structuur van de in Spider beheerde Websites.
- Testrapport.

Plan van Aanpak

Afstudeeropdracht

student:	Dhr. M. Verheul
keurend docent:	Dhr. J.J. van Beijnum
afstudeerbedrijf:	Marbel Software B.V. te Noordwijkerhout
bedrijfsmentor:	Dhr. Ing. G. Scheeres
datum:	10 juni 2005
afstudeerrichting:	Informatica en Informatiekunde: VIA
afstudeerblok	2005-1.1
versie:	1

Inhoudsopgave

1. Inleiding

Een plan van aanpak is een document waarin alle afspraken staan tussen de opdrachtgever en de uitvoerende partij, in dit geval de afstudeerder. Het plan van aanpak omvat afspraken met betrekking tot de omvang van de opdracht, de verwachte duur van het project, de te volgen aanpak, de projectinrichting en kwaliteitsborging. Een deel van de informatie in dit plan van aanpak staat ook al beschreven in de opdrachtomschrijving. Dit plan van aanpak gaat dieper in op de opdracht aan de hand van de volgende hoofdstukken.
Hoofdstuk 2 “Opdrachtomschrijving” bevat een uitvoerige beschrijving van de opdracht en een lijst met op te leveren producten en diensten. Verder worden er de randvoorwaarden en risicofactoren beschreven.
In hoofdstuk 3 “Aanpak” wordt beschreven hoe het project precies uitgevoerd gaat worden. Er wordt overeenstemming bereikt over de te volgen weg naar het eindresultaat.
In hoofdstuk 4 “Projectinrichting” wordt beschreven hoe het project ingericht moet worden om aan de aanpak uit hoofdstuk 3 te kunnen voldoen.
Tenslotte wordt in hoofdstuk 5 “Kwaliteitsborging” beschreven aan welke kwaliteitseisen het resultaat moet voldoen en hoe die kwaliteit bereikt kan worden.

1.1. Achtergrond en aanleiding van de opdracht

Marbel Software, opgericht in november 1990, is gespecialiseerd in het ontwikkelen en implementeren van maatwerk (software) systemen, zowel binnen Local Area Networks als Wide Area Networks. Deze systemen kunnen variëren van administratief tot en met technische (datacommunicatie) systemen. Bij het ontwikkelen van deze systemen past Marbel Software moderne technieken zoals XML en SOAP toe.
Spider is een content management systeem (CMS) voor websites dat Marbel Software oorspronkelijk heeft ontwikkeld voor NDLite, de jongerensite van het Nederlands Dagblad. Met Spider is het mogelijk om redacteurs en moderators aan te wijzen die allen verantwoordelijk zijn voor hun eigen content. Verder kunnen bezoekers van de website reageren op berichten en kunnen ze een sms-bericht ontvangen als er belangrijke nieuwe content is geplaatst op de website. Spider is volledig gebaseerd op XML en XSL-T en is geschreven voor het ontwikkelplatform ASP.NET in de programmeertaal C#. Spider wordt nu ook ingezet als CMS bij kleinere websites. Uit de vragen van gebruikers van Spider die geregeld binnenkomen, blijkt dat veel gebruikers moeite hebben met het veranderen van de opmaak en de structuur van een website met Spider. De afstudeeropdracht draait om het verbeteren van Spider zodat het voor gebruikers eenvoudiger wordt om de opmaak en de structuur van een website aan te kunnen passen.

2. Opdrachtomschrijving

In dit hoofdstuk wordt de opdracht uitgebreid beschreven. De opdracht wordt afgebakend door middel van het beantwoorden van “waarom”, “waarover” en “wat”-vragen.

2.1. Opdrachtgever

De opdrachtgever is Marbel Software. De contactpersoon binnen Marbel Software is Martin Weijers.
De gebruikers zijn klanten van Marbel Software die gebruik maken van Spider.
De opdrachtgever geeft feedback op de opgeleverde documentatie en producten. Binnen het bedrijf wordt er geen methode of werkwijze voorgeschreven voor softwareontwikkelingsprojecten.

2.2. Probleemstelling

Regelmatig komt het voor dat gebruikers (gebruikersgroep webmasters) van Spider naar Marbel Software bellen omdat ze de structuur en de opmaak van hun website willen aanpassen.
Zelf kunnen zij dit niet omdat ze niet overweg kunnen met de technieken XML en XSL-T. In Spider is het gebruik van deze technieken nodig om de structuur en de opmaak van een website aan te passen.

2.3. Doelstelling opdracht

De doelstelling van de afstudeeropdracht is het verbeteren van Spider zodat gebruikers (gebruikersgroep webmasters) zelfstandig in staat zijn om de structuur en de opmaak van hun website aan te passen zonder ervaring te hoeven hebben met de technieken XML en XSL-T.

2.4. Opdrachtformulering

Eerst moet er worden onderzocht waar gebruikers precies problemen ondervinden bij het gebruik van Spider. Als deze problemen in kaart zijn gebracht, kan er een ontwerp worden gemaakt van de te ontwikkelen oplossing. Vervolgens wordt deze oplossing ontwikkeld en geïmplementeerd. Het is bij aanvang van het project nog niet helemaal duidelijk wat voor oplossing er precies ontwikkeld gaat worden. Dit hangt af van de wensen van de opdrachtgever en de gebruikers.

2.5. Op te leveren producten en diensten

In het kader van de afstudeeropdracht zullen de volgende producten worden opgeleverd aan de opdrachtgever:

- Plan van Aanpak
- Definitiestudie
- Pilotontwikkelplannen
- Gebruikersdocumentatie
- Systeemdocumentatie
- Volledig werkende GUI voor het ontwerpen / aanpassen van de opmaak en de structuur van de in Spider beheerde website’s.

De eerste drie producten worden gemaakt om het afstudeerproces te ondersteunen.
De laatste drie producten vormen samen het uiteindelijke resultaat en leveren een directe bijdrage aan de in paragraaf 2.3 geformuleerde doelstelling.

2.6. Randvoorwaarden

De volgende niet-inhoudelijke eisen worden gesteld aan het project:

Tijdspad: De afstudeerperiode loopt van 7 februari 2005 tot en met 10 juni 2005.
Werkwijze: De opdrachtgever stelt geen eisen aan de te volgen werkwijze.
Werkplek: De afstudeerder dient te beschikken over een werkplek met een pc, printer en internet.
Documentatie: De opgeleverde documentatie dient duidelijk en volledig te zijn zodat de opdrachtgever goed op de hoogte is van het proces en de gemaakte keuzes. Aan de hand van de documenten kan het eindproduct beoordeeld worden.
Betrokkenen: Het is van belang dat de betrokkenen (opdrachtgever, ontwikkelaars, klanten) voldoende medewerking verlenen aan het project.

2.7. Risicofactoren

De volgende factoren kunnen het slagen van het afstudeerproject in de weg staan:

- Gebrek aan technische kennis bij de afstudeerder
  - Tijdens het afstudeerproject wordt gewerkt met technieken die nog niet geheel bekend zijn bij de afstudeerder (C#, ASP.NET, XSL-T).
  - Om het risico van het optreden van deze factor zo veel mogelijk te beperken, heeft de afstudeerder verschillende boeken aangeschaft om meer te leren over deze nieuwe technieken.
- Conflicten met de opdrachtgever
  - Tijdens het afstudeerproject zouden conflicten met de opdrachtgever kunnen optreden als er sprake is van onvoldoende communicatie of als de odrachtgever of afstudeerder zich niet aan de afspraken houdt.
  - De kans op het optreden van deze factor is minimaal als er vanaf beide kanten goed wordt gecommuniceerd en er goede afspraken kunnen worden gemaakt.
- Gebrek aan tijd
  - Er zou een gebrek aan tijd kunnen ontstaan doordat de te verrichten activiteiten onderschat worden of doordat bepaalde activiteiten meer tijd kosten.
  - Om te voorkomen dat deze factor optreedt, wordt er goed nagedacht over de planning in dit plan van aanpak. Verder dient deze planning natuurlijk ook gevolgd te worden.

3. Aanpak

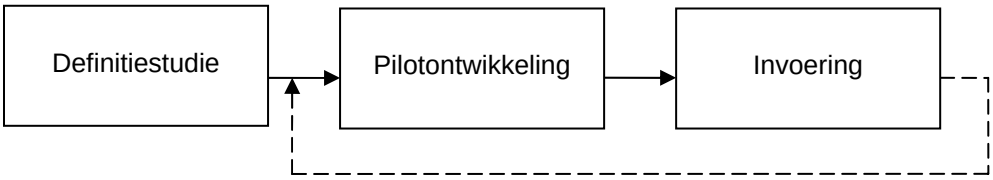
In dit hoofdstuk wordt beschreven hoe het project uitgevoerd gaat worden. Het beschrijft de weg die gevolgd gaat worden naar het gewenste eindresultaat.

3.1. Methoden en technieken

De volgende methoden en technieken zullen tijdens de uitvoering van de opdracht toegepast worden.

3.1.1. Methoden

Bij het afstudeerproject zal gebruik worden gemaakt van IAD als systeemontwikkelingmethodiek. IAD betekent Interactive / Iterative / Incremental Application Design. ‘Interactive’ betekent dat er ‘samen’ (of in samenspraak) met de gebruikers wordt ontwikkeld. ‘Iterative’ betekent dat er wordt ontwikkeld volgens een vast en herhalend proces. ‘Incremental’ betekent dat bij elke iteratieslag er nieuwe functionaliteit bijkomt die bijdraagt aan een grotere bruikbaarheid van het systeem.



Figuur 1: Incrementeel ontwikkelen

Een belangrijke keuze die gemaakt moet worden bij het gebruik van IAD als systeemontwikkeling-methodiek is de te gebruiken iteratiestrategie. Bij dit afstudeerproject zal de iteratiestrategie ‘incrementeel ontwikkelen’ gebruikt worden om de volgende redenen.

- Er is sprake van een relatief klein informatiesysteem.
- De projectomgeving is stabiel.
- De gebruikers zijn weinig beschikbaar voor workshops.

In het afstudeerproject is er echter geen ruimte (tijd) voor een iteratieslag. Er is dus eigenlijk niet echt sprake van een iteratief proces. De iteratiestrategie bepaalt echter ook waar de nadruk op wordt gelegd bij het doorlopen van de activiteiten binnen elke fase.
Om er voor te zorgen dat het eindproduct voldoet aan de wensen van de gebruiker wordt er gebruik gemaakt van interviews om de systeemeisen op te stellen.

3.1.2. Technieken

De volgende technieken zullen worden gebruikt

- UML Use-case diagram
- UML Class diagram
- Navigatieschema
- Interviews
- Timeboxing

3.2. Standaards, richtlijnen en procedures

Om te zorgen voor consistentie in de documenten die opgeleverd worden, zijn de volgende richtlijnen opgesteld.

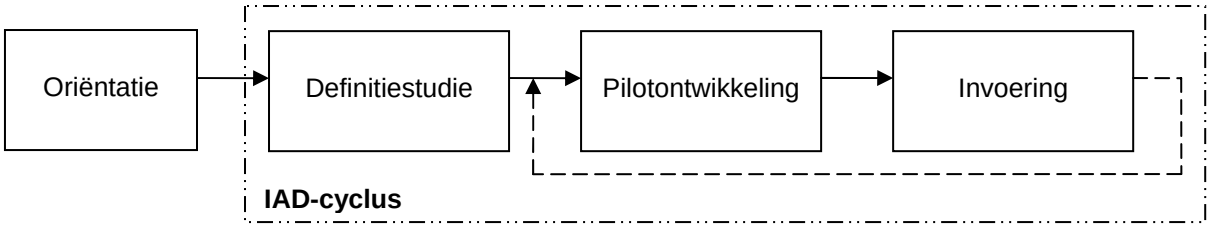
- Elke pagina heeft een koptekst de volgende elementen
  - Logo
  - Titel document
  - Naam auteur
  - Studentennummer
  - Maand, jaar
- Elke pagina heeft een voettekst met daarin het paginanummer en het aantal pagina's
- Na een kop volgt altijd één 'wit'-regel
- Na een paragraaf of hoofdstuk volgen altijd twee 'wit'-regels
- Er wordt gebruik gemaakt van de volgende opmaakprofielen.
- Alle alinea's worden links uitgelijnd
- Tussen een alinea en een opsomming zit altijd één 'wit'-regel

Kop 1: Arial, 12pt, vet.
Kop 2: Arial, 11pt, vet.
Kop 3: Arial, 10pt, vet.
Bodytekst: Arial, 10pt.
Bijschrift: Arial 8pt, vet.

3.3. Werkzaamheden

Het afstudeerproject wordt uitgevoerd volgens de IAD-systeemontwikkelmethodiek. Daarom wordt grotendeels dezelfde fasering als bij IAD gehanteerd. Het afstudeerproject bestaat uit de volgende fases.

0. Oriëntatie
1. Definitiestudie
2. Pilotontwikkeling
3. Invoering



Figuur 2: IAD cyclus uitgebreid met oriëntatie-fase

De oriëntatiefase bestaat niet in binnen IAD. Deze fase is toegevoegd om de voorbereidende werkzaamheden in onder te brengen zoals het schrijven van dit plan van aanpak.
Binnen elke fase worden een aantal activiteiten uitgevoerd. De volgende activiteiten worden in chronologische volgorde uitgevoerd.

- **Oriëntatie**
  - Opstellen plan van aanpak
- **Definitiestudie**
  - Definiëren ontwikkelscenario
  - Definiëren systeemeisen
    - Interviewen van de ontwikkelaars om een beeld te krijgen van de vragen die gebruikers stellen
    - Analyseren van de doelgroep (gebruikers)
    - Interviewen van de gebruikers om hun wensen in kaart te brengen
  - Bepalen systeemconcept
    - Globaal beschrijven van de beoogde oplossing op basis van de systeemeisen
  - Beschouwen technische structuur
    - Onderzoeken van de haalbaarheid door middel van experimentele proto-typing
  - Opstellen pilotplan
- **Pilotontwikkeling**
  - Specificeren globaal-functionele structuur pilot
    - Detailleren van het systeemconcept
    - Maken van schermontwerpen
    - Vaststellen events
  - Specificeren globaal-technische structuur pilot
    - Interviewen van de ontwikkelaars om alle opties van Spider in kaart te brengen
  - Specificeren globaal-organisatorische inrichting
  - Opstellen pilotontwikkelplan
  - Ontwerpen en bouwen software-bouweenheden
  - Samenstellen handleidingen
- **Invoering**
  - Invoeren pilot (integreren binnen Spider)
  - Verzamelen feedback

3.4. Planning

Bij dit afstudeerproject wordt gebruik gemaakt van time-boxing. Time-boxing is een planningstechniek waarbij de activiteiten worden opgesplitst in zo klein mogelijke eenheden. Als bijlage zijn planningsheets opgenomen die gebruikt zullen worden met de techniek time-boxing. Op het moment van het schrijven van dit plan van aanpak is nog niet bekend welke software-bouweenheden er zullen zijn. De software-bouweenheden worden nader gespecificeerd aan het begin van de fase pilotontwikkeling.

4. Projectinrichting

In dit hoofdstuk wordt beschreven hoe het project wordt ingericht.

4.1. Projectorganisatie

De student is geheel zelf verantwoordelijk voor het resultaat van het afstudeerproject. Wel zijn er vanuit de organisatie en vanuit school een aantal betrokkenen.

- Vanuit de organisatie houdt de bedrijfsmentor toezicht op het proces en de opgeleverde producten. Er is wekelijks overleg tussen de bedrijfsmentor en de student. De bedrijfsmentor is Dhr. G. Scheeres.
- Voor het opstellen van de opdrachtomschrijving is er vanuit school een keurend docent toegewezen. In overleg met de keurend docent wordt de afstudeeropdracht bijgesteld totdat deze voldoet aan de eisen.
- Vanuit school zijn er twee examinatoren die de opgeleverde producten zullen evalueren en beoordelen. De examinatoren zijn op het moment van het schrijven van dit plan van aanpak nog niet toegewezen.

4.2. Informatie

Tijdens de afstudeerperiode is de student 5 dagen per week aanwezig op het afstudeerbedrijf. De hele dag door is de student in de gelegenheid om vragen te stellen aan collega's of aan de bedrijfsmentor. Verder is er wekelijks elke vrijdag ruimte ingepland met de bedrijfsmentor om de voortgang van het project te bespreken.

4.3. Faciliteiten

De volgende faciliteiten zijn nodig voor het uitvoeren van het afstudeerproject.

- Een modern werkstation met voldoende processorcapaciteit en geheugen (2GHz/512MB)
- Internet aansluiting
- Microsoft Windows XP
- Internet Information Services 5.1
- Microsoft Office XP
- Microsoft Visual Studio 2003 .NET
- Adobe Photoshop
- Boeken over ASP.Net, C#, XML en XSL-T

5. Kwaliteitsborging

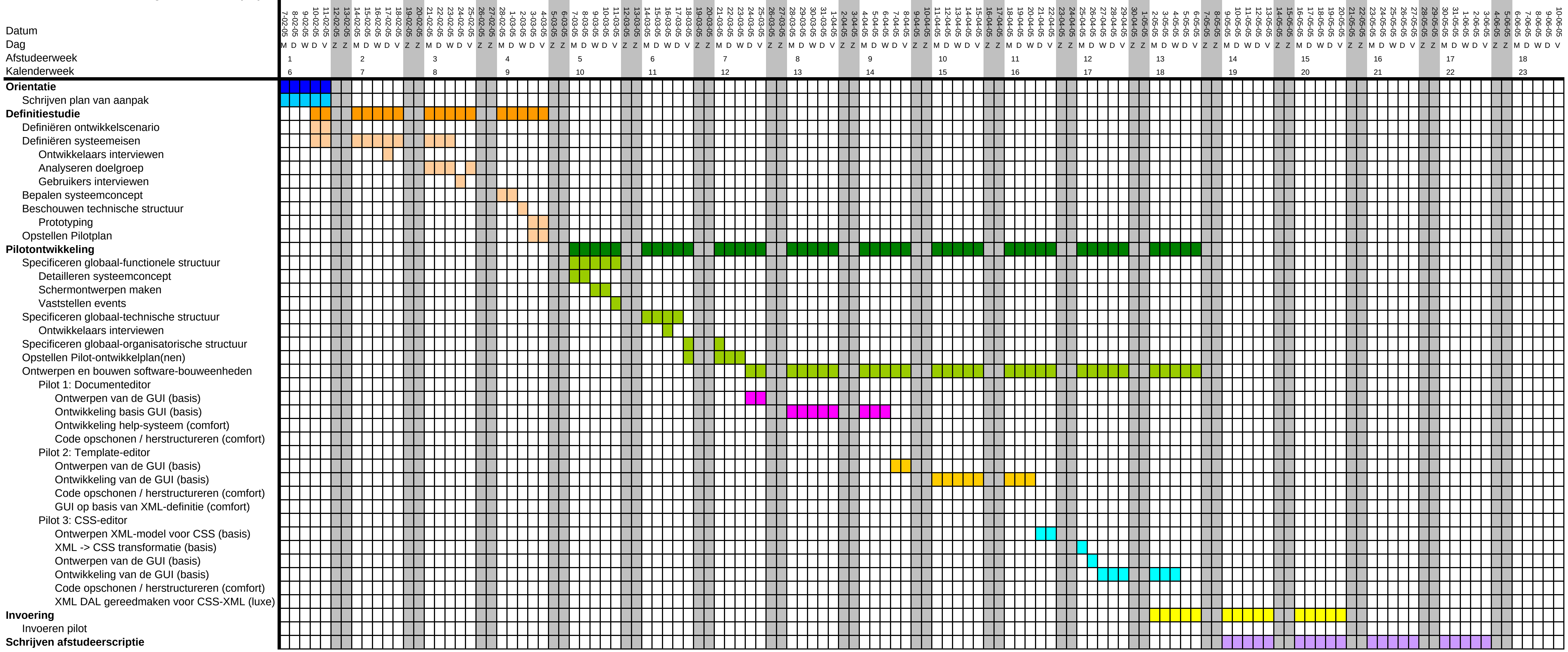
In dit hoofdstuk wordt beschreven aan welke kwaliteitseisen de op te leveren producten moeten voldoen, hoe deze kwaliteit bereikt moet worden en hoe dit kan worden gecontroleerd.

5.1. Kwaliteit

In de volgende tabel staat beschreven welke kwaliteitseisen er worden gesteld aan de verschillende producten, hoe deze kwaliteit kan worden bereikt en hoe kan worden gecontroleerd of aan deze eisen voldaan is.

Product	**Eis**	**Hoe te bereiken**
Documenten	De opgeleverde documenten moeten duidelijk en volledig zijn.	Voordat de documenten worden opgeleverd, worden ze door een buitenstaander gelezen.
	De opgeleverde documenten moeten consistent zijn.	De standaarden uit hoofdstuk 3.2 gebruiken.
Ontwikkelde software	Bestaande gebruikers moeten zelfstandig overweg kunnen met nieuwe functionaliteit zonder eerst een handleiding te hoeven lezen.	Het zorgen voor grafische interfaces zodat gebruikers zich niet bezig hoeven te houden met codes.

Planningsheet afstudeerproject



Definitiestudie

Afstudeeropdracht

student:	Dhr. M. Verheul
keurend docent:	Dhr. J.J. van Beijnum
afstudeerbedrijf:	Marbel Software B.V. te Noordwijkerhout
bedrijfsmentor:	Dhr. Ing. G. Scheeres
datum:	10 juni 2005
afstudeerrichting:	Informatica en Informatiekunde: VIA
afstudeerblok	2005-1.1
versie:	1.2

Inhoudsopgave

1. Inleiding

Dit document is het eindproduct van de eerste fase binnen de ontwikkelmethodiek IAD, de definitiestudie. Het is geschreven in het kader van het afstudeerproject bij Marbel software. Hoofdstuk 2 is het plan van aanpak dat opgesteld is bij aanvang van het project. Later is dit hoofdstuk uitgebreid met de paragraaf systeemconcept. In hoofdstuk 3 wordt de huidige situatie in kaart gebracht en de gebruikers van Spider geanalyseerd. In hoofdstuk 4 worden de systeemeisen opgesteld die precies beschrijven wat het systeem wel en wat het systeem niet mag doen. Hoofdstuk 5 geeft een beschrijving van de gewenste situatie. In hoofdstuk 6 wordt de werking en de structuur van Spider beschreven. Tot slot wordt in hoofdstuk 7 beschreven hoe het project wordt opgesplitst in een viertal pilots en hoe deze worden ontwikkeld.

2. Plan van aanpak

Dit hoofdstuk is aanvankelijk geschreven bij aanvang van het project en is later uitgebreid met paragraaf 2.6 (systeemconcept). Het systeemconcept is de eerste activiteit die is uitgevoerd in de fase definitiestudie van de ontwikkelmethodiek IAD. Dit plan van aanpak dient niet alleen als omschrijving van de IAD-fase ‘definitiestudie’ maar als omschrijving van het gehele project.

2.1. Inleiding

Een plan van aanpak is een document waarin alle afspraken staan tussen de opdrachtgever en de uitvoerende partij, in dit geval de afstudeerder. Het plan van aanpak omvat afspraken met betrekking tot de omvang van de opdracht, de verwachte duur van het project, de te volgen aanpak, de projectinrichting en kwaliteitsborging. Een deel van de informatie in dit plan van aanpak staat ook al beschreven in de opdrachtomschrijving. Dit plan van aanpak gaat dieper in op de opdracht aan de hand van de volgende paragrafen.
Paragraaf 2.2 “Opdrachtomschrijving” bevat een uitvoerige beschrijving van de opdracht en een lijst met op te leveren producten en diensten. Verder worden er de randvoorwaarden en risicofactoren beschreven.
In paragraaf 2.3 “Aanpak” wordt beschreven hoe het project precies uitgevoerd gaat worden. Er wordt overeenstemming bereikt over de te volgen weg naar het eindresultaat.
In paragraaf 2.4 “Projectinrichting” wordt beschreven hoe het project ingericht moet worden om aan de aanpak uit paragraaf 2.3 te kunnen voldoen.
Tenslotte wordt in paragraaf 2.5 “Kwaliteitsborging” beschreven aan welke kwaliteitseisen het resultaat moet voldoen en hoe die kwaliteit bereikt kan worden.

2.1.1. Achtergrond en aanleiding van de opdracht

Marbel Software, opgericht in november 1990, is gespecialiseerd in het ontwikkelen en implementeren van maatwerk (software) systemen, zowel binnen Local Area Networks als Wide Area Networks. Deze systemen kunnen variëren van administratief tot en met technische (datacommunicatie) systemen. Bij het ontwikkelen van deze systemen past Marbel Software moderne technieken zoals XML en SOAP toe.
Spider is een content management systeem (CMS) voor websites dat Marbel Software oorspronkelijk heeft ontwikkeld voor NDLite, de jongerensite van het Nederlands Dagblad. Met Spider is het mogelijk om redacteurs en moderators aan te wijzen die allen verantwoordelijk zijn voor hun eigen content. Verder kunnen bezoekers van de website reageren op berichten en kunnen ze een sms-bericht ontvangen als er belangrijke nieuwe content is geplaatst op de website. Spider is volledig gebaseerd op XML en XSL-T en is geschreven voor het ontwikkelplatform ASP.NET in de programmeertaal C#. Spider wordt nu ook ingezet als CMS bij kleinere websites. Uit de vragen van gebruikers van Spider die geregeld binnenkomen, blijkt dat veel gebruikers moeite hebben met het veranderen van de opmaak en de structuur van een website met Spider. De afstudeeropdracht draait om het verbeteren van Spider zodat het voor gebruikers eenvoudiger wordt om de opmaak en de structuur van een website aan te kunnen passen.

2.2. Opdrachtomschrijving

In dit hoofdstuk wordt de afstudeeropdracht beschreven en worden er randvoorwaarden en risicofactoren gesteld met betrekking tot het afstudeertraject.

2.2.1. Opdrachtgever

De opdrachtgever is Marbel Software. De contactpersoon binnen Marbel Software is Martin Weijers.
De gebruikers zijn klanten van Marbel Software die gebruik maken van Spider.
De opdrachtgever geeft feedback op de opgeleverde documentatie en producten. Binnen het bedrijf wordt er geen methode of werkwijze voorgeschreven voor softwareontwikkelingsprojecten.

2.2.2. Probleemstelling

Regelmatig komt het voor dat gebruikers (gebruikersgroep beheerders) van Spider naar Marbel Software bellen omdat ze de structuur en de opmaak van hun website willen aanpassen.
Zelf kunnen zij dit niet omdat ze niet overweg kunnen met de technieken XML en XSL-T. In Spider is het gebruik van deze technieken nodig om de structuur en de opmaak van een website aan te passen.

2.2.3. Doelstelling opdracht

De doelstelling van de afstudeeropdracht is het verbeteren van Spider zodat gebruikers (gebruikersgroep beheerders) zelfstandig in staat zijn om de structuur en de opmaak van hun website aan te passen zonder ervaring te hoeven hebben met de technieken XML en XSL-T.

2.2.4. Opdrachtformulering

Eerst moet er worden onderzocht waar gebruikers precies problemen ondervinden bij het gebruik van Spider. Als deze problemen in kaart zijn gebracht, kan er een ontwerp worden gemaakt van de te ontwikkelen oplossing. Vervolgens wordt deze oplossing ontwikkeld en geïmplementeerd. Het is bij aanvang van het project nog niet helemaal duidelijk wat voor oplossing er precies ontwikkeld gaat worden. Dit hangt af van de wensen van de opdrachtgever en de gebruikers.

2.2.5. Op te leveren producten en diensten

In het kader van de afstudeeropdracht zullen de volgende producten worden opgeleverd aan de opdrachtgever:

- Plan van Aanpak
- Definitiestudie
- Pilotontwikkelplannen
- Gebruikersdocumentatie
- Systeemdocumentatie
- Volledig werkende GUI voor het ontwerpen / aanpassen van de opmaak en de structuur van de in Spider beheerde website’s.

De eerste drie producten worden gemaakt om het ontwikkeltraject te ondersteunen.
De laatste drie producten vormen samen het uiteindelijke resultaat en leveren een directe bijdrage aan de in paragraaf 2.3 geformuleerde doelstelling.

2.2.6. Randvoorwaarden

De volgende niet-inhoudelijke eisen worden gesteld aan het project:

Tijdspad: De afstudeerperiode loopt van 7 februari 2005 tot en met 10 juni 2005.
Werkwijze: De opdrachtgever stelt geen eisen aan de te volgen werkwijze.
Werkplek: De afstudeerder dient te beschikken over een werkplek met een pc, printer en internet.
Documentatie: De opgeleverde documentatie dient duidelijk en volledig te zijn zodat de opdrachtgever goed op de hoogte is van het proces en de gemaakte keuzes. Aan de hand van de documenten kan het eindproduct beoordeeld worden.
Betrokkenen: Het is van belang dat de betrokkenen (opdrachtgever, ontwikkelaars, klanten) voldoende medewerking verlenen aan het project.

2.2.7. Risicofactoren

De volgende factoren kunnen het slagen van het afstudeerproject in de weg staan:

- Gebrek aan technische kennis bij de afstudeerder
  - Tijdens het afstudeerproject wordt gewerkt met technieken die nog niet geheel bekend zijn bij de afstudeerder (C#, ASP.NET, XSL-T).
  - Om het risico van het optreden van deze factor zo veel mogelijk te beperken, heeft de afstudeerder verschillende boeken aangeschaft om meer te leren over deze nieuwe technieken. Verder kan er op elk moment tijdens het afstudeertraject de hulp van collega's worden ingeroepen.
- Conflicten met de opdrachtgever
  - Tijdens het afstudeerproject zouden conflicten met de opdrachtgever kunnen optreden als er sprake is van onvoldoende communicatie of als de opdrachtgever of afstudeerder zich niet aan de afspraken houdt.
  - De kans op het optreden van deze factor is minimaal als er vanaf beide kanten goed wordt gecommuniceerd en er goede afspraken kunnen worden gemaakt.
- Gebrek aan tijd
  - Er zou een gebrek aan tijd kunnen ontstaan doordat de te verrichten activiteiten onderschat worden of doordat bepaalde activiteiten meer tijd kosten.
  - Om te voorkomen dat deze factor optreedt, wordt er goed nagedacht over de planning in dit plan van aanpak. Verder gaat er gewerkt worden met de techniek time-boxing. Dat houdt in dat binnen een pilot alle deelactiviteiten een prioriteit krijgen. Deze prioriteit wordt toegekend in overleg met de opdrachtgever. Eerst worden de pilotdelen met de hoogste prioriteit ontwikkeld. Op deze manier wordt er in elk geval resultaat behaald voor de opdrachtgever. Verder dient deze planning natuurlijk ook gevolgd te worden.

2.3. Aanpak

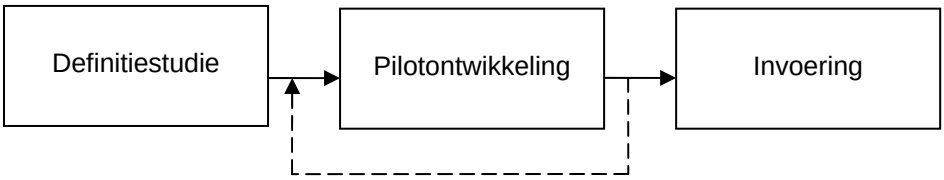
In dit hoofdstuk wordt beschreven hoe het project uitgevoerd gaat worden. Het beschrijft de weg die gevolgd gaat worden naar het gewenste eindresultaat.

2.3.1. Methoden en technieken

De volgende methoden en technieken zullen tijdens de uitvoering van de opdracht toegepast worden.

2.3.1.1. Methoden

Bij het afstudeerproject zal gebruik worden gemaakt van IAD als systeemontwikkelingmethodiek. IAD betekent Interactive / Iterative / Incremental Application Design. ‘Interactive’ betekent dat er ‘samen’ (of in samenspraak) met de gebruikers wordt ontwikkeld. ‘Iterative’ betekent dat er wordt ontwikkeld volgens een vast en herhalend proces. ‘Incremental’ betekent dat bij elke iteratieslag er nieuwe functionaliteit bijkomt die bijdraagt aan een grotere bruikbaarheid van het systeem.



Figuur 1: Incrementeel ontwikkelen (RAD)

Een belangrijke keuze die gemaakt moet worden bij het gebruik van IAD als systeemontwikkeling-methodiek is de te gebruiken iteratiestrategie. Bij dit afstudeerproject zal de iteratiestrategie ‘incrementeel ontwikkelen’ gebruikt worden om de volgende redenen.

- Het systeem kan goed van tevoren worden gespecificeerd.
- De omvang van het systeem is te overzien.
- Stapsgewijze invoering van het systeem is niet mogelijk (afstudeerproject).

In het afstudeerproject is er echter geen ruimte (tijd) voor een iteratieslag. Er is dus eigenlijk niet echt sprake van een iteratief proces. De iteratiestrategie bepaalt echter ook waar de nadruk op wordt gelegd bij het doorlopen van de activiteiten binnen elke fase. Bij dit ontwikkeltraject is er geringe gebruikersparticipatie. Om er toch voor te zorgen dat het eindproduct voldoet aan de wensen van de gebruiker worden er andere technieken ingezet zoals interviews.

2.3.1.2. Technieken

De volgende technieken zullen worden gebruikt

- UML Use-case diagram
- UML Class diagram
- Navigatieschema
- Interviews
- Persona’s
- Timeboxing

2.3.2. Standaards, richtlijnen en procedures

Om te zorgen voor consistentie in de documenten die opgeleverd worden, zijn de volgende richtlijnen opgesteld.

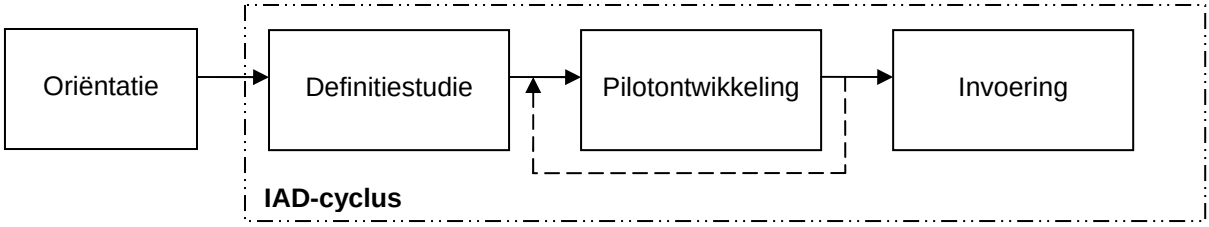
- Elke pagina heeft een koptekst de volgende elementen
  - Logo
  - Titel document
  - Naam auteur
  - Studentennummer
  - Maand, jaar
- Elke pagina heeft een voettekst met daarin het paginanummer en het aantal pagina's
- Na een kop volgt altijd één 'wit'-regel
- Na een paragraaf of hoofdstuk volgen altijd twee 'wit'-regels
- Er wordt gebruik gemaakt van de volgende opmaakprofielen.
- Alle alinea's worden links uitgelijnd
- Tussen een alinea en een opsomming zit altijd één 'wit'-regel

Kop 1: Arial, 12pt, vet.
Kop 2: Arial, 11pt, vet.
Kop 3: Arial, 10pt, vet.
Kop 4: Arial, 9pt, vet.
Bodytekst: Arial, 10pt.
Bijschrift: Arial 8pt, vet.

2.3.3. Werkzaamheden

Het afstudeerproject wordt uitgevoerd volgens de IAD-systeemontwikkelmethodiek. Daarom wordt grotendeels dezelfde fasering als bij IAD gehanteerd. Het afstudeerproject bestaat uit de volgende fases.

0. Oriëntatie
1. Definitiestudie
2. Pilotontwikkeling
3. Invoering



Figuur 2: IAD cyclus uitgebreid met oriëntatie-fase

De oriëntatiefase bestaat niet in binnen IAD. Deze fase is toegevoegd om de voorbereidende werkzaamheden in onder te brengen zoals het schrijven van dit plan van aanpak.
Binnen elke fase worden een aantal activiteiten uitgevoerd. De volgende activiteiten worden in chronologische volgorde uitgevoerd.

- **Oriëntatie**
  - Opstellen plan van aanpak
  - Opstellen planning
- **Definitiestudie**
  - Definiëren ontwikkelscenario
  - Definiëren systeemeisen
    - Interviewen van de ontwikkelaars om een beeld te krijgen van de vragen die gebruikers stellen
    - Analyseren van de doelgroep (gebruikers)
    - Interviewen van de gebruikers om hun wensen in kaart te brengen
  - Bepalen systeemconcept
    - Globaal beschrijven van de beoogde oplossing op basis van de systeemeisen
  - Beschouwen technische structuur
    - Onderzoeken van de haalbaarheid door middel van experimentele proto-typing
  - Opstellen pilotplan
- **Pilotontwikkeling**
  - Specificeren globaal-functionele structuur pilot
    - Detailleren van het systeemconcept
    - Maken van schermontwerpen
    - Vaststellen events
  - Specificeren globaal-technische structuur pilot
    - Interviewen van de ontwikkelaars om alle opties van Spider in kaart te brengen
  - Specificeren globaal-organisatorische inrichting
  - Opstellen pilotontwikkelplan
  - Ontwerpen en bouwen software-bouweenheden
  - Samenstellen handleidingen
- **Invoering**
  - Invoeren pilot (integreren binnen Spider)
  - Verzamelen feedback

2.3.4. Planning

Bij dit afstudeerproject wordt gebruik gemaakt van time-boxing. Time-boxing is een planningstechniek waarbij de activiteiten worden opgesplitst in zo klein mogelijke eenheden. Als bijlage zijn planningsheets opgenomen die gebruikt zullen worden met de techniek time-boxing. Op het moment van het schrijven van dit plan van aanpak is nog niet bekend welke software-bouweenheden er zullen zijn. De software-bouweenheden worden nader gespecificeerd aan het begin van de fase pilotontwikkeling.

2.4. Projectinrichting

In dit gedeelte wordt beschreven hoe het project organisatorisch wordt ingericht.

2.4.1. Projectorganisatie

De student is geheel zelf verantwoordelijk voor het resultaat van het afstudeerproject. Wel zijn er vanuit de organisatie en vanuit school een aantal betrokkenen.

- Vanuit de organisatie houdt de bedrijfsmentor toezicht op het proces en de opgeleverde producten. Er is wekelijks overleg tussen de bedrijfsmentor en de student. De bedrijfsmentor is Dhr. G. Scheeres.
- Voor het opstellen van de opdrachtomschrijving is er vanuit school een keurend docent toegewezen. In overleg met de keurend docent wordt de afstudeeropdracht bijgesteld totdat deze voldoet aan de eisen.
- Vanuit school zijn er twee examinatoren die de opgeleverde producten zullen evalueren en beoordelen. De examinatoren zijn op het moment van het schrijven van dit plan van aanpak nog niet toegewezen.

2.4.2. Informatie

Tijdens de afstudeerperiode is de student 5 dagen per week aanwezig op het afstudeerbedrijf. De hele dag door is de student in de gelegenheid om vragen te stellen aan collega's of aan de bedrijfsmentor. Verder is er wekelijks elke vrijdag ruimte ingepland met de bedrijfsmentor om de voortgang van het project te bespreken.

2.4.3. Faciliteiten

De volgende faciliteiten zijn nodig voor het uitvoeren van het afstudeerproject.

- Een modern werkstation met voldoende processorcapaciteit en geheugen (2GHz/512MB)
- Internet aansluiting
- Microsoft Windows XP
- Internet Information Services 5.1
- Microsoft Office XP
- Microsoft Visual Studio 2003 .NET
- Adobe Photoshop
- Boeken over ASP.Net, C#, XML en XSL-T

2.5. Kwaliteitsborging

In dit hoofdstuk wordt beschreven aan welke kwaliteitseisen de op te leveren producten moeten voldoen, hoe deze kwaliteit bereikt moet worden en hoe dit kan worden gecontroleerd.

2.5.1. Kwaliteit

In de volgende tabel staat beschreven welke kwaliteitseisen er worden gesteld aan de verschillende producten, hoe deze kwaliteit kan worden bereikt en hoe kan worden gecontroleerd of aan deze eisen voldaan is.

Product	Eis	Hoe te bereiken
Documenten	De opgeleverde documenten moeten duidelijk en volledig zijn.	Voordat de documenten worden opgeleverd, worden ze door een buitenstaander gelezen.
	De opgeleverde documenten moeten consistent zijn.	De standaarden uit hoofdstuk 3.2 gebruiken.
Ontwikkelde software	Bestaande gebruikers moeten zelfstandig overweg kunnen met nieuwe functionaliteit zonder eerst een handleiding te hoeven lezen.	Het zorgen voor grafische interfaces zodat gebruikers zich niet bezig hoeven te houden met codes.
	De code die geschreven gaat worden om de nieuwe functionaliteit te bereiken, dient correct te zijn en moet aansluiten bij de bestaande architectuur.	Voordat er ontwikkeld gaat worden, dient er overleg plaats te vinden met de ontwikkelaars om afspraken te maken over de applicatiearchitectuur.
	De code die geschreven gaat worden dient consistent te zijn.	De code dient geschreven te worden volgens de Microsoft Internal Coding Guidelines[1].

Figuur 3: Kwaliteitseisen

[1] Zoals beschreven op http://blogs.msdn.com/brada/articles/361363.aspx

2.6. Ontwikkelscenario

In deze paragraaf die later is toegevoegd aan het plan van aanpak wordt het ontwikkelscenario gedefinieerd. Dit is de eerste activiteit in de fase definitiestudie van IAD. Het belangrijkste doel van deze activiteit is het vast stellen van de reikwijdte en de context van het ontwikkelproject.

2.6.1. Doelstellingen en reikwijdte van het project

De doelstelling van dit project is het verbeteren van Spider zodat gebruikers (gebruikersgroep beheerders) zelfstandig in staat zijn om de structuur en de opmaak van hun website aan te passen zonder ervaring te hoeven hebben met de technieken XML en XSL-T. Het project heeft betrekking op het verbeteren van het gebruiksgemak en niet op het toevoegen van nieuwe mogelijkheden aan Spider. Uit een eerder rapport is al gebleken dat gebruikers van Spider problemen zouden kunnen ondervinden bij het inrichten van een CSS-stylesheet, een document, een template en een style. Dit zijn activiteiten binnen Spider die nu moeten gebeuren door middel van het intikken van code. Door middel van interviews met gebruikers en ontwikkelaars wordt eerst vastgesteld of deze activiteiten inderdaad de grootste knelpunten zijn voor de gebruikers. Als dit zo is, zullen hier GUI's voor ontwikkeld worden.

2.6.2. Technische structuur

Het ontwikkelproject betreft een aanvulling op bestaande software (Spider). Als basis wordt dan ook de technische structuur van Spider gebruikt. Deze structuur is op het moment van het schrijven van dit systeemconcept nog niet op alle punten bekend. Door middel van interviews met de ontwikkelaars van Spider zal de technische structuur verder worden vastgesteld. Wel is al bekend dat Spider ontwikkeld is in de taal C# op het platform ASP.NET.

2.6.3. Cruciale succesfactoren

De volgende succesfactoren zijn cruciaal voor het slagen van het project.

- Het opstellen van een goede planning en het handhaven van deze planning
- Goede communicatie tussen alle betrokkenen (afstudeerder, bedrijfsmentor, ontwikkelaars, gebruikers)
- Uitbreiding van de technische kennis van de afstudeerder.

2.6.4. Verwachte impact op de gebruikersorganisatie

Het project omvat verbeteringen op het gebied van gebruiksvriendelijkheid. Verwacht wordt daarom dat er weinig of geen bijscholing hoeft plaats te vinden bij de gebruikers bij invoering van het project. Impact op de gebruikersorganisatie zal dus zijn dat er minder tijd nodig is voor gebruikers om een website in te richten en aan te passen in Spider.

2.6.5. Kenmerken van het ontwikkelproces

Omdat het project een afstudeeropdracht is die wordt uitgevoerd door één persoon, ziet het ontwikkelproces er anders uit dan bij projecten die worden ontwikkeld in teamverband. Het ontwikkelproces van dit project heeft de volgende kenmerken.

- Iteratie
  - Zoals al eerder in dit plan van aanpak is toegelicht, zal er gebruik gemaakt worden van de iteratiestrategie 'incrementeel ontwikkelen'. Er zullen echter geen iteratieslagen plaatsvinden. De keuze voor deze iteratiestrategie richt zich dus meer op de activiteiten waar tijdens dit project nadruk op worden gelegd.
- Workshops
  - Omdat er sprake is van een relatief klein ontwikkeltraject en er geringe gebruikersparticipatie is, zal er gebruik worden gemaakt van interviews in plaats van workshops.
- Teams
  - Het project wordt zelfstandig uitgevoerd. Er zal dus niet in teams worden gewerkt.
- Time-boxing
  - Één van de cruciale succesfactoren is het opstellen en hanteren van een goede planning. Daarom is er een planning opgesteld die gebruikt zal worden met de techniek time-boxing. De planning is als bijlage opgenomen bij dit plan van aanpak.
- Parallellisme
  - Tijdens dit project zal er gebruik gemaakt worden van parallellisme. In hoofdstuk 7 wordt duidelijk hoe parallellisme wordt toegepast.

2.6.6. Eerste indicatie pilot-strategie

Er van uitgaande dat de uitkomsten uit een eerder rapport kloppen (dit wordt geverifieerd door middel van interviews) kan de volgende pilot-indeling worden gemaakt:

- Pilot 1: GUI voor document-editor (XML)
- Pilot 2: GUI voor template-editor (XML)
- Pilot 3: GUI voor CSS-editor (CSS/XML)
- Pilot 4: GUI voor style-editor (XSL-T)

Elke pilot omvat een GUI voor een bepaald programma-onderdeel. Pilot 1 en 2 hebben veel overeenkomsten met elkaar omdat er in beide pilots door middel van een GUI een bepaald XML-fragment moet worden geschreven. Deze pilots komen daarom in aanmerking voor parallelle ontwikkeling.

2.6.7. Hergebruik

Bij dit project zal op de volgende punten gebruik worden gemaakt van hergebruik.

- Pilot 1 en 2: Document- en template-editor
  - De document- en template-editor lijken qua techniek en functionaliteit op elkaar. Bij beide pilots moet er een grafische interface ontwikkeld worden om een XML-document volgens een bepaald XML-dialect te genereren. Als functies voor deze editors globaal worden gedefinieerd en generiek worden geschreven, kunnen ze vanuit beide programmaonderdelen worden aangeroepen.
- Pilot 3: CSS-editor
  - Tijdens een eerder project is al een GUI voor de CSS-editor van Spider gemaakt. Deze editor is echter nooit in gebruik genomen. Bepaalde onderdelen en concepten van deze editor zouden gebruikt kunnen worden in pilot 4. Wel dient de programmacode herschreven te worden aangezien er ontwikkeld is in VB.NET en er in dit project geprogrammeerd gaat worden met C#.

2.6.8. Conversiestrategie

Bij dit project zal geen gebruik gemaakt worden van gegevensconversies. Zoals beschreven in de vorige paragraaf zal pilot 4 gedeeltelijk gebaseerd zijn op een eerder ontwikkelde GUI. Technisch gezien is het mogelijk om VB-code te converteren naar C#-code. Hier wordt echter geen gebruik van gemaakt omdat deze GUI behoorlijk onoverzichtelijk geschreven is.

2.6.9. Test- en acceptatiestrategie

Een groot deel van de testen zullen plaatsvinden tijdens de ontwikkeling van de software-bouweenheden. Hierbij zal gebruik worden gemaakt van de zogenaamde white-box testtechniek. Whitebox-testen houdt in dat alle programmadelen op het kleinste niveau getest wordt. Hierbij moet gedacht worden aan elke lus, elk event en elke variabele. Er is gekozen voor deze testtechniek omdat de globale werking van Spider al getest is tijdens de initiële ontwikkeling van Spider.
Na de invoering zal er bovendien ook nog een globale test uitgevoerd worden waarbij een bepaald scenario uitgevoerd moet worden. Er wordt dan gekeken of er problemen optreden bij het uitvoeren van dit scenario.

3. Beschrijving huidige situatie

In dit hoofdstuk wordt de huidige situatie beschreven de verschillende soorten gebruikers in kaart gebracht en hun taken geanalyseerd. Omdat er geen gebruikersobservatie is gedaan, kan er geen taakanalyse worden gemaakt.

3.1. Over Spider

De doelstelling van de afstudeeropdracht is het verbeteren van Spider zodat gebruikers (gebruikersgroep beheerders) zelfstandig in staat zijn om de structuur en de opmaak van hun website aan te passen zonder ervaring te hoeven hebben met de technieken XML en XSL-T. Het is dus belangrijk om te beseffen dat de bezoekers van de website die beheerd wordt in Spider NIET tot de gebruikers worden gerekend.

3.2. Toepassingen van Spider

Spider is oorspronkelijk ontwikkeld als CMS met redactionele mogelijkheden voor nieuws-websites. Doordat de structuur van Spider zeer open is, kan het vrij eenvoudig worden uitgebreid met extra functionaliteit waardoor het bij meer organisaties ingezet kan worden. Op dit moment wordt Spider bij de volgende soort organisaties gebruikt.

Huidige toepassingen van Spider	
Online krant / community	Spider is oorspronkelijk ontwikkeld voor NDLite, de jongerensite van het Nederlands Dagblad. Alle functionaliteit die Spider biedt, wordt gebruikt bij dit type gebruiker. Er worden nieuwsberichten geplaatst door verschillende webredacteuren waarop gereageerd kan worden door de bezoekers die zich hebben aangemeld bij de website.
Informatievoorziening binnen een sportvereniging	Sinds december 2004 wordt Spider ook gebruikt voor de website van de voetbalvereniging Foreholte. Het is de bedoeling dat de website op den duur het clubblad gaat vervangen. De website heeft dus vooral een informatieve functie met wedstrijdverslagen, afgelastingen en andere belangrijke mededelingen.
Website voor kleine bedrijven	Tot slot wordt Spider gebruikt bij kleinere bedrijven die zich graag willen presenteren op internet en controle willen hebben op de inhoud van de website.

Figuur 4: Verschillende soorten toepassingen van Spider op dit moment

Daar kunnen in de toekomst meer soorten organisaties bijkomen. Momenteel wordt er gedacht aan de ontwikkeling van een modulair systeem, waardoor de functionaliteit van Spider eenvoudig kan worden uitgebreid. De volgende toepassingen zouden dan gerealiseerd kunnen worden.

Mogelijke nieuwe toepassingen van Spider	
Webwinkel	Als Spider uitgebreid wordt met een webwinkel-module, zou het breder ingezet kunnen worden. De verwachting is dat zo'n module in de nabije toekomst ontwikkeld gaat worden.
Workflow management systeem	Vanwege de open structuur van Spider en het gebruik van XML Web Services, is integratie met andere software goed te realiseren. Een workflow management systeem zou dan een mogelijkheid kunnen zijn. Hier is op het moment echer geen vraag naar en zal dus voorlopig niet worden ontwikkeld.

Figuur 5: Mogelijke toepassingen van Spider in de toekomst

3.3. Beschrijving gebruikersgroepen en hun karakteristieken

Voordat er gezocht kan worden naar een oplossing voor het probleem dat geschetst is in paragraaf 2.2.2, moet er eerst meer bekend zijn over de ‘gebruiker’. Binnen Spider zijn de volgende gebruikersgroepen te onderscheiden.

- **Redacteurs**
- Hoofdredacteurs
- Marketing medewerkers
- **Moderators**
- Opmaak redacteurs
- **Site beheerders**
- Winkel beheerders

De vetgedrukte gebruikersgroepen zijn momenteel het belangrijkst. De overige groepen hebben allemaal overlapping van taken of worden (nog) niet gebruikt in Spider en zullen hier niet besproken worden.
De belangrijkste gebruikersgroepen worden toegelicht in de volgende tabel.

Gebruikersgroepen binnen Spider	
Redacteurs	Een redacteur schrijft artikelen die door een moderator geplaatst kunnen worden op de website. Bij het uitvoeren van zijn taak hoeft de redacteur geen specifieke technische kennis te hebben van HTML. Voor het toevoegen of wijzigen van artikelen kan een ingebouwde HTML-editor worden gebruikt.
Moderators	Een moderator keurt artikelen van redacteurs en/of reacties van bezoekers en beslist of een artikel of reactie geplaatst wordt op de website. Zonodig kan een moderator het artikel nog aanpassen voordat hij het plaatst. De moderatiefunctie kan ook worden uitgeschakeld.
Opmaak redacteurs	De opmaakredacteur heeft rechten om CSS-instellingen aan te passen om de opmaak van de website te veranderen.
Site beheerders	De beheerder van de website heeft alle rechten die de andere gebruikers hebben. Daarnaast kan de beheerder de vormgeving en/of structuur aanpassen en nieuwe pagina’s toevoegen. Het aanpassen van de opmaak van de website gebeurt in de CSS-gedeelte. De beheerder dient hiervoor wel de nodige kennis van CSS te hebben omdat hij de gehele stylesheet zelf in moet typen. Het toevoegen van een nieuwe pagina met nieuwe functionaliteit zijn ook taken die onder de verantwoordelijkheid van de beheerder vallen. Op dit moment is het bewerken van een documentdefinitie of een template echter nog een complex proces, aangezien de beheerder zelf XML-codes in moet voeren. Daarom schakelt een beheerder van een website nu meestal Marbel Software in als hier wijzigingen in moeten worden aangebracht.

Figuur 6: Gebruikersgroepen binnen Spider

3.2.1. Persona's

Om goed in te kunnen spelen op de wensen van de gebruikers in de hierboven gespecificeerde organisaties, moet je kunnen inleven in elke type gebruiker. Daarom worden hieronder een aantal persona's gespecificeerd. Het maken van persona's is een hulpmiddel dat de auteur en software-ontwerper Alan Cooper heeft ontwikkeld op basis van een oude techniek uit de reclamewereld. Om hun producten beter aan de man te kunnen brengen gebruikten marketingmensen de demografische gegevens die ze hadden over hun doelgroep. Daarmee probeerden ze een bepaald stereotype neer te zetten die stond voor de doelgroep die ze wilden bereiken. Ze bouwden op die manier een identificatie op met hun doelgroep, waardoor ze beter konden zien wat aan zou sluiten bij de verwachtingen van de persoon die ze wilden bereiken.

Henk van Lierop

Henk (39) is sinds 2 jaar eindredacteur van de internetredactie van een groot landelijk dagblad. Henk heeft een goede baan want sinds dat hij deze functie heeft, kon zijn vrouw Lia minder gaan werken om meer tijd met de kinderen door te brengen. Voorheen werkte Henk als freelance publicist. Hij verkocht zijn artikelen aan allerlei opiniebladen. Omdat Henk steeds vaker onderzoek moest verrichten voor de artikelen die hij schreef, raakte hij in aanraking met internet. Hij gebruikte internet toen vooral voor het zoeken naar specifieke informatie en het versturen van email. Omdat Henk meer zekerheid wilde, heeft hij gesolliciteerd naar een vaste aanstelling bij het dagblad waar hij nu werkt. Zelf schrijft Henk nu niet zoveel meer. Hij heeft veel meer een coördinerende rol en houdt zich bezig met het controleren van de artikelen van de redacteurs. Henk is van mening dat een goede krant meegaat met zijn tijd. Zowel in vorm als in inhoud. Ook vindt hij dat je lezers alleen kan binden als je vernieuwend blijft. Daarom experimenteert Henk regelmatig met de lay-out van de digitale krant.

Simone van Bleystaede

Simone (24) is net afgestudeerd aan de school voor journalistiek in Zwolle. Vanaf haar 12e schrijft ze al stukjes voor de schoolkrant. Toen wist ze al dat ze journalist zou worden. Nu heeft ze dan eindelijk dat diploma waar ze 4 jaar voor geknokt heeft. Gelukkig kon ze meteen aan de slag bij het plaatselijke regionale dagblad. Daar schrijft ze nu stukjes voor. Van Internet wist Simone nog niets toen ze begon aan haar opleiding. Nu worden de artikelen die Simone voor de West-Friesche Courant schrijft zelfs gepubliceerd op internet. Simone denkt in zinnen en taal is haar leven. Van typografie wil ze niets weten. Gelukkig hoeft ze zich daar niet mee bezig te houden op haar werk. Daar zorgt de typograaf voor.

Cees van Amerongen

Als zelfstandig ondernemer heeft Cees het er maar druk mee. Toen hij 5 jaar geleden zijn bedrijfje NeatHeat begon, had hij kunnen bedenken dat de zaken nu zo goed gaan. Als importeur van centrale verwarmingselementen is Cees voordurend onderweg van leverancier naar klant. Cees, die nu 30 is, zit voor zijn werk vaak in het buitenland en heeft er daarom bewust voor gekozen om voorlopig vrijgezel te blijven. Cees denkt er over na om een website te maken. De domeinnaam heeft hij al geregistreerd. Maar omdat Cees slechte verhalen gehoord heeft van kennissen over webdesigners die voor duizenden euro's een mooie website maken die vervolgens niet aan te passen is, wacht hij nog even met de inrichting van de website.

Bovenstaande persona's kunnen helpen om beter inzicht te krijgen in wat er leeft bij de verschillende soorten gebruikers van Spider. Wat wel moet worden beseft, is dat de persona's geen echte gebruikers zijn. Om te weten te komen of een bedachte oplossing echt aan zal slaan bij een gebruiker, zal toch naar de mening van een echte gebruiker gevraagd moeten worden.

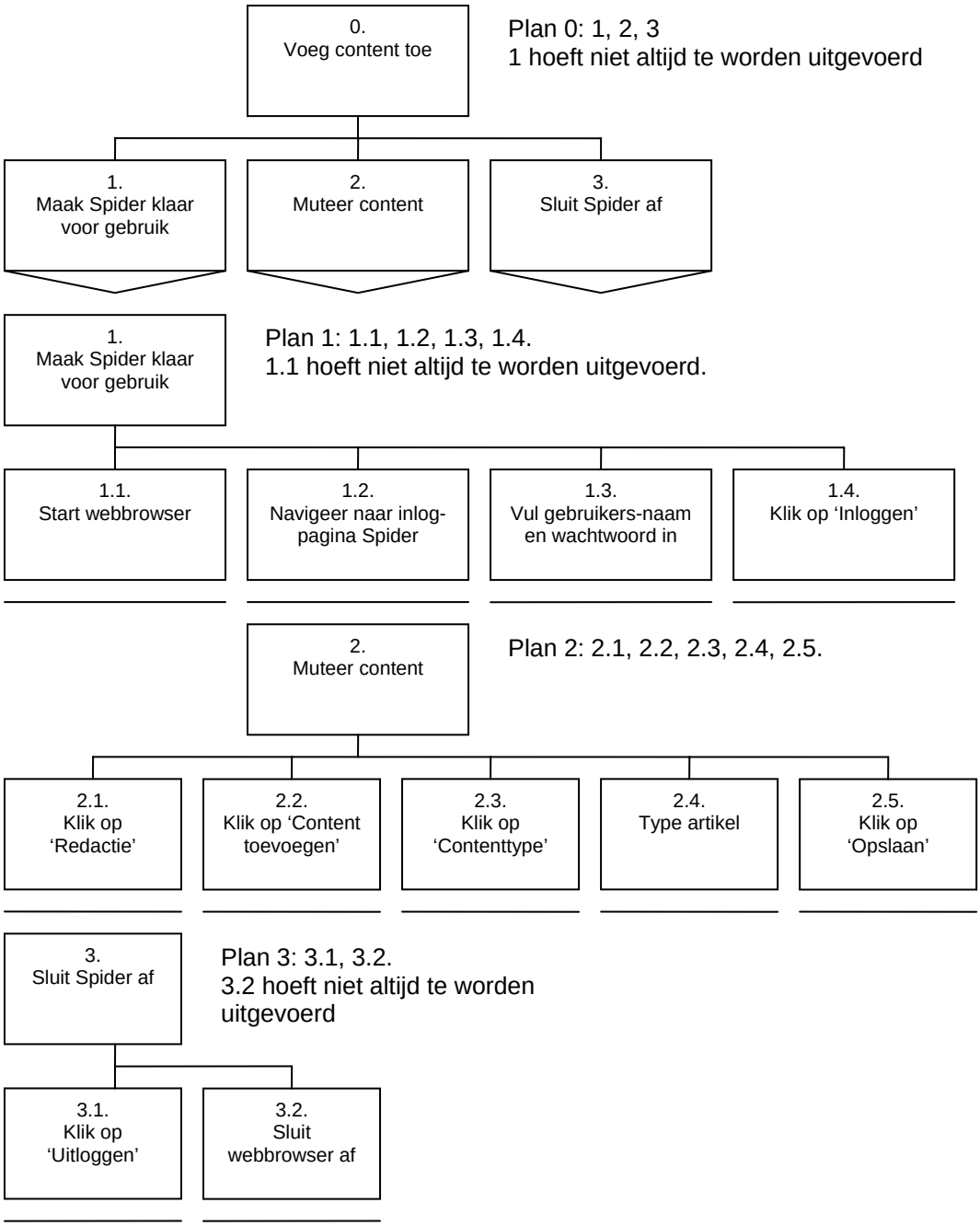
3.3. Taakdiagrammen huidige situatie

In deze paragraaf worden de taken van de gebruiker, zoals deze nu worden uitgevoerd met Spider, in kaart gebracht door middel van taakdiagrammen. De taakdiagrammen zijn onderverdeeld per gebruikersgroep. De gebruikersgroep opmaak redacteurs is weggelaten omdat de taak verander opmaak al is opgenomen bij de gebruikersgroep site beheerders.

3.3.1. Gebruikersgroep redacteurs

De volgende redacteurstaken kunnen worden onderscheiden.

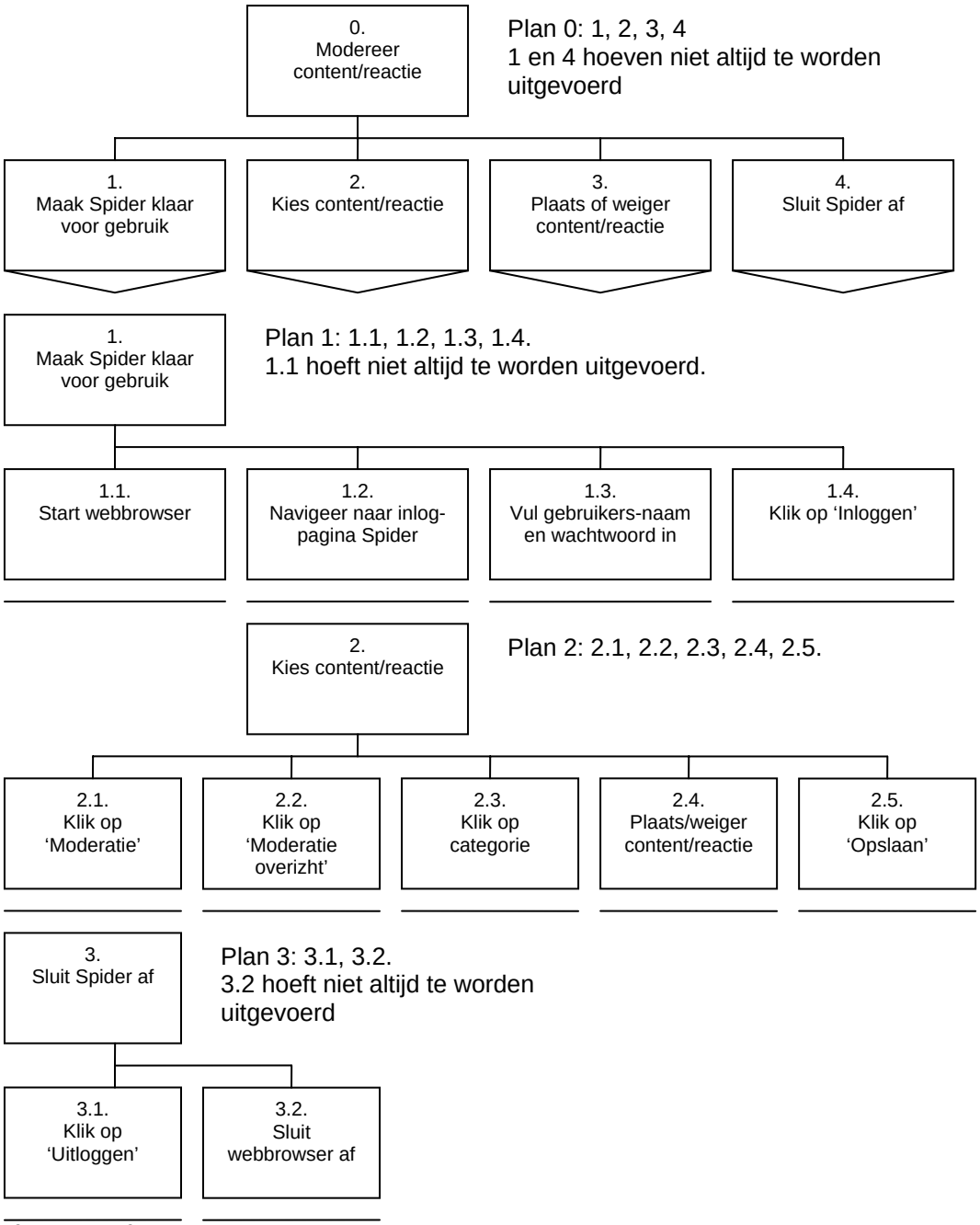
- Content toevoegen



Figuur 7: Content toevoegen

3.3.2. Gebruikersgroep moderators

De moderators kunnen content of reacties modereren. Dit proces gaat exact hetzelfde in zijn werk.

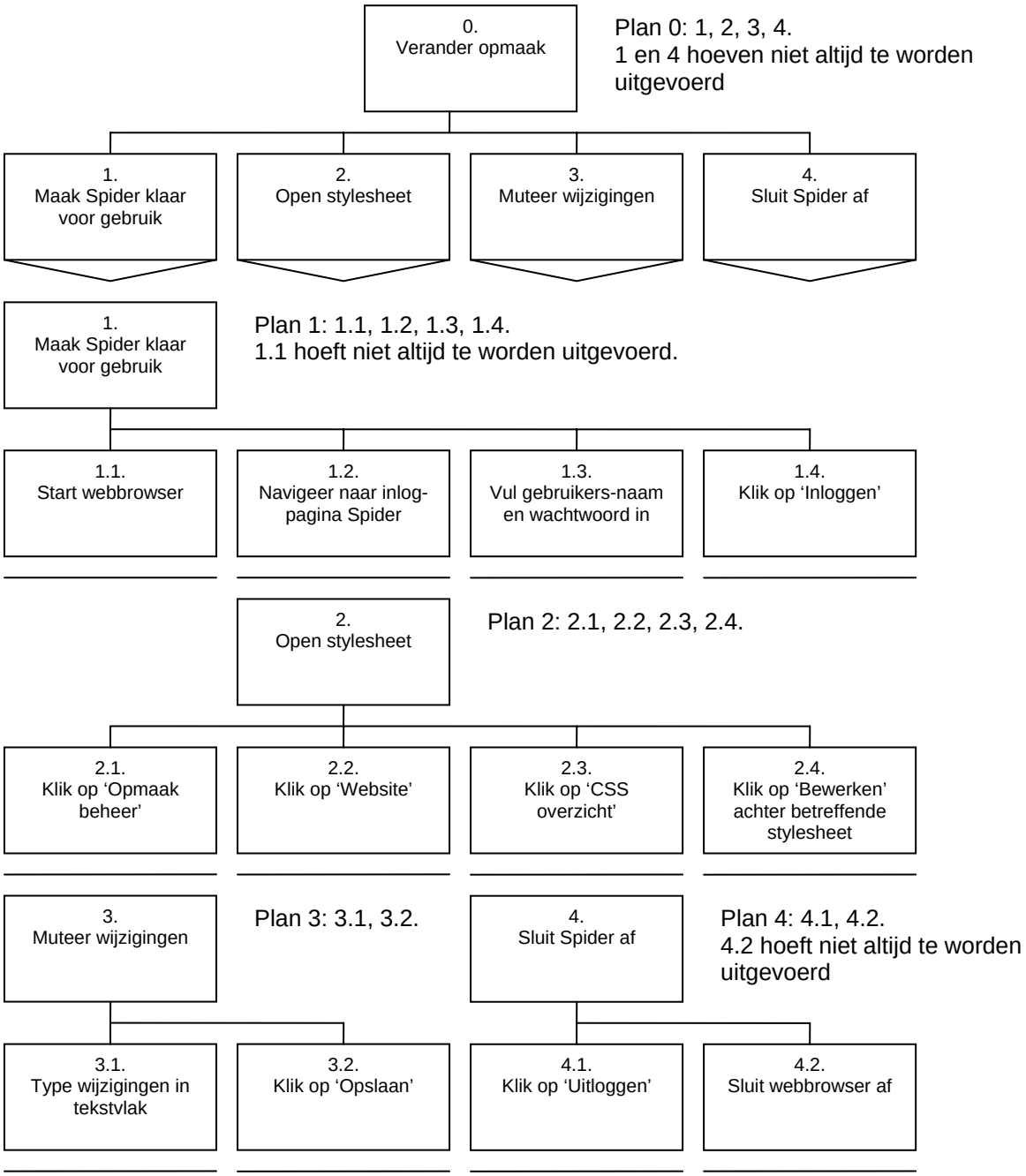


Figuur 8: Modereren van content

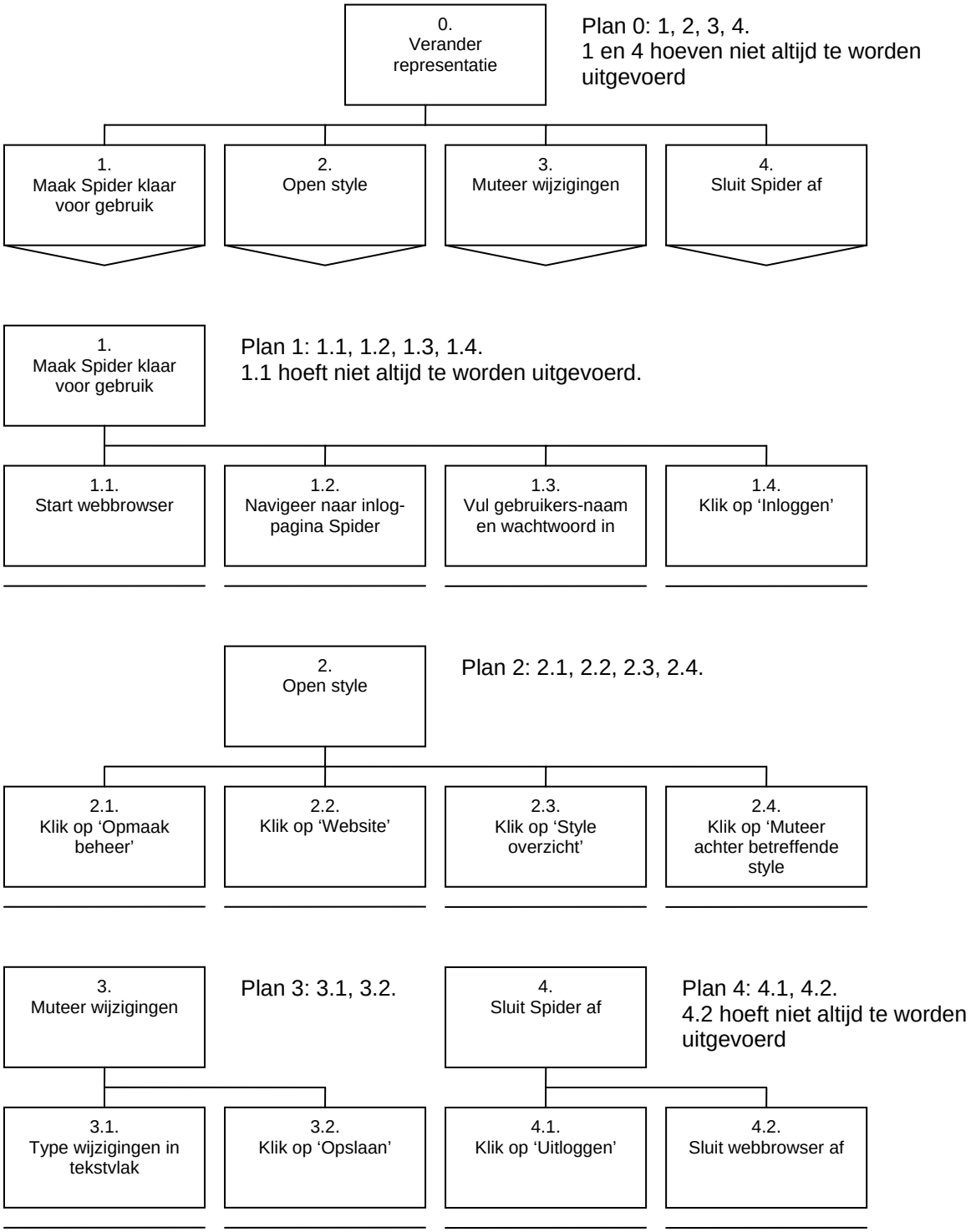
3.3.2. Gebruikersgroep site-beheerders

De volgende site-beheerderstaken zijn te onderscheiden.

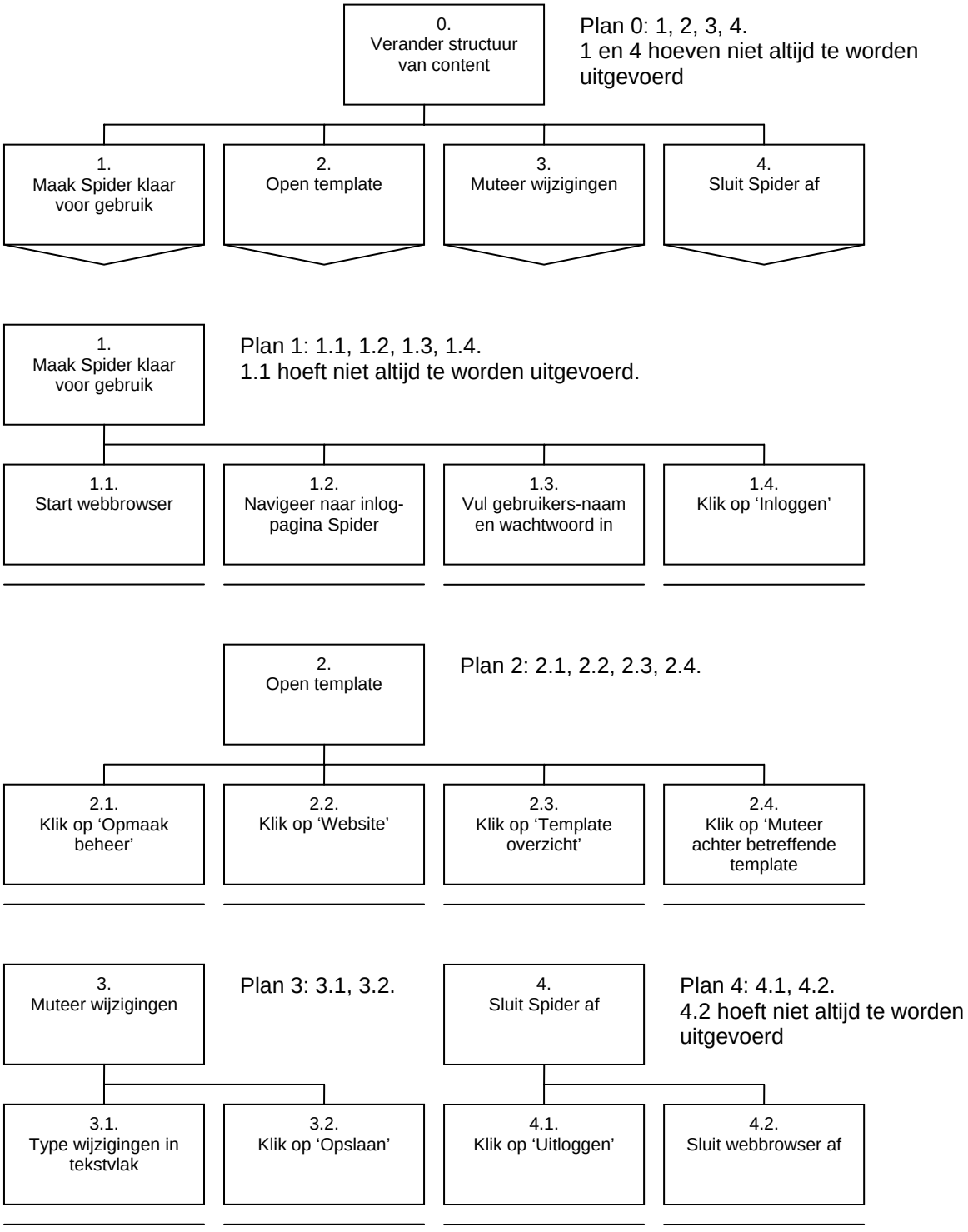
- Opmaak veranderen
- Representatie van content veranderen
- Structuur van content veranderen



Figuur 9: Taakdiagram Opmaak veranderen (CSS)



Figuur 10: Representatie van content veranderen (XSL-T)



Figuur 11: Structuur van content veranderen (XML)

Uit de bovenstaande taakdiagrammen valt te concluderen dat de drie site-beheerstaken momenteel erg veel op elkaar lijken. De taken zijn moeilijk op te splitsen in kleinere stukken. Het daadwerkelijke muteren van de stylesheet, style of template gebeurt door middel van het intikken van codes in een groot tekstvlak. Voor een gebruiker die zelf niet werkzaam is in de ICT is het dus eigenlijk onmogelijk om deze zaken aan te passen.

3.4. Interview met de ontwikkelaars van Spider

Het volgende interview is afgenomen bij de twee ontwikkelaars van Spider, Sinan Yildiz en Michael Kroes, om enerzijds de systeemeisen te kunnen opstellen en anderzijds meer te weten te komen over de technische structuur.

3.4.1. Vragen over gebruikers

1. Ik heb begrepen dat het nog wel eens voorkomt dat gebruikers vragen stellen over de werking van Spider of een verzoek hebben om nieuwe functionaliteit toe te voegen. Hoe vaak komt dit voor?

   Michael: Best wel vaak. Zo'n 2 keer per week ongeveer. Bij een hoop van de gevallen gaat het om vragen waarbij de functionaliteit die ze zoeken al in Spider zit, maar dat ze niet weten hoe ze het moeten implementeren.

2. Kan je voorbeelden geven van die vragen?

   Michael: Er staat nu bijvoorbeeld een verzoekje open van Nedag (klant) om ervoor te zorgen dat alleen geregistreerde gebruikers kunnen reageren op content. Of misschien weet je nog wel bij Foreholte dat het de bedoeling was om een opsomming te geven van al het nieuws op de voorpagina. De functionaliteit was al aanwezig maar het zat alleen in mijn hoofd.
   Sinan: Het nadeel is ook gewoon dat er geen handleiding is waarin alle functionaliteit beschreven wordt.

3. Bepaalde taken binnen Spider zoals het veranderen van de opmaak (CSS) en de structuur (XML/XSL-T) is redelijk technisch. Voor wat voor soort gebruiker is die functionaliteit bedoeld?

   Michael: Het is bedoeld voor de site administrators maar in veel gevallen kunnen ze er niet mee uit de voeten. Het zijn dingen die wij dan voor ze implementeren en daar genereren we dan ook uren voor. Maar als je bijvoorbeeld kijkt naar een CSS-editor waar we het al wel eens over gehad hebben, het zou mooi zijn als je via zo'n editor wat van de functionaliteit terug kan geven naar de gebruiker.
   Bij veel vragen die we krijgen gaat het om hele eenvoudige taken waar wij eigenlijk helemaal geen trek in hebben zoals een documentje toevoegen, kleuren aanpassen en dat soort dingen. Als daar een oplossing voor wordt gemaakt, zou het ons erg ontlasten.

4. Over welke onderdelen van Spider ben je tevreden?

   Michael: De flexibiliteit, het kan eigenlijk overal op ingezet worden. Niet alleen op websites. Denk aan fax-gateways of een workflow-systeem. Met een paar kleine aanpassingen zou het zo geïmplementeerd kunnen worden. De content-codebase is gewoon goed gebouwd en daar kan je van profiteren. Of bijvoorbeeld RSS, dat ga ik binnenkort ook implementeren. Het is waarschijnlijk alleen even een XSL-T style genereren en Spider spuugt RSS uit. Verder ben ik wel tevreden over het hele profile-systeem. Daarbij komen controls zoals textbox-jes op een pagina terecht doordat ze als aspx-component via een XSL-T worden gespecificeerd.

5. Over welke onderdelen van Spider ben je minder tevreden?

   Michael: Ja, toch eigenlijk ook wel de functionaliteit

   Door de bomen het bos niet meer kunnen zien?

   Michael: Inderdaad, de functionaliteit is echt enorm, terwijl de code op zich voor ons prima onderhoudbaar is. Als er dingen echt gewijzigd moeten worden is het meestal een wijziging aan de content toevoegen-pagina, afgezien van het toevoegen van functionaliteit voor een specifiek contentitem.

6. Heb je bepaalde ideeën in je hoofd over nieuwe functionaliteit in Spider? Welke?

   Michael: Ja, iets wat in de toekomst misschien interessant zou zijn is soort webwinkel mogelijkheid. Er is nu nog geen behoefte aan bij klanten, maar het zou heel goed geïmplementeerd kunnen worden binnen de bestaande structuur van Spider. Als ik het zou moeten ontwikkelen, dan gaat het waarschijnlijk een soort plugin worden die klanten dan extra aan zouden kunnen schaffen.

7. Spider wordt nu ingezet als CMS bij een krant, een voetbalvereniging en bij wat kleine bedrijven. Zie je mogelijkheden waardoor het aantal bedrijven en/of het soort bedrijven uitgebreid kan worden? Welke?

   Michael: Ja, wat ik net zei van die webwinkel-plugin. Het maakt niet uit hoe groot je website is, op zich kan Spider alles. Als er vraag naar is, kan het ontwikkeld worden.

3.4.2. Vragen over de technische structuur

De oplossing die ontwikkeld gaat worden, betreft een aanvulling op een bestaand product. Daarom is het belangrijk dat het precies duidelijk is hoe Spider technisch in elkaar zit. Het volgende interview heb ik afgenomen de ontwikkelaars van Spider, Sinan Yildiz en Michael Kroes.

1. Op het netwerk kan ik geen systeemdocumentatie over Spider vinden. Ik heb alleen een gebruikershandleiding gevonden. Is er behalve deze handleiding ook systeemdocumentatie beschikbaar? Bijvoorbeeld objectmodellen?

   Michael: Het systeem is niet echt goed gedocumenteerd. De structuur is wel makkelijk te achterhalen door een pagina weer te geven en achter de url &xml=true mee te geven. Dan wordt de hele xml-structuur van die pagina weergegeven. Dat is denk ik de beste manier om te achterhalen hoe content in Spider is opgebouwd. Verder raad ik je aan om de code goed te bekijken. De code is op zich goed gedocumenteerd.

2. Spider maakt voor het verwerken van data voor een groot deel gebruik van XML. Ook wordt er gebruik gemaakt van een Microsoft SQL Server database voor de opslag van data. Wat voor data wordt opgeslagen in XML-bestanden en wat voor data wordt opgeslagen in de database?

   Michael: Contentitems worden opgeslagen op disk in XML formaat, plaatjes XSL-T's en documenten komen ook gewoon op disk terecht. De verwijzing naar die content komt terecht in de SQL database.

3. Waarom is er überhaupt gekozen voor de opslag van gegevens in de SQL Server database terwijl de licentiekosten hiervoor toch redelijk duur zijn? Zijn er alternatieven? Zo ja, welke?

   Michael: Waarom? We hebben daar destijds voor gekozen… We wilden eerst iets lichtgewichters gebruiken. MSDE bijvoorbeeld. Maar we wisten ook nog niet of een een indexing service moesten gebruiken, daarvoor moet de content op disk staan. Achteraf had het misschien beter geweest, met name door de komst van SQL Server 2005, die echt met XML overweg kan, om de content in de database op te slaan. Is niet gebeurd, kan eventueel wel toegevoegd worden. Maar we bieden wel de mogelijkheid om de content gezipt en eventueel encrypted op disk op te slaan. Maar de overweging was toch vooral het verhaal van de indexing server.

3.5. Enquête gebruikers van Spider

De volgende vragenlijst heb ik gestuurd naar de volgende gebruikers van Spider.

- Linda Stelma, redacteur voor NDLite. (gebruikersgroep redacteurs)
- Wessel van Soest, directeur van het adviesbureau Argentum. (gebruikersgroep redacteurs/moderators/site-beheerders)
- Peter Brink, webmaster van het Nederlands Dagblad en NDLite.

Ik heb antwoorden teruggehad van Linda Stelma en Wessel van Soest. In de volgende tabel worden de vragen en de antwoorden weergegeven.

Nr	Vragen	Antwoorden Linda Stelma	Antwoorden Wessel van Soest
1	Wat is uw functie binnen het bedrijf waar u werkzaam bent?	Chef internetredactie, jongerenredacteur Clou en www.ndlite.nl	Directeur
2	Wat zijn uw dagelijkse werkzaamheden?	Onderwerpen zoeken, ANP-berichten bewerken en online zetten, artikelen schrijven en de internetredactie aansturen.	Alles. Daarbij kun je denken aan: inkoop, verkoop, administratie, personeelsmanagement, projectmanagement, relatiemanagement etc. En ook backoffice taken zoals websitebeheer. De overige 6 mensen van Argentum zijn adviseurs die fulltime op opdrachten werken.
3	Hoe lang maakt u al gebruik van Spider?	Sinds april 2004.	Sinds drie maanden.
4	Had u voordat u Spider gebruikte al ervaring/kennis in de ICT en hoe zou u deze willen typeren?	HTML-kennis. Ook al eerder gewerkt met een CMS in een productieomgeving.	Gebruiker van de MS-Office producten.
5	Waarom heeft u destijds gekozen voor Spider?	Spider was voor onze organisatie het goedkoopste alternatief waarbij we wel zelf invloed konden blijven uitoefenen.	Geen bewuste keuze. Door het gebruik van Small Business Server en het gehele beheer van taken bij Marbel was het logisch dat wij zelf onze website gingen beheren, daarvoor had Marbel Spider in de aanbieding.
6	Wat is uw algemene indruk van het product?	De basis werkt. Er is nog wel veel ontwikkeling nodig.	Het doet voor mij wat het moet doen, maar ik vind de opbouw en structuur niet helemaal logisch. Als je het niet dagelijks gebruikt dan is het wel een zoektocht. Daarnaast ontbreekt een duidelijke handleiding.
7	In welke regelmaat maakt u gebruik van Spider?	Dagelijks.	Eens in de twee weken.
8	Van welke functionaliteit maakt u het meest gebruik?	Het redactionele gedeelte.	Redactie en moderatie.
9	Wat zijn uw ervaringen bij het gebruik van die functionaliteit?	Redelijk. Sommige dingen werken omslachtig. Niet alles werkt. Soms loopt het systeem om onverklaarbare redenen vast. Ook mis ik functies.	Het aanpassen van een stuk tekst in niet moeilijk. Wel het aanpassen van lettertype e.d. Het toevoegen van een pagina is lastiger en daarvoor

Nr	Vragen	Antwoorden Linda Stelma	Antwoorden Wessel van Soest
			ontbreekt een logisch stappenplan.
10	Zijn er onderdelen binnen Spider die u niet volledig begrijpt?	Rechten beheer en Opmaak beheer. Daar hoef ik met niet in te verdiepen, daar hebben we een webmaster voor.	Toevoegen van een link, een plaatje, een doorklik naar een stuk Word-tekst, het toevoegen van meta-tags etc.
11	Wat zou er volgens u duidelijker kunnen?	Hoe je nieuwe categorieën kunt toevoegen.	Een simpel stappenplan: Wat moet ik doen als ik tekst toevoeg, als ik een plaatje wil, als ik een link wil, als ik...
12	Over welke functionaliteit / mogelijkheden binnen Spider bent u tevreden?	Over de HTML-mogelijkheden binnen de artikelen. Over de mogelijkheden een foto+onderschrift toe te voegen.	Redactie
13	Waar bent u minder tevreden over?	Over de snelheid, de statische volgorde van artikelen en over bepaalde dingen in de sms-service. Zie ook antwoord 9.	Het overige.
14	Heeft u ideeën voor eventuele nieuwe functionaliteit binnen Spider die het gebruik zouden kunnen vergemakkelijken?	Het zou mooi zijn als we bij een bericht een toekomstige plaatsingsdatum en -tijd in zouden kunnen voegen. Op die manier kun je berichten van te voren klaar zetten en ze later automatisch online laten zetten. Ook zou het mooi zijn als op de pagina 'Content toevoegen' het lettertype standaard ingesteld kan worden. Nu moeten we artikelen eerst naar notepad kopiëren en zodat de tekst verdana 10 zwart meekrijgt en daarna moeten we het weer in m-spider plakken. Die stap zou ertussenuit gehaald kunnen worden.	Online handleiding. Maar wel een handleiding die is geschreven voor de leek, dus in Jip-en-Janneke taal.
15	Heeft u verder nog opmerkingen of zaken die u kwijt wilt over Spider?	Nee, ik kan op dit moment niets meer bedenken.	Nee.
16	Zou ik naar aanleiding van uw antwoorden nogmaals (telefonisch) contact met u mogen opnemen?	Uiteraard! (In week 9 ben ik echter met vakantie)	Dat mag.

Uit de gegeven antwoorden wordt duidelijk dat zich problemen voordoen op verschillende vlakken. Het redactie-gedeelte wordt het beste begrepen. Het beheer-gedeelte is volstrekt onduidelijk bij Wessel van Soest en zal dus grondig moeten worden verbeterd.

4. Systeemeisen

In dit hoofdstuk worden de eisen vastgesteld waar de te ontwikkelen software aan moet voldoen. De eisen zijn opgesplitst in de volgende vijf categorieën.

- Basissysteemeisen
- Interface-eisen
- Integriteitseisen
- Performance-eisen
- Operationele eisen

Bij dit project gaat het om het verbeteren van het gebruikersgemak van Spider. De onderstaande eisen hebben dus geen betrekking op de technische structuur van Spider die al ontwikkeld is maar hebben betrekking op de wijzigingen die zullen worden aangebracht aan bepaalde onderdelen van Spider. De systeemeisen hebben betrekking op alle gebruikersgroepen op het gehele systeem. Als een eis betrekking heeft op de taken van een bepaalde gebruikersgroep, dan zal deze gebruikersgroep worden vermeld refererend naar paragraaf 3.3. Om ervoor te zorgen dat er gemakkelijk verwezen kan worden naar de onderstaande systeemeisen zijn ze genummerd per categorie. Bovendien worden in paragraaf 4.6 alle eisen geprioriteerd zodat direct duidelijk is wat de belangrijkste eisen zijn voor het systeem dat ontwikkeld gaat worden.

4.1. Basissysteemeisen

In deze paragraaf worden de basissysteemeisen gespecificeerd. Basissysteemeisen zijn functionele eisen waaraan het systeem moet voldoen.

BS.01 Gebruikers (gebruikersgroep site-beheerders) moeten wijzingen kunnen aanbrengen in de opmaak van een website zonder dat ze kennis hoeven te hebben van CSS.
BS.02 Gebruikers (gebruikersgroep site-beheerders) moeten nieuwe pagina's kunnen toevoegen aan een website zonder dat ze kennis hoeven te hebben van XML dat gebruikt wordt binnen Spider om pagina's te definiëren.
BS.03 Gebruikers (gebruikersgroep site-beheerders) moeten content-templates kunnen toevoegen aan een website zonder dat ze kennis te hoeven hebben van XML dat gebruikt wordt om nieuwe templates te definiëren binnen Spider.

4.2. Interface-eisen

In deze paragraaf wordt gespecificeerd hoe nieuwe gebruikersinterfaces eruit komen te zien en aan welke eisen deze moeten voldoen.

IF.01 De gebruikersinterfaces moeten goed worden weergegeven in Microsoft Internet Explorer 5.5 of hoger.
IF.02 De gebruikersinterfaces moeten aansluiten bij de huidige vormgeving van Spider.
IF.03 De gebruikersinterfaces moeten te gebruiken zijn zonder dat er aanvullende training voor nodig is.
IF.04 De gebruikersinterfaces moeten worden uitgerust met ingebouwde helpfunctionaliteit.
IF.05 Ervaren gebruikers moeten de mogelijkheid hebben om de gebruikersinterfaces uit te kunnen schakelen.
IF.06 De gebruikersinterfaces moeten een correcte invoer van de gebruikers afdwingen.

4.3. Integriteitseisen

De volgende kenmerken moeten een bijdrage leveren aan de betrouwbaarheid en de accuratesse van het systeem.

IG.01 De gegevens die worden ingebracht in Spider dienen centraal te worden opgeslagen zodat ze op elk moment geraadpleegd kunnen worden en gemakkelijk kunnen worden gebackupt.
IG.02 Er moeten meerdere versies van de ingebrachte gegevens bewaard worden, zodat in geval van een vergissing door de gebruiker, een vorige versie van de gegevens kunnen worden herroepen.

4.4. Performance-eisen

Doordat er sprake is van een webapplicatie is het moeilijk om concrete getallen te geven als het gaat om laadtijd en/of responsetijd. Deze worden namelijk beïnvloed door meerdere factoren zoals

- de verbindingssnelheid van de gebruiker
- de verbindingssnelheid van webserver
- de hoeveelheid tegelijk ingelogde gebruikers op het systeem
- de hoeveelheid gebruikers die gebruik maken van de internetverbinding
- het tijdstip van de dag waarop er gebruik gemaakt wordt van het systeem (hoeveelheid dataverkeer op internet).

Onderstaande eisen worden gesteld aan de te leveren prestaties van het systeem.

PF.01 De responsetijd van het systeem op acties van de gebruiker dient te worden beperkt tot maximaal 1 seconde als het gaat om het bewerken van kleine hoeveelheden informatie. (tot 5000 karakters) bij een verbindingssnelheid van 100 kbps.
PF.02 De te ontwikkelen gebruikersinterfaces dienen geladen te zijn in maximaal 1 seconde bij een verbindingssnelheid van 100 kbps.

4.5. Operationele eisen

De volgende kenmerken houden verband met de veranderingen in de organisatie die de invoering van het systeem teweeg zal brengen.

OP.01 Spider wordt ontwikkeld en beheerd door Marbel Software. De nieuwe functionaliteit die in het kader van dit project wordt ontwikkeld, zal worden meegenomen bij de invoering van een nieuwe release van Spider bij de gebruikers.
OP.02 De nieuwe functionaliteit die ontwikkeld gaat worden, moet beheerd kunnen worden door Marbel Software.
OP.03 Als een nieuwe versie van Spider met de in dit project ontwikkelde functionaliteit bij een klant ingevoerd gaat worden, moeten de gebruikers goed met de veranderingen overweg kunnen. Daarom zal er een uitgebreide beschrijving worden gemaakt waarin de veranderingen worden toegelicht.

4.6. Geconsolideerde systeemeisen

De volgende lijst komt voort uit de voorgaande paragrafen en is geprioriteerd op basis van de mate van belang met de belangrijkste eis bovenaan.

BS.02 Gebruikers (gebruikersgroep site-beheerders) moeten nieuwe pagina's kunnen toevoegen aan een website zonder dat ze kennis hoeven te hebben van XML dat gebruikt wordt binnen Spider om pagina's te definiëren.
BS.01 Gebruikers (gebruikersgroep site-beheerders) moeten wijzingen kunnen aanbrengen in de opmaak van een website zonder dat ze kennis hoeven te hebben van CSS.
BS.03 Gebruikers (gebruikersgroep site-beheerders) moeten nieuwe content-templates kunnen toevoegen aan een website zonder dat ze kennis te hoeven hebben van XML dat gebruikt wordt om nieuwe templates te definiëren binnen Spider.
IF.06 De gebruikersinterfaces moeten een correcte invoer van de gebruikers afdwingen.
IF.05 Ervaren gebruikers moeten de mogelijkheid hebben om de gebruikersinterfaces uit te kunnen schakelen.
IF.04 De gebruikersinterfaces moeten worden uitgerust met ingebouwde helpfunctionaliteit.
IF.03 De gebruikersinterfaces moeten te gebruiken zijn zonder dat er aanvullende training voor nodig is.
IF.02 De gebruikersinterfaces moeten aansluiten bij de huidige vormgeving van Spider.
IF.01 De gebruikersinterfaces moeten goed worden weergegeven in Microsoft Internet Explorer 5.5 of hoger.
IG.03 Er moeten meerdere versies van de ingebrachte gegevens bewaard worden, zodat in geval van een vergissing door de gebruiker, een vorige versie van de gegevens kunnen worden herroepen.
IG.01 De gegevens die worden ingebracht in Spider dienen centraal te worden opgeslagen zodat ze op elk moment geraadpleegd kunnen worden en gemakkelijk kunnen worden gebackupt.
PF.01 De responsetijd van het systeem op acties van de gebruiker dient te worden beperkt tot maximaal 1 seconde als het gaat om het bewerken van kleine hoeveelheden informatie. (tot 5000 karakters) bij een verbindingssnelheid van 100 kbps.
PF.02 De te ontwikkelen gebruikersinterfaces dienen geladen te zijn in maximaal 1 seconde bij een verbindingssnelheid van 100 kbps.
OP.03 Als een nieuwe versie van Spider met de in dit project ontwikkelde functionaliteit bij een klant ingevoerd gaat worden, moeten de gebruikers goed met de veranderingen overweg kunnen. Daarom zal er een uitgebreide beschrijving worden gemaakt waarin de veranderingen worden toegelicht.
OP.01 Spider wordt ontwikkeld en beheerd door Marbel Software. De nieuwe functionaliteit die in het kader van dit project wordt ontwikkeld, zal worden meegenomen bij de invoering van een nieuwe release van Spider bij de gebruikers.
OP.02 De nieuwe functionaliteit die ontwikkeld gaat worden, moet beheerd kunnen worden door Marbel Software.

5. Systeemconcept

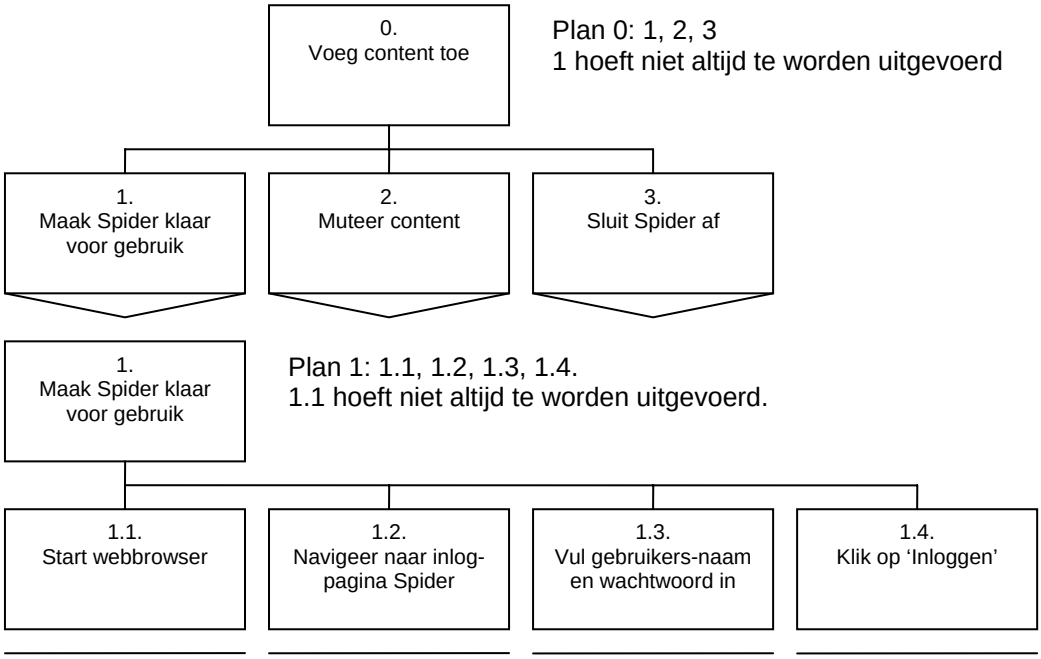
In dit hoofdstuk wordt op basis van de systeemeisen zoals die zijn opgesteld in hoofdstuk 4 een beschrijving op globaal niveau gegeven van de oplossing die ontwikkeld zal gaan worden.

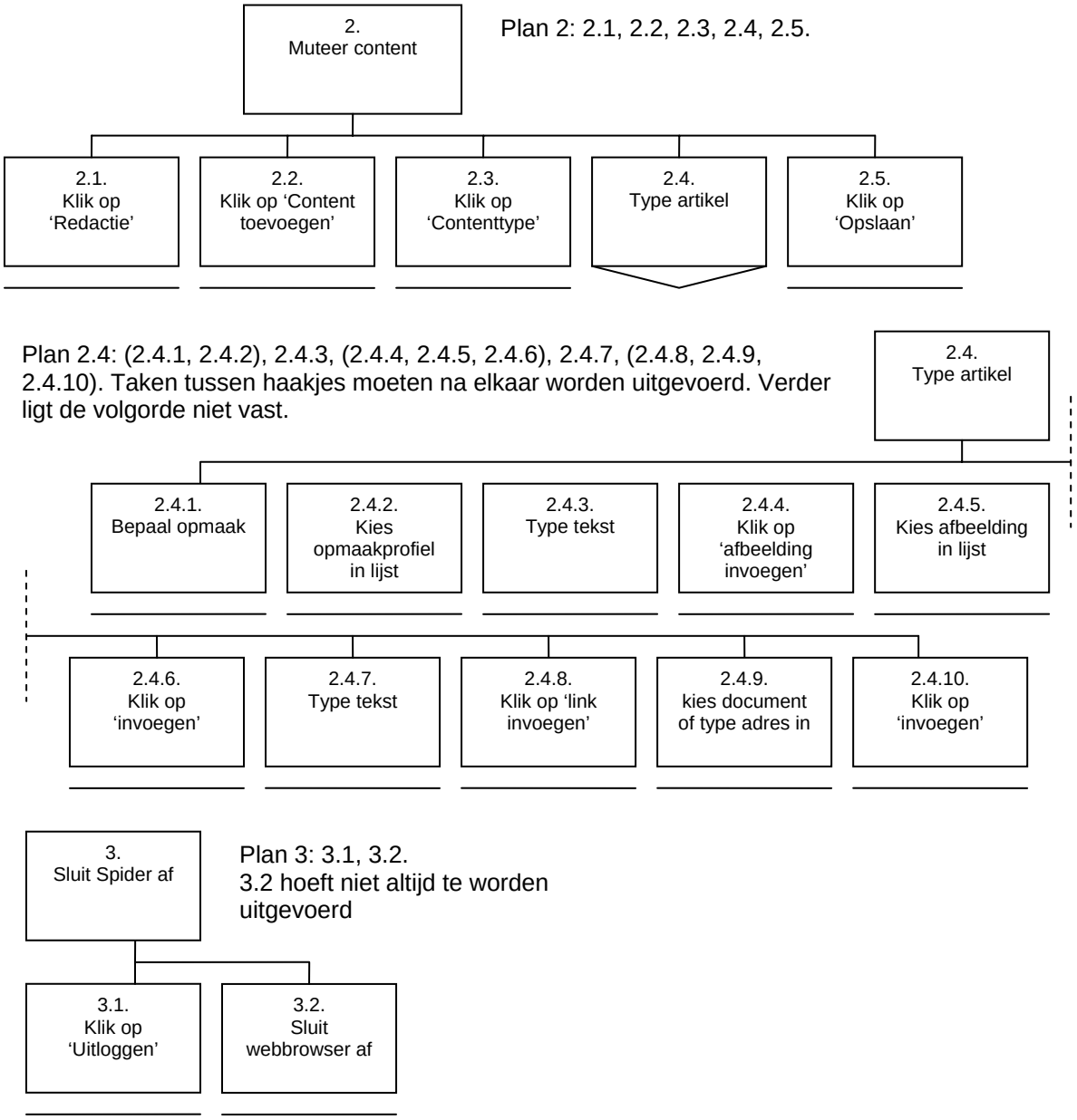
5.1. Taakdiagrammen gewenste situatie

Uit de vragenlijst die door de gebruikers is ingevuld, bleek dat op verschillende plaatsen in het systeem bepaalde functionaliteit erg onduidelijk is. Voor de redacteurs blijkt het bijvoorbeeld lastig te zijn om het lettertype van een artikel aan te passen. Verder begrijpen site-beheerders weinig van de mogelijkheden die Spider biedt voor het dynamisch beheren van een website. Om site-beheerders te ondersteunen bij het beheren van een website moet er een taakgerichte gebruikersinterface komen die de gebruiker begeleidt in de verschillende beheerstaken. Het grote tekstvlak dat door de gebruiker gevuld moet worden met codes moet vervangen worden door een scherm waarbij de gebruiker door middel van het drukken op knoppen dezelfde taken uitvoert. De taken van de gebruiker zullen dus vooral verder gespecificeerd worden. Een taak als ‘muteer wijzigingen in tekstvlak’ wordt dan bijvoorbeeld opgesplitst in ‘Klik op nieuwe sectie’, ‘Vul naam van de sectie in’, ‘Klik op elementen toevoegen aan sectie’, ‘Vul naam element in’, ‘Vink element-opties aan’, ‘Kies bijbehorende categorieën in de lijst’, etc.

5.1.1. Gebruikersgroep redacteurs

Bij deze gebruikersgroep zal de taak ‘content toevoegen’ worden verbeterd. Hierbij gaat het om taak 2.4. (Type artikel).





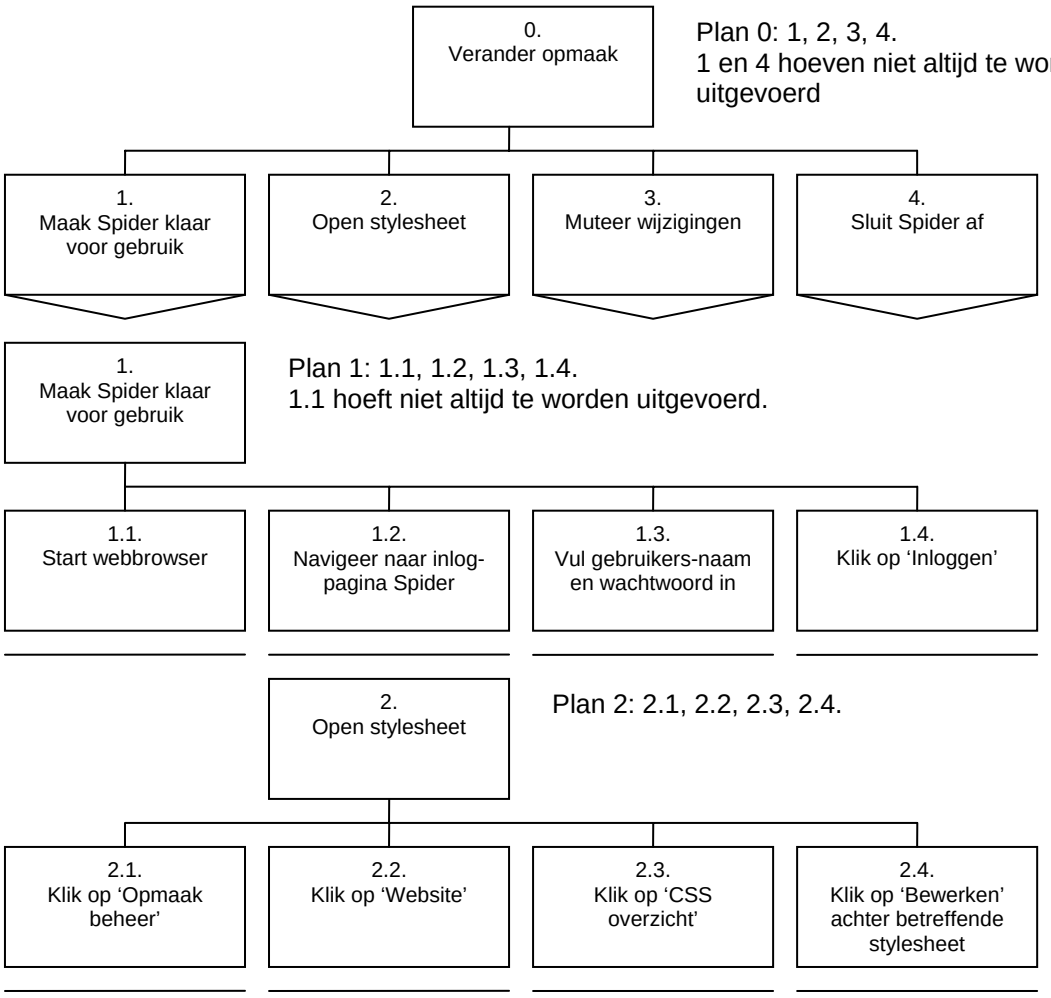
Figuur 12: Nieuw taakdiagram content toevoegen

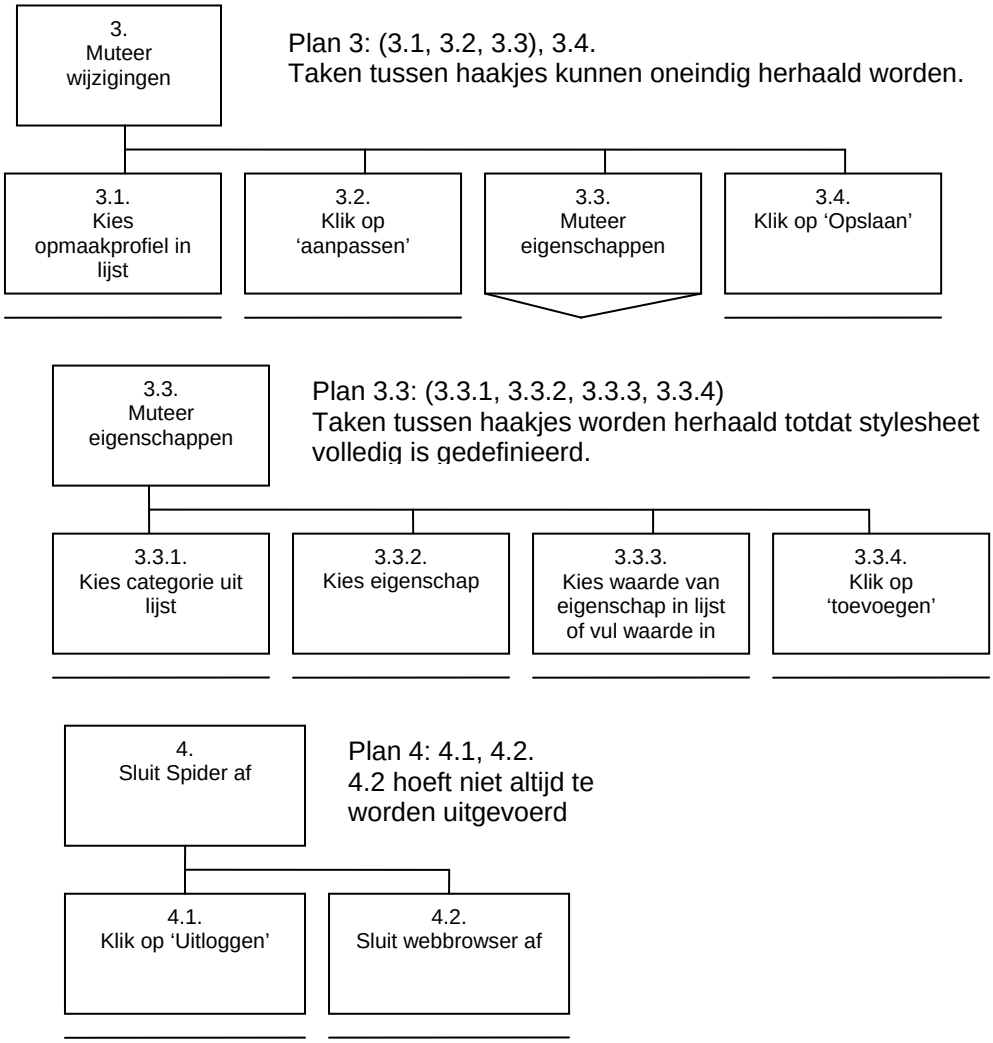
5.1.2. Gebruikersgroep moderators

De taken voor de gebruikersgroep moderators worden al volledig gestuurd door een grafische interface. Hierin zullen geen wijzigingen worden aangebracht aangezien daar geen behoefte aan is bij de gebruikers.

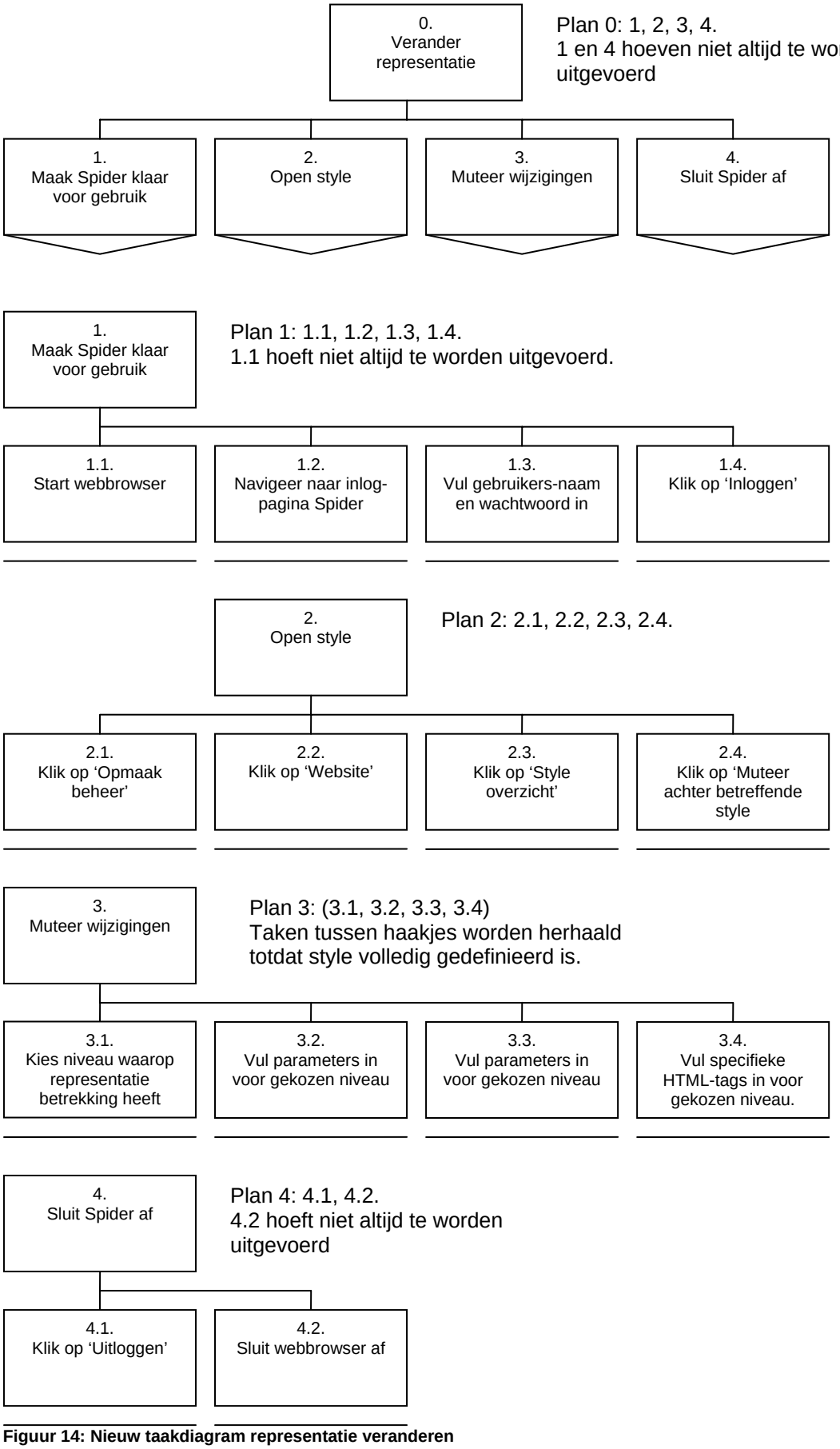
5.1.3. Gebruikersgroep site-beheerders

Het beheergedeelte zal de meest ingrijpende wijzigingen ondergaan. Het tekstvlak waar de gebruiker codes in dient te typen wordt vervangen door een grafische interface waarbij de taken worden uitgevoerd door middel van het klikken op knoppen en het kiezen van opties uit lijsten.
In onderstaande taakdiagrammen worden de nieuwe taken gespecificeerd. Op het moment van het schrijven van deze definitiestudie is nog niet volledig bekend hoe de schermen eruit komen te zien dus is het nog niet mogelijk de taken te definiëren op detailniveau.

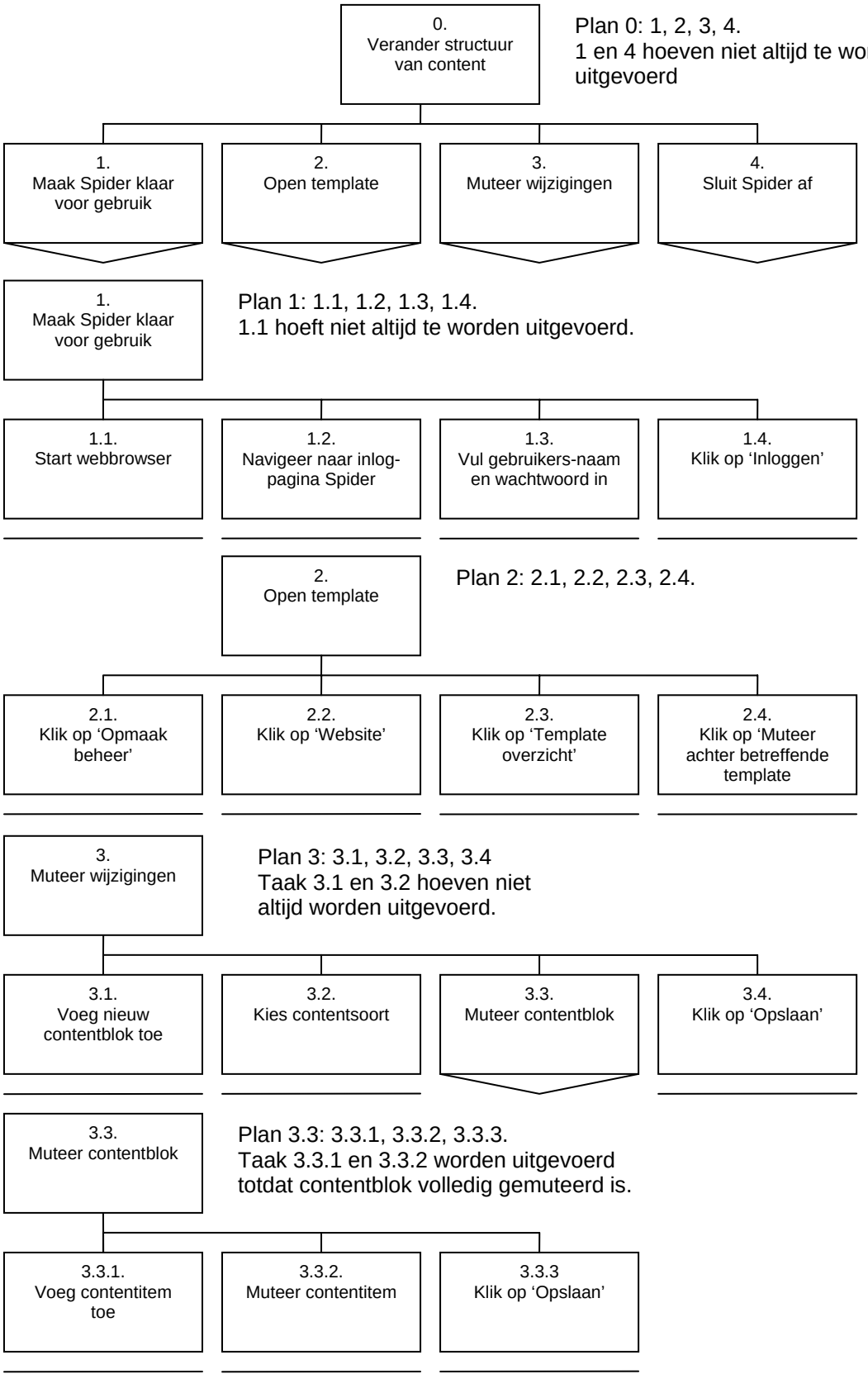


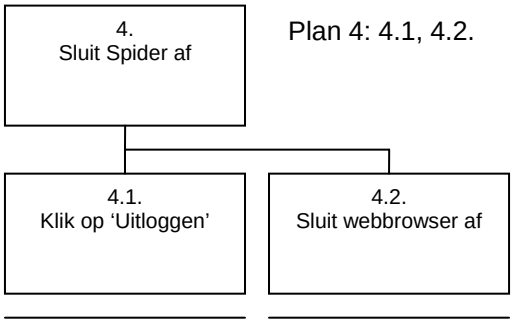


Figuur 13: Nieuw taakdiagram opmaak veranderen

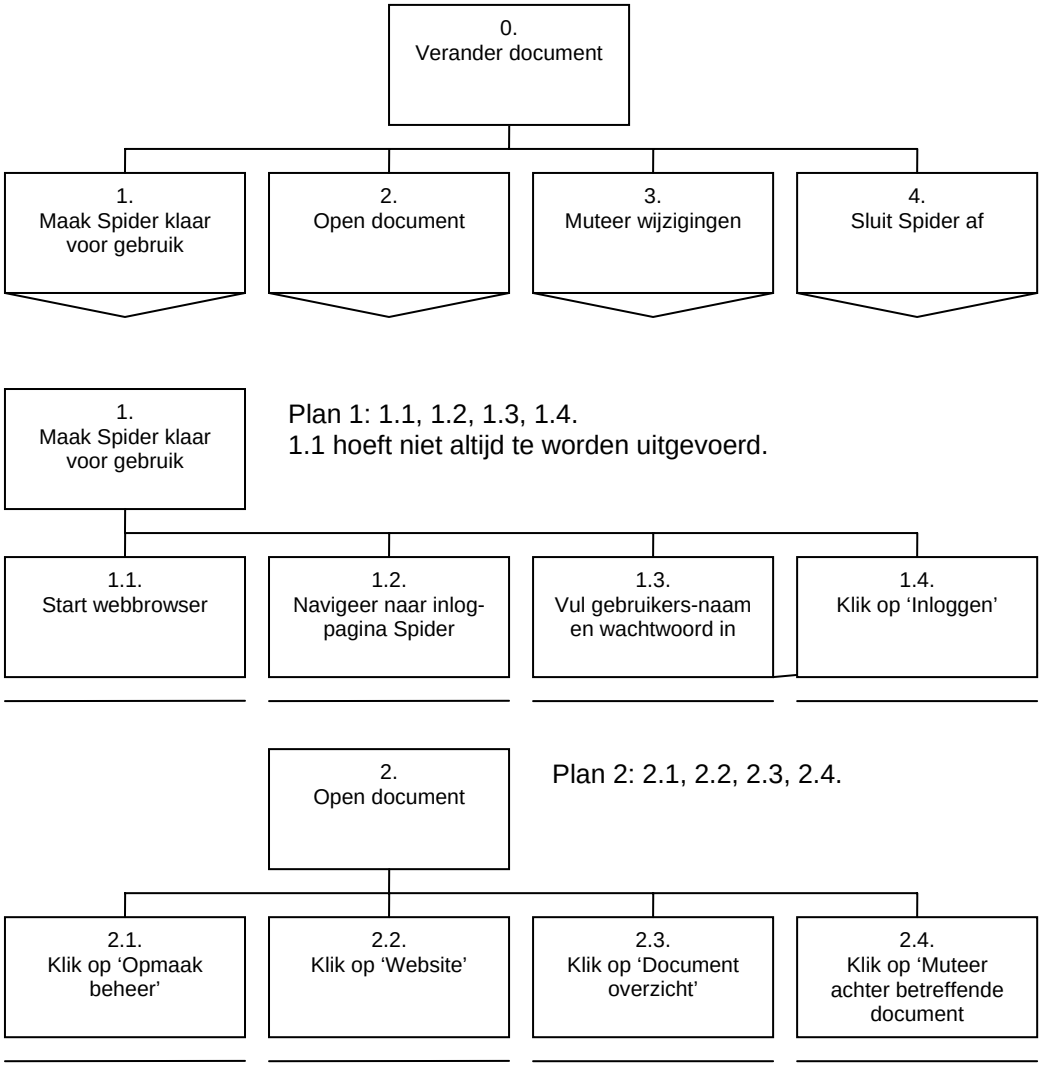


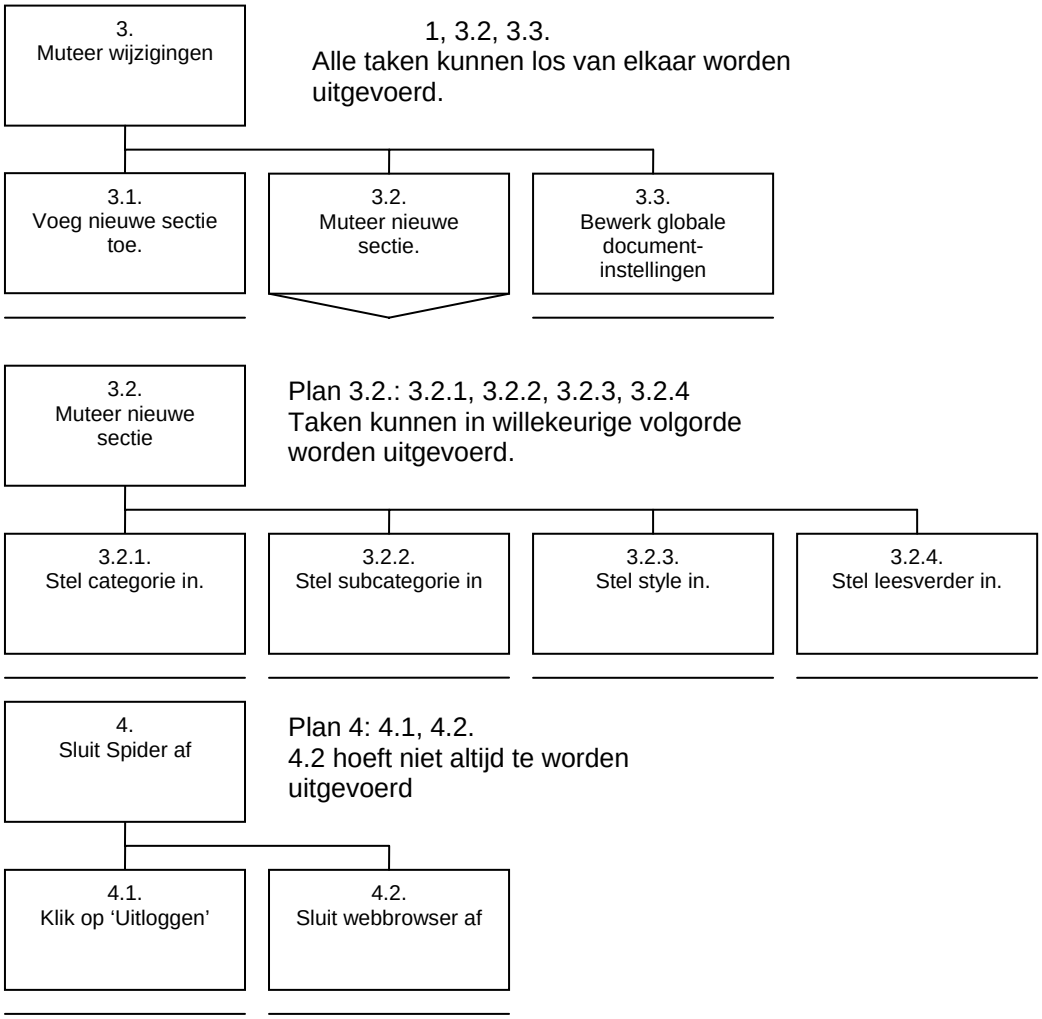
Figuur 14: Nieuw taakdiagram representatie veranderen


0.
Verander structuur van content
Plan 0: 1, 2, 3, 4.
1 en 4 hoeven niet altijd te worden uitgevoerd
1.
Maak Spider klaar voor gebruik
2.
Open template
3.
Muteer wijzigingen
4.
Sluit Spider af
1.
Maak Spider klaar voor gebruik
Plan 1: 1.1, 1.2, 1.3, 1.4.
1.1 hoeft niet altijd te worden uitgevoerd.
1.1.
Start webbrowser
1.2.
Navigeer naar inlog-pagina Spider
1.3.
Vul gebruikers-naam en wachtwoord in
1.4.
Klik op 'Inloggen'
2.
Open template
Plan 2: 2.1, 2.2, 2.3, 2.4.
2.1.
Klik op 'Opmaak beheer'
2.2.
Klik op 'Website'
2.3.
Klik op 'Template overzicht'
2.4.
Klik op 'Muteer achter betreffende template
3.
Muteer wijzigingen
Plan 3: 3.1, 3.2, 3.3, 3.4
Taak 3.1 en 3.2 hoeven niet altijd worden uitgevoerd.
3.1.
Voeg nieuw contentblok toe
3.2.
Kies contentsoort
3.3.
Muteer contentblok
3.4.
Klik op 'Opslaan'
3.3.
Muteer contentblok
Plan 3.3: 3.3.1, 3.3.2, 3.3.3.
Taak 3.3.1 en 3.3.2 worden uitgevoerd totdat contentblok volledig gemuteerd is.
3.3.1.
Voeg contentitem toe
3.3.2.
Muteer contentitem
3.3.3
Klik op 'Opslaan'



Figuur 15: Nieuw taakdiagram structuur veranderen

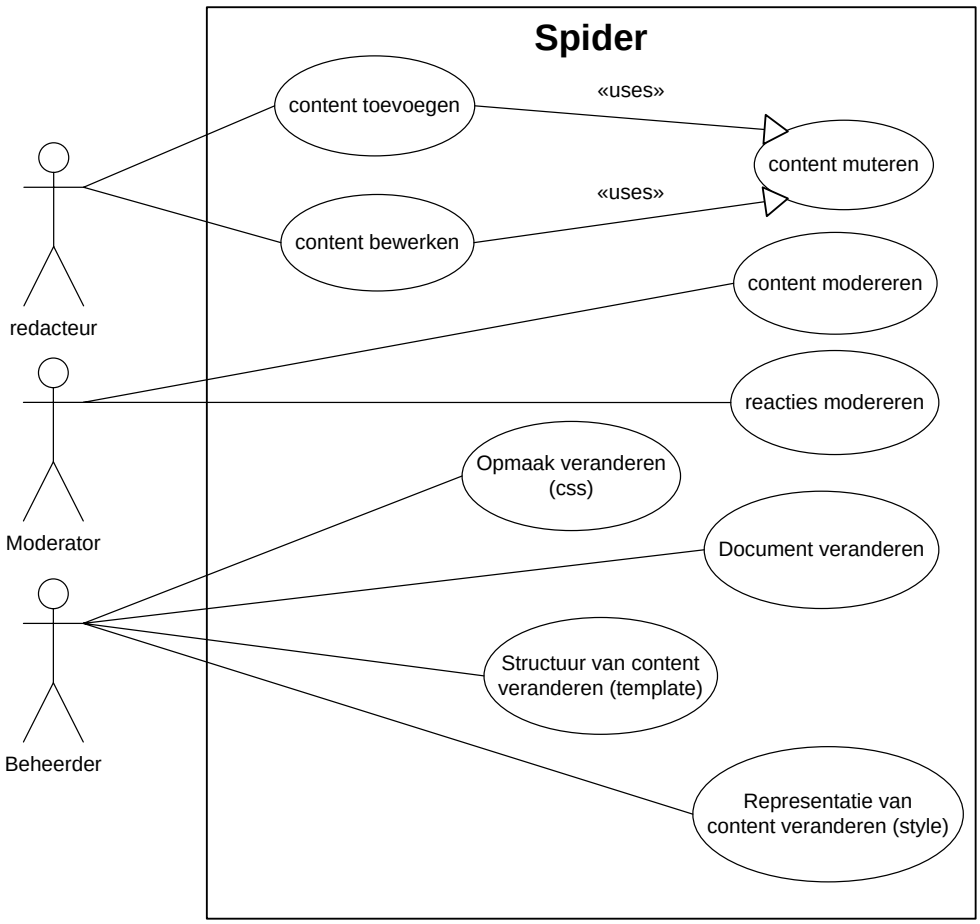




Figuur 16: Nieuw taakdiagram document veranderen

5.2. Use case

Het onderstaande use case diagram geeft inzicht in de werking van het systeem. De actoren aan de linkerkant komen overeen met de verschillende gebruikersgroepen. Het rechthoekige vlak stelt de grenzen van het informatiesysteem voor. De ovalen zijn de daadwerkelijke use cases die overeenkomen met de taken uit hoofdstuk 5.1. Het diagram beschrijft zowel de huidige als de gewenste situatie van het systeem doordat de taken globaal gezien niet veranderen.



Figuur 17: Use case diagram Spider

Use case beschrijving content toevoegen	
Naam	Content toevoegen
Samenvatting	Een redacteur voegt content toe volgens een bepaalde template in een bepaalde categorie.
Actoren	Redacteur
Aannamen of precondities	• De gebruiker is ingelogd. • De gebruiker heeft rechten om content te muteren in de categorie.
Beschrijving scenario's	1. De gebruiker klikt in het menu op redactie. 2. De gebruiker klikt in het menu op content toevoegen. 3. De gebruiker klikt in het menu op het gewenste contenttype. 4. De gebruiker schrijft het artikel. 5. De gebruiker selecteert de categorie waar content in geplaatst moet worden. 6. De gebruiker Klikt op 'Opslaan'.
Uitzonderingen	
Resultaat of postcondities	Er is nieuwe content geplaatst.

Figuur 18: Use case beschrijving content toevoegen

Use case beschrijving content bewerken	
Naam	Content bewerken
Samenvatting	Een redacteur opent bestaande content, voert wijzigingen door en slaat de wijzigingen op.
Actoren	Redacteur
Aannamen of precondities	• De gebruiker is ingelogd. • De gebruiker heeft rechten om content te muteren in de categorie.
Beschrijving scenario's	1. De gebruiker klikt in het menu op redactie. 2. De gebruiker klikt in het menu op content-overzicht. 3. De gebruiker klikt in het menu op de gewenste categorie. 4. De gebruiker klikt in het menu op de gewenste subcategorie. 5. De gebruiker klik in de lijst op de content die hij wil bewerken. 6. De gebruiker brengt wijzigingen aan in de content. 7. De gebruiker Klikt op 'Opslaan'.
Uitzonderingen	
Resultaat of postcondities	Er is nieuwe content geplaatst.

Figuur 19: Use case beschrijving content bewerken

Use case beschrijving content modereren	
Naam	Content modereren
Samenvatting	Een moderator opent bestaande content, past zonodig de tekst aan en keurt de content goed of af.
Actoren	Moderator
Aannamen of precondities	• De gebruiker is ingelogd. • De gebruiker heeft rechten om content te modereren in de categorie. • Modereren is ingeschakeld voor de te modereren content.
Beschrijving scenario's	1. De gebruiker klikt in het menu op moderatie. 2. De gebruiker klikt in het menu op content-moderatie overzicht 3. De gebruiker klikt in het menu op de gewenste categorie. 4. De gebruiker klikt in het menu op de gewenste subcategorie. 5. De gebruiker klikt in de lijst op de content die hij wil modereren. 6. De gebruiker past de content zonodig aan en keurt goed/af..
Uitzonderingen	
Resultaat of postcondities	Content is gemodereerd.

Figuur 20: Use case beschrijving content modereren

Use case beschrijving ea	
Naam	Reactie modereren
Samenvatting	Een moderator opent een reactie van een bezoeker, verandert zonodig de tekst en keurt de reactie goed of af.
Actoren	Moderator
Aannamen of precondities	• De gebruiker is ingelogd. • De gebruiker heeft rechten om reacties te modereren in de categorie.
Beschrijving scenario's	1. De gebruiker klikt in het menu op moderatie. 2. De gebruiker klikt in het menu op reactie-moderatie overzicht. 3. De gebruiker klikt in de lijst op de reactie die hij wil modereren. 4. De gebruiker past de reactie zonodig aan en keurt goed/af.
Uitzonderingen	
Resultaat of postcondities	Reactie is gemodereerd.

Figuur 21: Use case beschrijving reactie modereren

Use case beschrijving document veranderen	
Naam	Document veranderen
Samenvatting	Een beheerder maakt een nieuw document aan of wijzigt een bestaand document en wijzigt de secties die worden weergegeven in dat document.
Actoren	Beheerder
Aannamen of precondities	• De gebruiker is ingelogd. • De gebruiker heeft beheerderrechten
Beschrijving scenario's	1. De gebruiker klikt in het menu op opmaak

	beheer. 2. De gebruiker klikt in het menu op website 3. De gebruiker klikt in het menu op document overzicht. 4. De gebruiker klikt in de lijst met documenten op 'Muteer' achter het te muteren document of voegt een nieuw document toe met de knop 'nieuw document'. 5. De gebruiker vult bovenaan de pagina de globale eigenschappen van het document in. 6. De gebruiker voegt een nieuwe sectie toe. 7. De gebruiker klikt in de lijst met gedefinieerde secties op wijzigen. 8. De gebruiker past de sectie-eigenschappen aan en klikt op 'opslaan'. Indien er syntaxfouten bevinden in de XML-code, treedt de uitzondering XMLSyntaxFout op.
Uitzonderingen	[XMLSyntaxFout]: Geeft een melding dat het document niet kan worden opgeslagen omdat er een fout zit in de XML-syntax.
Resultaat of postcondities	Document is aangepast.

Figuur 22: Use case beschrijving document veranderen

Use case beschrijving represen n (style)	
Naam	Representatie veranderen
Samenvatting	Een beheerder maakt een nieuwe style of past een bestaande style aan.
Actoren	Beheerder
Aannamen of precondities	• De gebruiker is ingelogd. • De gebruiker heeft rechten om styles te muteren.
Beschrijving scenario's	1. De gebruiker klikt in het menu op opmaak beheer. 2. De gebruiker klikt in het menu op style overzicht. 3. De gebruiker klikt in de lijst met styles op een style die hij wil muteren. 4. De gebruiker kiest in de style het niveaujwaarop de representatie betrekking heeft. 5. De gebruiker vult parameters in voor het gekozen niveau. 6. De gebruiker vult specifieke HTML-tags in voor het gekozen niveau. 7. De gebruiker klikt op 'opslaan'. Indien er syntaxfouten bevinden in XML treedt de uitzondering XMLSyntaxFout op.
Uitzonderingen	[XMLSyntaxFout]: Geeft een melding dat de style niet kan worden opgeslagen omdat er een fout zit in de XML-syntax.
Resultaat of postcondities	Representatie is veranderd

Figuur 23: Use case beschrijving representatie veranderen

Use case beschrijving structuur van content veranderen (template)	
Naam	Structuur van content veranderen
Samenvatting	Een beheerder maakt een nieuwe template of past een bestaande template aan.

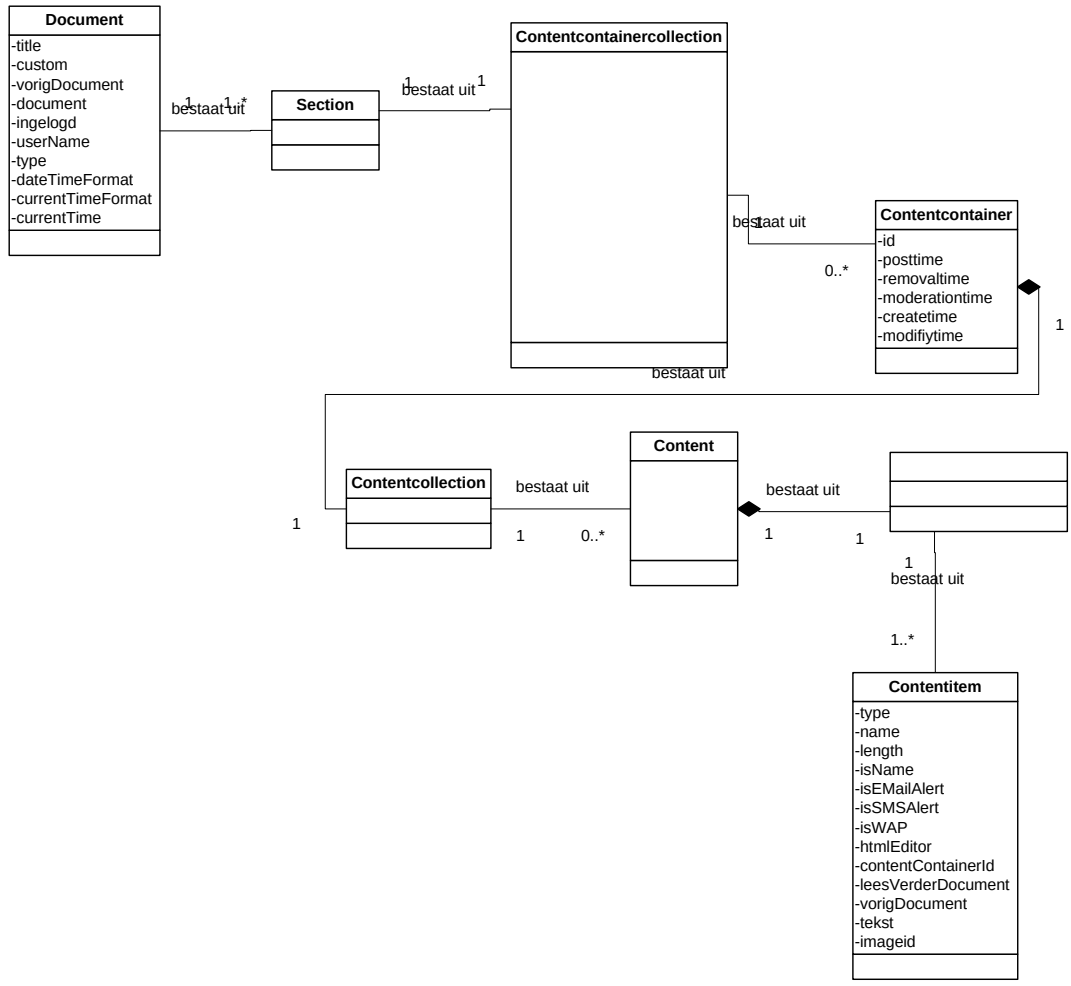
Actoren	Beheerder
Aannamen of precondities	• De gebruiker is ingelogd. • De gebruiker heeft rechten om templates te muteren.
Beschrijving scenario's	1. De gebruiker klikt in het menu op opmaak beheer. 2. De gebruiker klikt in het menu op template overzicht. 3. De gebruiker klikt in de lijst met templates op de template die hij wil muteren. 4. De gebruiker voegt een nieuw contentblok toe. 5. De gebruiker stelt de eigenschappen in voor het toegevoegde contentblok. 6. De gebruiker voegt nieuwe contentitems aan het contentblok toe. 7. De gebruiker klikt op 'opslaan'. Indien er syntaxfouten bevinden in XML treedt de uitzondering XMLSyntaxFout op.
Uitzonderingen	[XMLSyntaxFout]: Geeft een melding dat de template niet kan worden opgeslagen omdat er een fout zit in de XML-syntax.
Resultaat of postcondities	Template is toegevoegd of aangepast.

Figuur 24: Use case beschrijving structuur van content veranderen

Use case beschrijving op	
Naam	Opmaak veranderen
Samenvatting	Een beheerder opent een stylesheet en brengt wijzigingen aan.
Actoren	Beheerder
Aannamen of precondities	• De gebruiker is ingelogd. • De gebruiker heeft rechten om stylesheets te muteren.
Beschrijving scenario's	1. De gebruiker klikt in het menu op opmaak beheer. 2. De gebruiker klikt in het menu op css overzicht. 3. De gebruiker klikt in de lijst op de stylesheet die hij wil wijzigen. 4. De gebruiker klikt op het opmaakelement dat hij wilt wijzigen. 5. De gebruiker verandert de waarde van één van de eigenschappen of voegt een nieuwe eigenschap toe en stelt daar de waarde van in.
Uitzonderingen	
Resultaat of postcondities	Stylesheet is toegevoegd of aangepast.

5.3. Klassendiagram

Spider is voor groot deel gebaseerd op XML. Alle content wordt opgeslagen in XML bestanden en ook de content-templates, documentdefinities en de styles worden opgeslagen in XML bestanden. Als een bezoeker van de in Spider beheerde website een pagina opvraagt, wordt de XML getransformeerd naar XHTML zodat het kan worden weergegeven in een webbrowser. De verschillende XML bestanden hebben ieder een eigen structuur en hebben niet altijd relaties met elkaar. Daardoor is het niet mogelijk om de structuur van alle klassen in één diagram af te beelden. De verschillende contentklassen hebben wel relaties met elkaar. Daarom wordt hieronder voor de alle contentklassen het klassendiagram weergegeven.



Figuur 25: Klassendiagram content-klassen Spider

De klassen die afgebeeld zijn in het bovenstaande diagram worden toegelicht in de onderstaande tabellen.

Beschrijving klasse document		
Beschrijving		Een document is binnen Spider een definitie van een pagina die kan worden opgevraagd door de webbrowser. Een document kan momenteel worden gedefinieerd in Spider door middel van het intikken van XML tags. Deze definitie wordt opgeslagen op disk en wordt door de DocumentService gebruikt om webpagina's weer te geven.
Atributen	vorigDocument	Dit attribuut wordt door DocumentService ingesteld en bevat de naam van het vorige document dat is opgevraagd door de DocumentService.
	Document	Dit attribuut bepaalt welk document er wordt weergegeven. Het moet een geldige naam van een documentdefinitie bevatten.
	Ingelogd	Bepaalt of er is ingelogd. Indien de waarde true is, bevat het attribuut userName de gebruikersnaam.
	userName	Indien er is ingelogd als gebruiker bevat dit attribuut de gebruikersnaam.
	Type	Geeft aan om wat voor type document het gaat. 'web' voor webpagina.
	dateTimeFormat	Specificeert in welk formaat data en tijd worden moet worden weergegeven.
	currentTimeFormat	Specificeert in welk formaat de huidige tijd moet worden weergegeven.
	currentTime	Bevat huidige tijd volgens opgegeven formaat.
Structuur		<document> <head> <title> </title> <custom> </custom> </head> <sections> <section> <content/> </section> </sections> </document>

Figuur 26: Tekstuele beschrijving klasse document

Beschrijving klasse contentcontainercollection		
Beschrijving		Een contentcontainercollection wordt in een documentdefinitie gedefinieerd binnen een section-tag met een content-tag. Deze definitie bepaalt welke en hoeveel content er moet worden weergegeven in de sectie, uit welke categorieën en op welke wijze die moet worden gepresenteerd.
Atributen	Count	Geeft aan hoeveel content er moet worden weergegeven.
	Style	Geeft aan volgens welke style (XSL-T) de content moet worden weergegeven.
	Category	Geeft aan uit welke categorieën de content moet worden geselecteerd (scheidingstekens: `, of / of - of + of \`)
	SubCategory	Geeft aan uit welke subcategorieën de content moet worden geselecteerd (scheidingstekens: `, of / of - of + of \`)
	LeesVerderDocument	Geeft aan welk document er gebruikt moet worden voor het weergeven van het leesverder document
	LeesVerder	Bepaalt of er sprake is van een leesverderdocument.

	isSearch	Geeft aan dat de zoekresultaten moeten worden weergegeven.
	isSearchQuery	Geeft aan dat er een zoek-tekstbox moet worden weergegeven.
	isLogin	Geeft aan dat er tekstboxen moeten worden weergegeven om in te kunnen loggen.
	isRegister	Geeft aan dat er tekstboxen moeten worden weergegeven om te kunnen registreren.
	isProfile	Geeft aan dat er tekstboxen moeten worden weergegeven om een gebruikersprofiel te kunnen aanpassen.
	DateTimeFormat	Specificeert in welk formaat data en tijd worden moet worden weergegeven.
	SortOrder	Geeft aan hoe de content moet worden gesorteerd (asc of desc)
	SortOnTime	Geeft aan op welke tijd de content moet worden gesorteerd (posttime of moderatetime of creationtime of removaltime of modifytime)
Structuur		<contentcontainercollection> <contentcontainer> <contentcollection> <content> <contentitemcollection> <contentitem/> </contentitemcollection> </content> </contentcollection> </contentcontainer> </contentcontainercollection>

Figuur 27: Tekstuele beschrijving klasse contentcontainercollection

Bes klasse contentcontainer		
Beschrijving		Een contentcontainer is onderdeel van de contentcontainercollection en bevat één contentcollection. Elk artikel dat wordt toegevoegd in het redactiegedeelte van Spider genereert één contentcontainer.
Atributen	Id	Identificatienummer dat automatisch wordt gegenereerd als nieuwe content wordt toegevoegd.
	Posttime	Tijdstip waarop contentcontainer is geplaatst.
	Removaltime	Tijdstip waarop contentcontainer is verwijderd. (nergens geplaatst)
	Moderationtime	Tijdstip waarop contentcontainer is goedgekeurd.
	Createtime	Tijdstip waarop contentcontainer is gemaakt.
	Modifytime	Tijdstip waarop contentcontainer is aangepast.
Structuur		<contentcontainer> <contentcollection> <content> <contentitemcollection> <contentitem/> </contentitemcollection> </content> </contentcollection> </contentcontainer>

Figuur 28: Tekstuele beschrijving klasse contentcontainer

B		
Beschrijving		De contentklasse staat voor de onderdelen waaruit een contentcontainer is opgebouwd
Atributen	Type	Type kan 3 verschillende waarden hebben: content of meta of reactie. content staat voor de daadwerkelijke informatie.

		metacontent genereert de metatags voor zoekmachines. reactie betekent dat het een reactie van een gebruiker betreft.
	Redactie	Geeft aan dat de content afkomstig is van een redacteur.
	isReactie	Geeft aan of het een reactie betreft.
	Name	Website geeft aan dat het om webcontent gaat Mobile geeft aan dat het om wapcontent gaat.
	isHidden	Geeft aan of de content verborgen moet worden.
Structuur		<content> <contentitemcollection> <contentitem/> </contentitemcollection> </content>

Figuur 29: Tekstuele beschrijving klasse content

Bes		
Beschrijving		Het contentitem is de kleinste klasse. Een contentitem kan een afbeelding zijn. Meerdere contentitems vormen samen content.
Atributen	Type	Image of tekst, bij tekst moet de tekst in een CDATA sectie worden ondergebracht. In het geval van een image moet er een childelement imageid komen die het nummer van de afbeelding bevat.
	Name	De naam van het contentitem.
	Length	Het aantal karakters waaruit het contentitem mag bestaan in het geval van tekst.
	isEmailAlert	Geeft aan of het om een alert gaat die via email naar alle gebruikers (leden) van de site moet worden gestuurd.
	isSMSAlert	Geeft aan of het om een alert gaat die via SMS naar alle gebruikers (leden) van de site moet worden gestuurd.
	isWAP	Geeft aan of het om WAP content gaat.
	htmlEditor	Geeft aan of er een htmlEditor moet worden weergegeven bij het mutereren van de content.
	contentContainerId	Verwijzing naar het Id van de contentcontainer.
Structuur		<contentitem> <![CDATA[ ]]> </contentitem> Of <contentitem> <imageid></imageid> </contentitem>

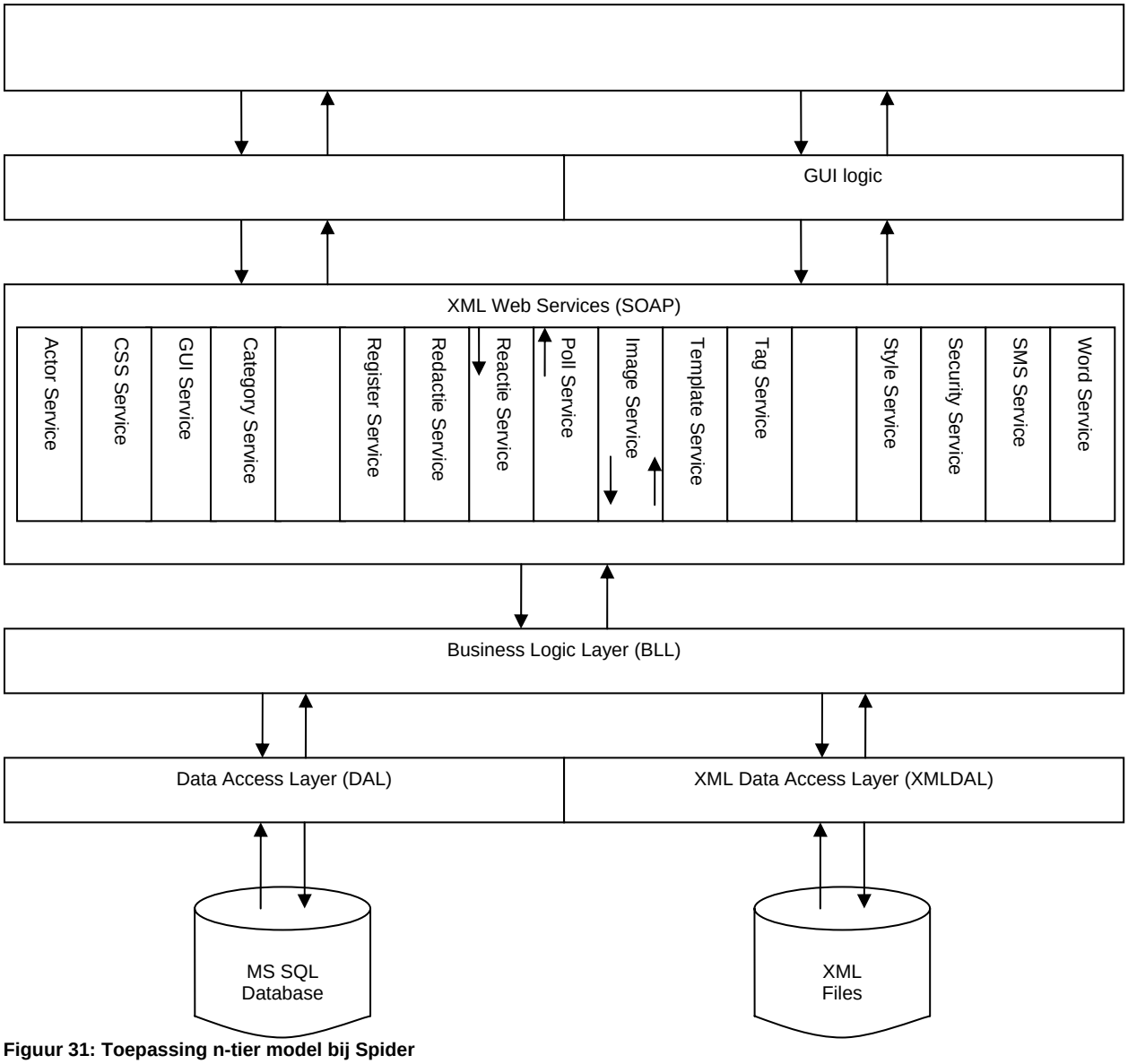
Figuur 30: Tekstuele beschrijving klasse contentitem

6. Technische structuur

In dit hoofdstuk zal enerzijds worden beschreven Hoe de applicatie Spider is opgebouwd. Dit is van groot belang omdat het bij dit project om een aanvulling op de huidige applicatie gaat. De aanvulling moet dus gebruik maken van dezelfde technische structuur als de rest van de applicatie. Anderzijds zal in dit hoofdstuk worden beschreven welke hard- en software er nodig is voor de ontwikkeling en implementatie van het systeem(concept).

6.1. Applicatiearchitectuur

Zoals eerder beschreven in dit document is Spider een webapplicatie (ASP.NET) die geprogrammeerd is in C#.NET. Er wordt geprogrammeerd volgens de n-tier architectuur. Dat wil zeggen dat de gehele applicatie is onderverdeeld in verschillende lagen. Elk van deze lagen voert specifieke taken uit zoals het laten zien van een gebruikersinterface, de opslag van data of het uitvoeren van berekeningen. In het geval van Spider zijn er 5 lagen. De onderstaande afbeelding illustreert dit 5-lagenmodel.

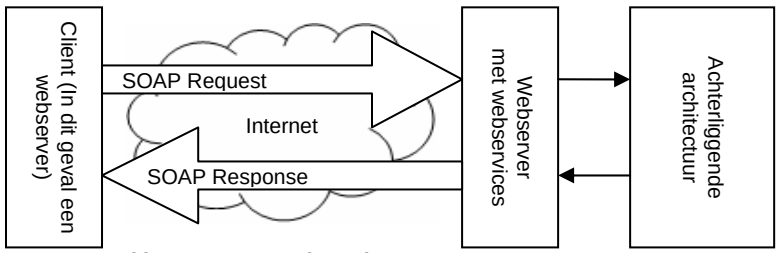


Figuur 31: Toepassing n-tier model bij Spider

De lagen van de n-tier architectuur mogen niet verward worden met de 7 lagen uit het OSI-model. N-tier is een applicatiearchitectuur en bevindt zich in zijn geheel binnen laag 7 van het OSI-model (applicatielaag).

6.2. XML webservices

In de vorige paragraaf is al duidelijk geworden dat Spider voor een belangrijk deel gebruik maakt van zogenaamde XML webservices. Een webservice is een component dat via het internet (HTTP) methodes beschikbaar stelt om programmafunctionaliteit te kunnen aanroepen vanuit verschillende programma's op verschillende platformen. Een webservice kan dus vanuit een webpagina benaderd worden, maar ook vanuit een windows- of linuxapplicatie. Het aanroepen van een methode van een webservice gebeurt door middel van SOAP-berichten. SOAP is in principe niets meer dan een aantal afspraken over de structuur van XML-boodschappen die verstuurd en ontvangen kunnen worden door een webserver.



Figuur 32: Werking van XML Webservices

In de volgende tabel staan de verschillende webservices van Spider met per webservice de beschikbare webmethodes.

Webservice	Beschrijving	Webmethode
ActorService	De ActorService is verantwoordelijk voor het afhandelen van verzoeken op het gebied van gebruikers, gebruikersrollen en bevoegdheden.	SetActorRightContentCatSubCat
		ChangePassword
		ActorIsAfnemer
		ActiveActor
		DeleteActorRole
		GetNotActorRole
		ChangeOtherPassword
		SetActor
		HaveRole
		DeleteActor
		DeActiveActor
		ActorIsModderator
		SetActorRegister
		GetNotActorRightContentCatSubCat
		GetAllActorRoles
		GetActorRole
		ChangeActorProfile
		ActorIsMarbel
		GetRolProfileType
		AddActorRole
		GetAllRol
		GetAllActor
		ActorIsGeregistreerdeAfnemer
		ActorIsMarketingMedewerker
		ActorIsOpmaakRedacteur
		GetActorProfile
		ActorIsAdvertentieBeheerder

Webservice	Beschrijving	Webmethode
		GetActorRightContentCatSubCat
		ActorIsSiteBeheerder
		GetActorProfileXML
		DeleteActorRightContentCatSubCat
		ActorIsRedacteur
		ActorIsWinkelbeheerder
		GetActor
		ActorIsGeautomatiseerdeProcessen
		ActorIsHoofdRedacteur
		GetProfileTypes
CategoryService	De CategoryService zorgt voor het aanmaken, verwijderen en weergeven van categorieën en subcategorieën.	CreateContentSubCategory
		GetSelectedSubCategory
		CreateContentCategory
		UpdateContentCategory
		DeleteContentCategory
		GetAllSubCategory
		GetContentCategory
		DeleteContentSubCategory
		GetCategory
		GetAllSelectedSubCategory
		CreateCatSubCat
		DeleteCatSubCat
		GetAllSubCategoryNotInCategory
		GetAllCategory
CSSService	De CSSService maakt, verwijderd en bewerkt CSS stylesheets en vraagt deze op.	Create
		Delete
		Get
		Koppel
		GetAllCSSDocument
		Update
		GetAll
		VerwijderKoppelingCSS
		GetAllCSSNotDocument
DocumentService	De DocumentService muteert en vraagt documenten op.	GetDocumentFile
		Statistics
		GetDefaultMobileDocument
		CheckLockingDocument
		GetDefaultDocument
		GetDocument
		GetDocumentLeesVerder
		DeleteDocument
		CreateDocument
		LockingDocument
		GetAllDocument
GuiService	De GuiService voert diverse taken uit die gerelateerd zijn aan de gebruikersinterface.	SetColor
		GetSysparam
		SetSysparam
		GetMenu
		GetGuiColors
ImageService	De ImageService is verantwoordelijk voor het uploaden en het laten zien van afbeeldingen.	DeleteImage
		GetImagePagesCount
		GetImage
		GetPagingImage
		UploadImage
		SetImage

Webservice	Beschrijving	Webmethode
PollService	De PollService maakt en configureert Polls.	GetStylePoll
		GetPollImage
		SetPoll
ReactieService	De ReactieService voegt commentaar toe aan content en geeft deze weer.	SetReactieModderate
		PutContentReactie
		CheckReactieModderate
		ContentCount
		GetAllReactie
RedactieService	De RedactieService zorgt voor de afhandeling van redactionele taken.	GetPagingPagesCount
		DeleteContentContainerKoppeling
		GetContentVersion
		AddContentContainerKoppeling
		Modderate
		GetAllContentType
		CreateContentKoppeling
		PutContent
		CreateNewContent
		DeleteContentCheck
		DeleteContentKoppeling
		DeleteContent
		GetAllContentKoppeling
		GetAllContentContainerKoppeling
		GetPagingContentModderate
		LockingContentContainer
		GetContentXML
		CheckLockingContentContainer
		GetPagingContent
		GetAllContent
		GetContent
		GetPagingPagesCountModderate
RegisterService	De RegisterService maakt nieuwe gebruikers aan en activeert ze.	RegisterActor
		RegisterActorActivate
SearchService	De SearchService laat op basis van een query zoekresultaten zien.	GetSearchResults
SecurityService	De SecurityService verifieert gebruikersnamen en wachtwoorden en bepaalt permissies.	EnableGroup
		GetUserRolesWithoutUser
		GetUserName
		ChangePassword
		RemoveUserRole
		AddUser
		GetUserRolesWithUser
		GetGroups
		ChangeGroupDescription
		EnableRight
		VerifyRight
		GetGroupRoles
		DisableUser
		ChangeRoleDescription
		AddUserRole
		RemoveUserGroup
		UserHasRole
		AddRole
		RemoveRoleRight
		DeleteUser
		UserRoles

Webservice	Beschrijving	Webmethode
		GetUserGroupsWithUser
		DisableRole
		RemoveGroupRole
		GetRoles
		ChangeRightDescription
		AddGroupRole
		DisableRight
		VerifyUser
		GetRights
		EnableUser
		AddRoleRight
		AddGroup
		AddUserGroup
		VerifyToken
		GetUserGroupsWithoutUser
		DisableGroup
		AddRight
		ChangeOtherUsersPassword
		EnableRole
SMSService	De SMSService kan SMS-berichten versturen naar leden die een 06-nummer hebben opgegeven.	VerstuurSMS
StyleService	De StyleService muteert XSL-T Styles.	DeleteStyle
		CheckLockingStyle
		GetStyle
		GetAllStyle
		LockingStyle
		CreateStyle
TemplateService	De TemplateService muteert content templates.	GetTemplateIsLayout
		CheckLockingTemplate
		CreateTemplate
		GetAllTemplate
		DeleteTemplate
		GetTemplate
		LockingTemplate

Bij de ontwikkeling aan Spider zal voor een groot deel gebruik worden gemaakt van bestaande services en methodes. Voor de ontwikkeling van een document-, template-, css- en style-editor moeten de document-, template- css- en styleservices worden uitgebreid met nieuwe webmethodes.

6.3. Benodigde hard- en software

Bij de ontwikkeling aan Spider zal gebruik worden gemaakt van de volgende hard- en software.

- Een modern werkstation met voldoende processorcapaciteit en geheugen (2GHz/512MB)
- Internet aansluiting
- Microsoft Windows XP
- Internet Information Services 5.1
- SourceGear Vault
- Microsoft Office XP
- Microsoft Visual Studio 2003 .NET

Bij de ontwikkeling en implementatie van dit project zal gebruik worden gemaakt van bestaande hard- en software. Er zullen geen nieuwe (extern ontwikkelde) componenten worden gebruikt.

7. Pilotplan

In dit hoofdstuk wordt beschreven hoe de te ontwikkelen functionaliteit wordt ondergebracht in verschillende pilots en wanneer deze pilots worden ontwikkeld. Een pilot is een deel van het systeem dat apart wordt ontwikkeld en gebruikt kan worden. Het pilotplan bevat een geprioriteerde lijst van pilots die sequentieel of parallel ontwikkeld en ingevoerd gaan worden. Verder wordt voor elke pilot wordt een inschatting gemaakt van de benodigde middelen en tijd.

7.1. Beschrijving pilots en prioritering

In deze paragraaf wordt aangegeven welke pilots er komen en uit welke pilotdelen ze bestaan. Verder wordt de prioriteit van de pilots vastgesteld. De nummering van de pilots geeft een indicatie van de prioriteit. Een gedetailleerde prioritering wordt beschreven in figuur 37.

Pilot 1: Documenteditor	
Omschrijving	De eerste pilot bevat alle functionaliteit voor de verbeterde documenteditor. Deze pilot zal de bestaande documenteditor van Spider vervangen.
Pilotdelen	• Ontwerpen van de GUI (basis) • Ontwikkeling basis GUI (basis) • Ontwikkeling help-systeem (comfort) • Code opschonen / herstructureren / commentariseren (comfort)
Schatting benodigde tijd	2 weken
Gewenst kwaliteitsniveau	Hoog, de documenteditor is qua functionaliteit de belangrijkste pilot omdat hier bij de gebruiker het meeste behoefte naar is.
Planning	Eerst wordt de GUI ontworpen en ontwikkeld. Dit zijn de basis-activiteiten. Tijdens de tweede iteratie wordt het help-systeem ontwikkeld dat ook gebruikt kan worden voor de andere pilots. Tenslotte wordt de code opgeschoond, hergestructureerd en van commentaar voorzien.

Figuur 33: Beschrijving pilot 1 – Documenteditor

Pilot 2: Template-editor	
Omschrijving	Deze pilot lijkt qua functionaliteit veel op de documenteditor. Daarom is de doorlooptijd van deze pilot korter dan de eerste. Door middel van de template-editor kunnen content templates worden ontworpen. De templates worden opgeslagen in XML formaat.
Pilotdelen	• Ontwerpen van de GUI (basis) • Ontwikkeling van de GUI (basis) • Code opschonen / herstructureren / commentariseren (comfort) • GUI op basis van XML-definitie (comfort)
Schatting benodigde tijd	1,5 week
Gewenst kwaliteitsniveau	Gemiddeld, Een contenttemplate wordt over het algemeen ontworpen als de website wordt ingericht. Daarna worden er zelden wijzigingen aan aangebracht.
Planning	Eerst wordt de GUI ontworpen en ontwikkeld. In de tweede iteratie wordt de code opgeschoond en wordt er een systeem ontwikkeld om de GUI te laten baseren op basis van een XML-definitie. Op deze manier kan nieuwe functionaliteit binnen Spider eenvoudig worden toegevoegd aan de GUI door de XML-definitie aan te passen.

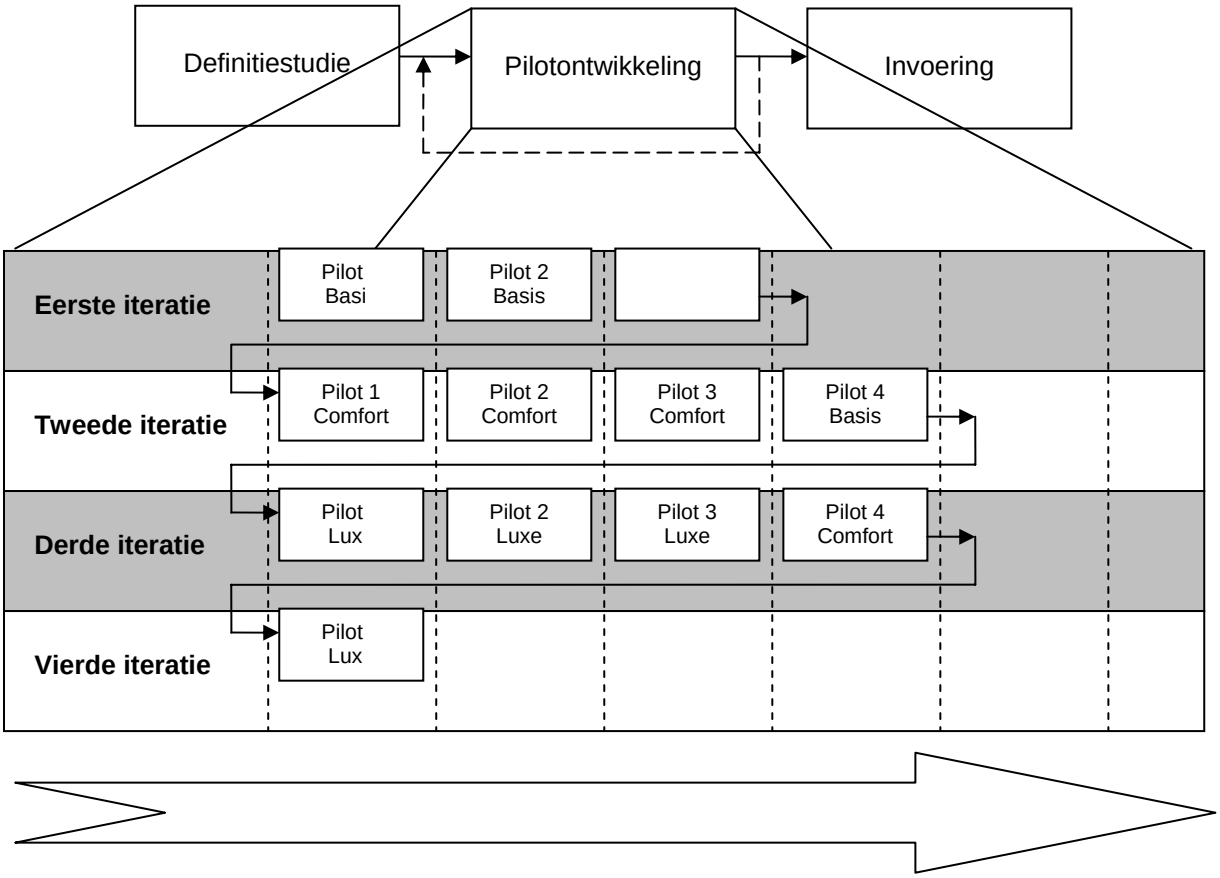
Figuur 34: Beschrijving pilot 2 – Template-editor

Pilot 3: CSS editor	
Omschrijving	De eerste pilot bevat alle functionaliteit voor de CSS-editor. Hiermee kan op eenvoudige wijze een stylesheet worden samengesteld. De stylesheet moet worden opgeslagen in XML-formaat. Daarvoor moet er een XSL-Transformatie plaatsvinden.
Pilotdelen	• Ontwerpen XML-model voor CSS (basis) • XML -> CSS transformatie (basis) • Ontwerpen van de GUI (basis) • Ontwikkeling van de GUI (basis) • Code opschonen / herstructureren / commentariseren (comfort) • XML DAL gereedmaken voor opslag CSS-XML (luxe)
Schatting benodigde tijd	2 weken
Gewenst kwaliteitsniveau	Hoog, er is bij de gebruikers van Spider veel vraag naar functionaliteit om op een makkelijke manier de opmaak te veranderen. Dit moet mogelijk worden gemaakt door de CSS editor.
Planning	Eerst moet er een XML-model ontworpen worden om CSS in op te kunnen slaan. Als dit gebeurd is, kan de transformatie van XML naar CSS via XSL-T ontwikkeld worden. Vervolgens kan de GUI ontworpen en ontwikkeld worden. In de tweede iteratie wordt de code opgeschoond en gecommentariseerd. Verder wordt de reeds ontwikkelde XML Data Access Layer aangepast om XML op te kunnen slaan.

Figuur 35: Beschrijving pilot 3 – CSS editor

Pilot 4: Style-editor	
Omschrijving	De Style-editor is veel complexer dan de editors uit de eerste twee pilots. Dit komt doordat deze editor de complexe XSL-T sheets moet kunnen interpreteren. Deze pilot heeft een lagere prioriteit dan de voorgaande pilots. Het ontwerpen van een GUI voor het bewerken van XSL-T is een complexe taak waarbij goed gekeken moet worden of de interface niet voor beperkingen zorgt in de functionaliteit van Spider.
Pilotdelen	• Ontwerpen van de GUI (basis) • Ontwikkeling van de GUI (basis) • Code opschonen / herstructureren / commentariseren (comfort)
Schatting benodigde tijd	2 weken
Gewenst kwaliteitsniveau	Gemiddeld
Planning	In de eerste iteratie wordt de GUI ontworpen en ontwikkeld. Vervolgens wordt in een tweede iteratie de code opgeschoond.

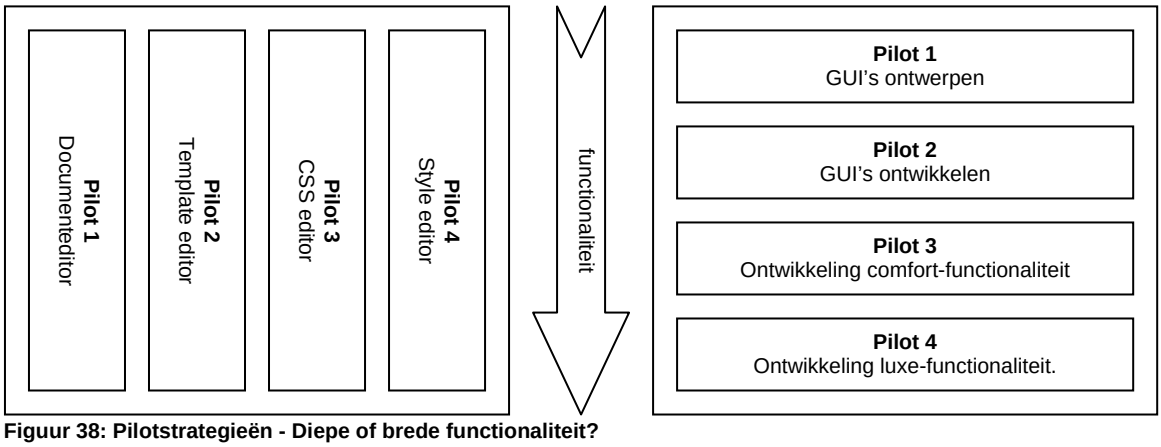
Figuur 36: Beschrijving pilot 4 – Style-editor



Figuur 37: Prioritering – Parallelle ontwikkeling en iteratie binnen de fase pilotontwikkeling

7.2. Beschrijving pilot-strategie

Er zijn verschillende manieren om een ontwikkelingstraject op te splitsen in verschillende pilots. Een voorbeeld van de afwegingen die gemaakt zijn bij het definiëren van de verschillende pilots, is de keuze tussen diepe en brede functionaliteit. Bij dit ontwikkeltraject is er gekozen voor de strategie waarbij de pilots een diepe functionaliteit hebben. Voordeel hiervan is dat als een bepaalde pilot wegens tijdgebrek niet meer ontwikkeld kan worden, de andere pilots nog wel ingevoerd kunnen worden. De kans is dus klein dat er geen resultaat wordt behaald voor de opdrachtgever.
De volgende afbeelding illustreert deze twee strategieën. Ook de pilotdelen zijn op deze manier ingericht. Zelfs na de eerste iteratie is er al resultaat voor de opdrachtgever.



Figuur 38: Pilotstrategieën - Diepe of brede functionaliteit?

7.3. Pilot acceptatie

Zoals beschreven in het plan van aanpak wordt er bij dit traject ontwikkeld volgens de iteratiestrategie ‘incrementeel ontwikkelen (RAD)’. Dat betekent dat de invoering in één keer plaats vind. De activiteit pilot acceptatie is een onderdeel van de fase invoering. De acceptatietest zal daarom aan het einde van het traject plaats vinden. De acceptatietest wordt uitgevoerd door de opdrachtgever op basis van de volgende criteria.

- Komt de functionaliteit van de pilot overeen met de systeemeisen? (per pilot)
- Is voldaan aan de integriteits- en interface-eisen? (per pilot)
- Is de systeemdocumentatie op orde? (per pilot)
- Kan het systeem worden beheerd door Marbel Software?

Bij een positieve uitkomst van de acceptatietest kan het systeem in gebruik worden genomen.

Pilotontwikkelplan pilot 1: Documenteditor

Afstudeeropdracht

student:	Dhr. M. Verheul
keurend docent:	Dhr. J.J. van Beijnum
afstudeerbedrijf:	Marbel Software B.V. te Noordwijkerhout
bedrijfsmentor:	Dhr. Ing. G. Scheeres
datum:	10 juni 2005
afstudeerrichting:	Informatica en Informatiekunde: VIA
afstudeerblok	2005-1.1
versie:	1

Inhoudsopgave

1. Inleiding

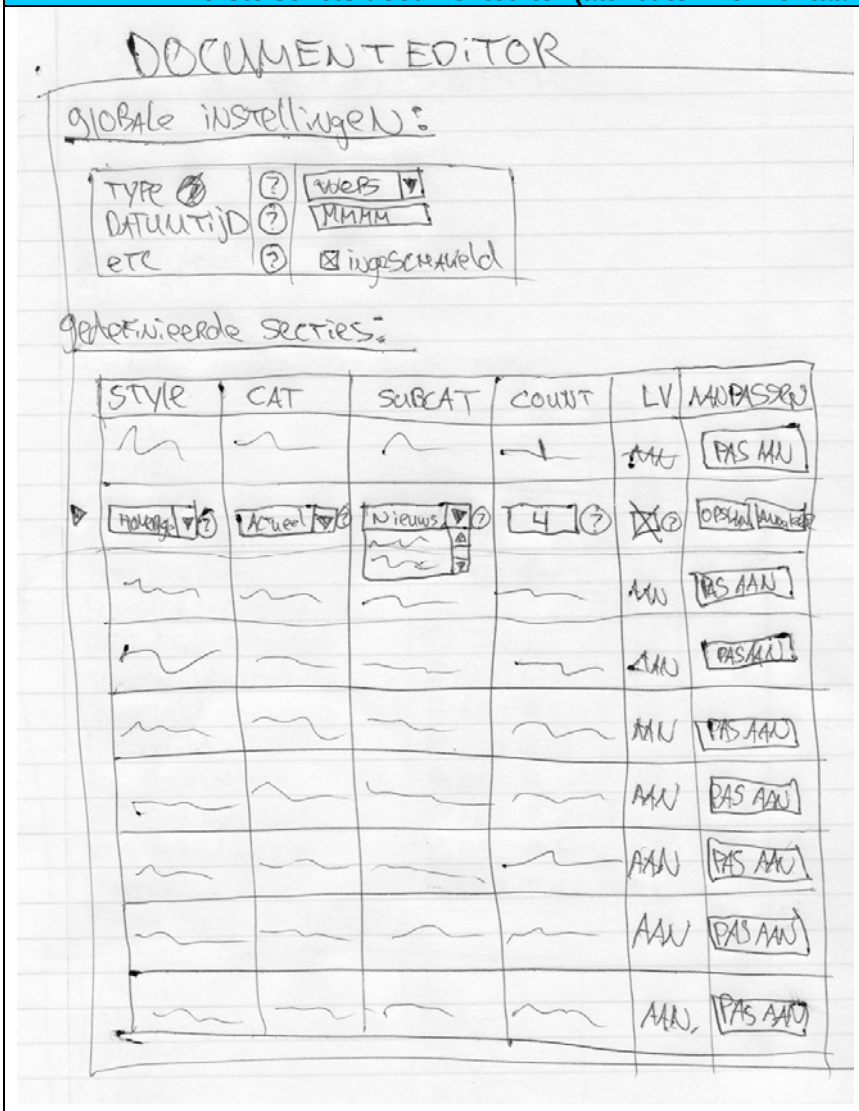
Dit document is een product van de tweede fase binnen de ontwikkelmethodiek IAD, de pilotontwikkeling. Het is geschreven in het kader van het afstudeerproject bij Marbel software. Dit pilotontwikkelplan gaat over de eerste pilot die ontwikkeld gaat worden, de documenteditor. In hoofdstuk 2 wordt de functionele structuur van de pilot beschreven. U leest hier onder andere welke schermen er ontwikkeld gaan worden, wat de functionaliteit per scherm is en wat de mogelijke acties zijn. Tot slot wordt in hoofdstuk 3 de achterliggende technische structuur van de pilot beschreven.

2. Globaal functionele structuur pilot

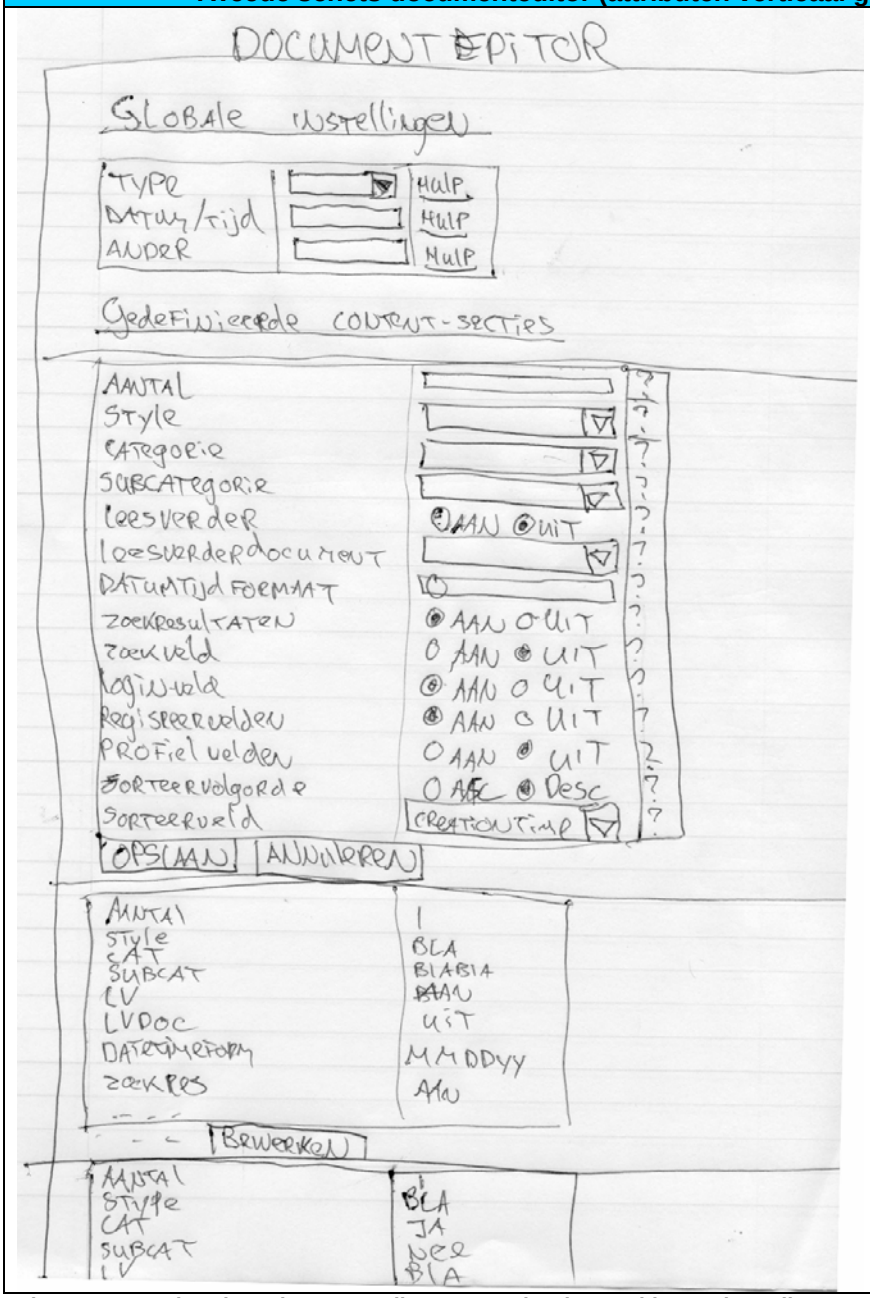
In dit hoofdstuk wordt de functionele structuur van de pilot documenteditor beschreven. In de definitiestudie is in de systeemeisen al globaal beschreven aan welke eisen de documenteditor moet voldoen. Deze zullen in dit hoofdstuk worden uitgewerkt.

2.1. Schermontwerpen

De gebruikersinterface voor de documenteditor moet duidelijk zijn voor beginnende gebruikers. De interface moet de gebruiker ondersteuning verlenen bij het opbouwen van een documentdefinitie. Op basis van de volgende schetsen zijn er schermontwerpen gemaakt.

Eerste schets documenteditor (attributen horizontaal gepositioneerd)	
	In deze eerste schets bestaat de documenteditor uit twee tabellen. Een met globale instellingen zoals de titel van het document en de datumtijd-instellingen en een tabel met daarin de verschillende content-secties. Achter elke rij staat een ‘pas aan’-knop. Als daar op gedrukt wordt verandert de rij in een aanpasbare rij. Afhankelijk van het soort veld dat aangepast moet worden, wordt er een ander schermelement geboden. Voor het aanpassen van vrije text is er een textbox, voor meerkeuze-opties een dropdownlistbox en voor aan/uit-opties een checkbox. Achter elk element dat gewijzigd kan worden, bevindt zich een vraagteken waarop geklikt kan worden. Er verschijnt dan helptekst. Een belangrijk nadeel van dit ontwerp is dat er maar ruimte is voor een beperkt aantal kolommen. Een content-sectie kan echter meer dan 15 attributen hebben. Deze passen niet als kolommen op het scherm.

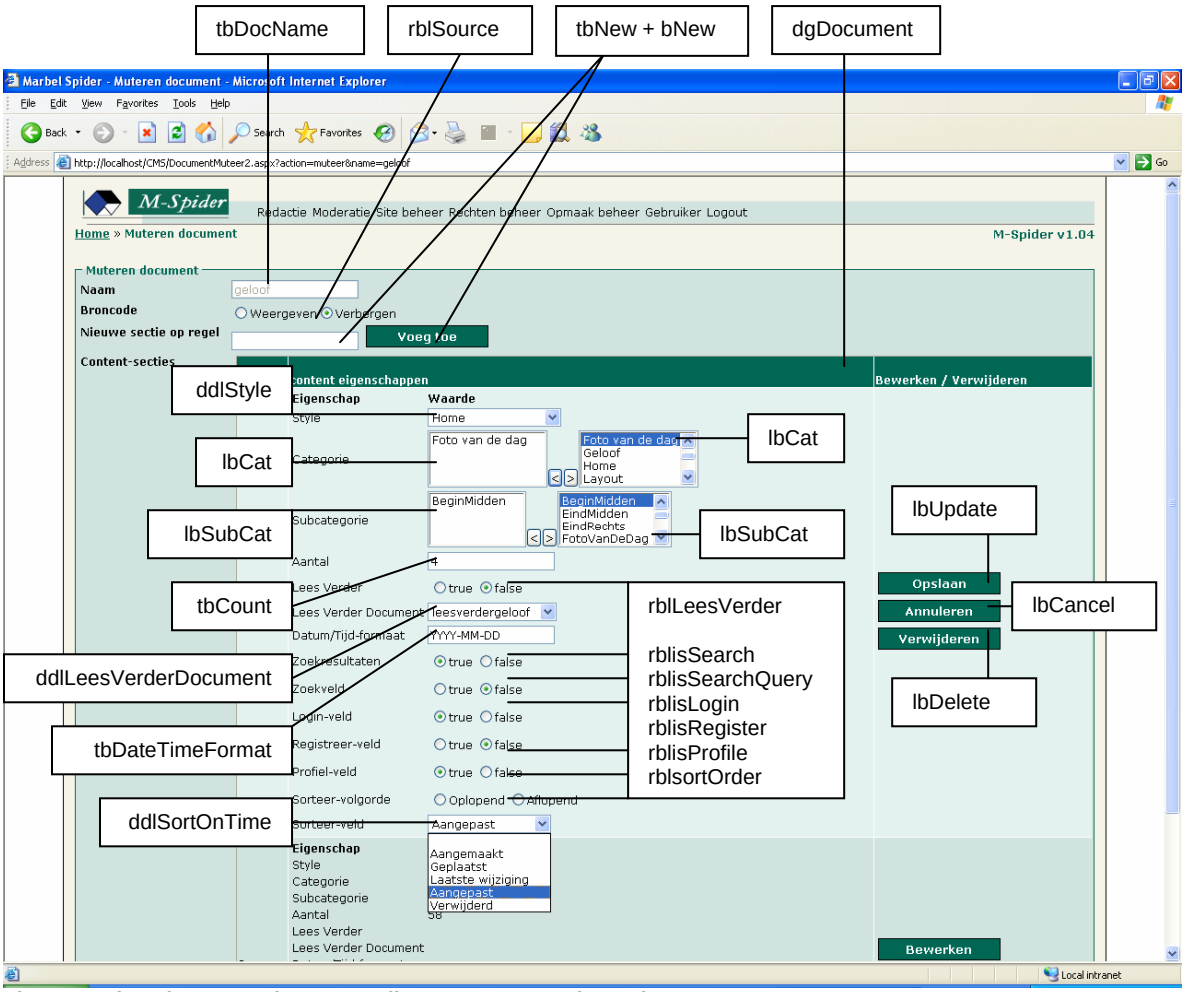
Figuur 1: Eerste schets documenteditor met horizontaal gepositioneerde attributen

Tweede schets documenteditor (attributen verticaal gepositioneerd)	
	Deze tweede schets gaat van het zelfde principe uit als het eerste ontwerp. De Datagrid met het beperkte aantal kolommen is hier echter vervangen door meerdere tabellen onder elkaar. De verschillende attributen worden hier onder elkaar in één tabel weergegeven zodat er geen ruimtegebrek meer kan optreden. Als er een regel moet worden aangepast Klikt de gebruiker op de ‘bewerken’-knop. In de tweede kolom van het betreffende item worden de waarden vervangen door schermelementen om de waarden aan te passen. Een nadeel van dit ontwerp is dat de gebruiker naar onderen moet scrollen als hij aan een document werkt dat veel content-secties bevat. Ook bij dit ontwerp is er weer de mogelijkheid tot het verkrijgen van ondersteuning door middel van het klikken op de helpknoppen.

Figuur 2: Tweede schets documenteditor met verticaal gepositioneerde attributen.

2.2. Digitaal ontwerp documenteditor

Op basis van de schetsen van paragraaf 2.1 en prototyping is het volgende digitale ontwerp gemaakt voor de documenteditor.



Figuur 3: Digtaal ontwerp documenteditor met namen schermelementen

Een belangrijke eis uit de systeemeisen is dat de documenteditor voorzien wordt van ingebouwde help-functionaliteit. Op welke wijze deze functionaliteit wordt geïmplementeerd, is op dit moment nog niet bekend.

3. Globaal-technische structuur pilot

In dit hoofdstuk wordt de technische structuur van de eerste pilot (documenteditor) zo gedetailleerd mogelijk beschreven. Omdat er gebruik wordt gemaakt van hergebruik zal deze technische structuur voor een deel terugkomen in de andere pilots.

3.1. Beschrijving documentdefinitie

De taak van de documenteditor is het genereren van een XML-bestand met een documentdefinitie. De XML-bestanden moeten voldoen aan een bepaalde structuur. Deze structuur is momenteel nergens gedocumenteerd. De structuur moet daarom achterhaald worden door in de broncode van Spider te zoeken. De belangrijkste deel van een documentdefinitie wordt gedefinieerd in de content-nodes die onderdeel zijn van een section-node. In de broncode van de klasse ‘section’, onderdeel van de Marbel.CMS.Businesslogic-namespace is daarom het grootste deel van de structuur van een document te achterhalen. Als de structuur eenmaal in kaart is gebracht, moet deze worden vastgelegd. Hierbij zijn een aantal aspecten van belang:

1. **De documenteditor moet kunnen controleren of een documentdefinitie geldig is opgebouwd.**

Door middel van een XML Schema Definition (XSD), kan de geldigheid van een XML-document gecontroleerd worden. Een XSD-bestand is een op XML-gebaseerd bestand waarin de structuur en relaties binnen een XML-bestand kunnen worden vastgelegd. XSD biedt zelfs mogelijkheden om de mogelijke waarden van een bepaald attribuut vast te leggen. Het XSD-bestand in figuur 4 (pagina 8) beschrijft de structuur van een Spider-documentdefinitie.

2. **De documenteditor moet een documentdefinitie kunnen opbouwen.**

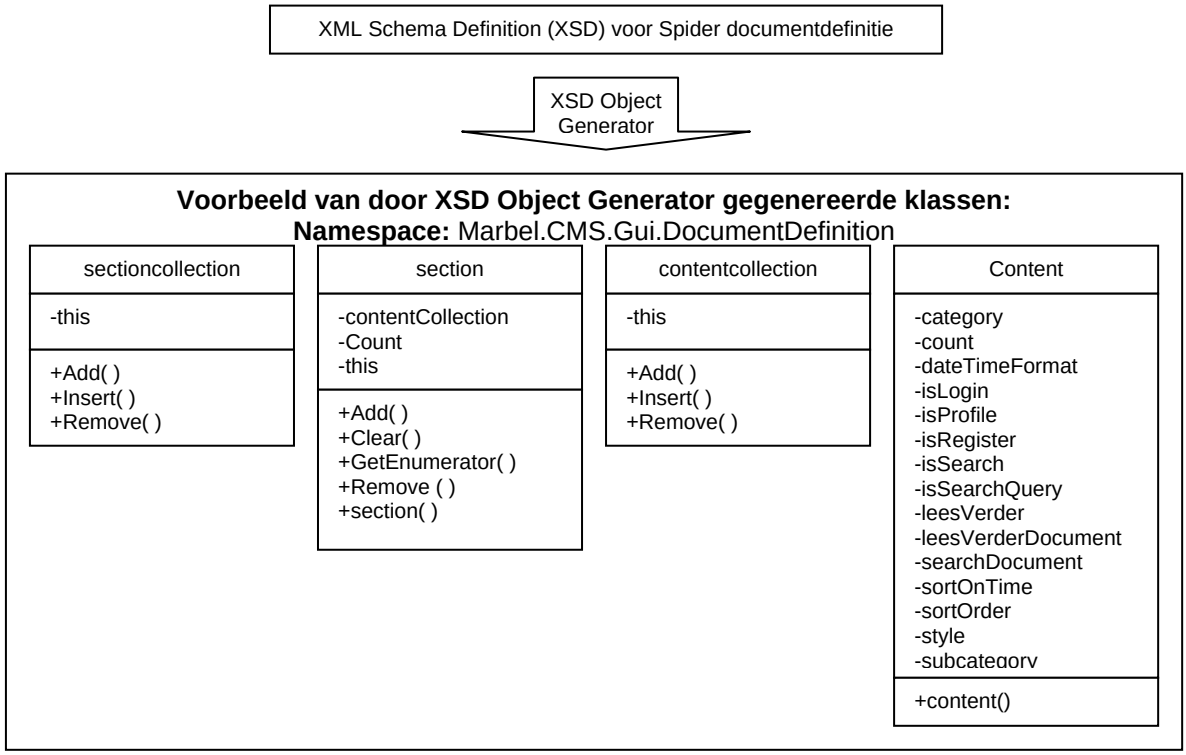
Normaal gesproken vereist het programmatisch opbouwen van een XML-bestand veel programmeerwerk en is het bovendien gevoelig voor fouten. Microsoft heeft tool ontwikkeld die op basis van een XSD-schema C# klassen genereert waarmee een XML-bestand gemaakt kan worden dat exact voldoet aan het gebruikte XSD-schema. Op deze manier kan een XML-bestand gemanipuleerd worden alsof het een object is. Als alle nodige bewerkingen op de objecten gedaan zijn, moeten de objecten omgevormd worden naar XML. Dit proces wordt serialisatie genoemd. Het omgekeerde proces (van XML naar objectinstantie) is deserialisatie. Figuur 5 (pagina 9) illustreert het objectgeneratie- en serialisatieproces aan de hand van een voorbeeld.

3. **De documenteditor moet ondersteuning bieden bij het opbouwen van een documentdefinitie**

In een XSD-bestand kan alleen de structuur van een XML-bestand worden vastgelegd. Binnen de documenteditor is het echter ook van belang dat de gebruiker voldoende ondersteuning krijgt bij het opbouwen van een documentdefinitie. Deze ondersteuning kan geboden worden door middel van een korte uitleg bij elk attribuut dat de gebruiker in dient te vullen. Omdat er regelmatig nieuwe functionaliteit voor Spider wordt ontwikkeld, veranderen ook de attributen en hun mogelijke waardes binnen een documentdefinitie. Om de ondersteuning bij de documenteditor toch op een eenvoudige manier up-to-date te kunnen houden, wordt deze vastgelegd in een apart XML-bestand. Een gedeelte van zo’n bestand is te vinden in figuur 6. (pagina 10).

```
<?xml version="1.0" ?>
<xs:schema id="SpiderDocument" targetNamespace="http://tempuri.org/spiderdocument.xsd"
xmlns:mstns="http://tempuri.org/spiderdocument.xsd"
 xmlns="http://tempuri.org/spiderdocument.xsd" xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns:msdata="urn:schemas-microsoft-com:xml-msdata" attributeFormDefault="qualified"
elementFormDefault="qualified">
 <xs:element name="document">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="head" minOccurs="0" maxOccurs="unbounded">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="title" type="xs:string" minOccurs="0" />
       <xs:element name="custom" type="xs:string" minOccurs="0" />
      </xs:sequence>
     </xs:complexType>
    </xs:element>
    <xs:element name="sections" minOccurs="0" maxOccurs="unbounded">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="section" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
         <xs:sequence>
          <xs:element name="content" minOccurs="0" maxOccurs="unbounded">
           <xs:complexType>
            <xs:attribute name="count"  type="xs:string" use="optional" />
            <xs:attribute name="style"  type="xs:string" use="optional" />
            <xs:attribute name="category"  type="xs:string" use="optional" />
            <xs:attribute name="subcategory"  type="xs:string" use="optional" />
            <xs:attribute name="leesVerderDocument"  type="xs:string" use="optional" />
            <xs:attribute name="dateTimeFormat"  type="xs:string" use="optional" />
            <xs:attribute name="isSearch" type="xs:string" use="optional" />
            <xs:attribute name="isSearchQuery" type="xs:string" use="optional" />
            <xs:attribute name="isLogin" type="xs:string" use="optional" />
            <xs:attribute name="isRegister" type="xs:string" use="optional" />
            <xs:attribute name="isProfile" type="xs:string" use="optional" />
            <xs:attribute name="sortOrder" type="xs:string" use="optional" />
            <xs:attribute name="sortOnTime" type="xs:string" use="optional" />
           </xs:complexType>
          </xs:element>
         </xs:sequence>
        </xs:complexType>
       </xs:element>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
   <xs:attribute name="type" form="unqualified" type="xs:string" />
   <xs:attribute name="dateTimeFormat" form="unqualified" type="xs:string" />
  </xs:complexType>
 </xs:element>
</xs:schema>
```

Figuur 4: XML Schema Definition (XSD) van de Spider Documentdefinitie

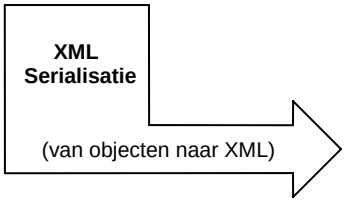


Toepassing van de gegenereerde klassen in C#:

```
using Marbel.CMS.Gui.DocumentDefinition;
public void genereerTweeSectiesMetContent()
{
// Maak objectinstanties van de section- en contentklassen:
section objSection1 = new section();
content objContent1 = new content();
section objSection2 = new section();
content objContent2 = new content();

// Vul eigenschappen van contentobjecten en voeg contentobjecten toe aan sectionobjecten
objContent1.count = "3";
objContent1.category = "nieuws";
objSection1.Add(objContent1);
objContent2.category = "belangrijk";
objContent2.leesVerder = "true";
objSection2.Add(objContent2);

// Instantieer XmlSerializer- en XmlTextWriterobjecten en voer serialisatie uit
XmlSerializer ser = new XmlSerializer(typeof(section));
XmlTextWriter writer = new XmlTextWriter("tweesections.xml", System.Text.Encoding.UTF8);
writer.Formatting = Formatting.Indented;
ser.Serialize(writer, objSection1);
ser.Serialize(writer, objSection2);
writer.Close();
}
```



Uitvoer: XML-bestand 'tweesections.xml':

```
<section>
    <content count="3" category="nieuws" />
</section>
<section>
    <content category="belangrijk" leesVerder="true" />
</section>
```

Figuur 5: Toepassing van de XSD Object Generator en het serialiseren van XML

```
<?xml version="1.0" encoding="utf-8" ?>
<editor name="documenteditor">
 <category name="document" path="document">
  <attribute name="type" description="Kies het type van dit document.">
   <value name="web" type="multiplechoice" description="Documenten van het type web kunnen
          weergegeven worden in een webbrowser." />
   <value name="anders" type="string" description="Vul een ander type in." />
  </attribute>
  <attribute name="dateTimeFormat" description="Manier waarop datums en tijden worden
             weergegeeven in dit document">
   <value name="format" type="string" description="Vul het datumtijd-formaat in. YYYY is een
          volledig jaar, MMMM is een volledig uitgeschreven maand. DDDDD is een volledig
          uitgeschreven dag, etc." />
  </attribute>
  <attribute name="currentTimeFormat" description="Manier waarop de huidige datum en tijd
             wordt weergegeven in dit document.">
   <value name="format" type="string" description="Vul het datumtijd-formaat in. YYYY is een
          volledig jaar, MMMM is een volledig uitgeschreven maand. DDDDD is een volledig
          uitgeschreven dag, etc." />
  </attribute>
 </category>
 <category name="head" path="document/head"></category>
 <category name="title" path="document/head/title" description="De titel van dit document, die
           bovenaan wordt weergegeven in de browser."></category>
 <category name="custom" path="document/head/custom" description="Hier kunt u extra
           functionaliteit toevoegen zoals javascript."></category>
 <category name="sections" path="document/sections"></category>
 <category name="section" path="document/sections/section"></category>
 <category name="content" path="document/sections/section/content">
  <attribute name="count" description="Het aantal artikelen dat moet worden weergegeven.">
   <value name="countint" type="int" description="Vul het aantal in." />
  </attribute>
  <attribute name="style" description="De stijl die bepaalt hoe de artikelen moeten worden
             weergegeven.">
   <value name="stylestring" type="stylestring" description="Kies de stijl." />
  </attribute>
  ...
 </category>
</editor>
```

Figuur 6: Structuur documentdefinitie en ondersteuning in XML-bestand

3.2. Beschrijving events

De volgende tabel die gebaseerd is op het ontwerp uit paragraaf 2.2 laat de verschillende acties zien die door de document-editor moeten worden uitgevoerd.

Events voor Document-editor			
Event	**User action**	**System response**	**Database response**
Page_Load	n.v.t.	• De pagina wordt geladen en de scherm-elementen geïnitialiseerd.	• DocumentService haalt XML-document op. • DataGrid wordt gebonden aan XML-document.
Annuleren_Click	Klikken op de knop 'Annuleren'.	• De wijzigingen worden genegeerd en het vorige scherm wordt geopend. • DocumentService maakt document actief voor bewerking.	n.v.t.
bNew_Click	Klikken op de knop 'Nieuwe sectie'.	n.v.t.	• Nieuwe sectie-node toevoegen aan XML-document. • DataGrid wordt gebonden aan XML-document.
dgDocument_CancelCommand	Klikken op de knop 'Annuleren' in betreffende DataGridRow.	• EditItemIndex wordt op -1 gezet.	• DataGrid wordt gebonden aan XML-document.
dgDocument_DeleteCommand	Klikken op de knop 'Verwij-deren' in de betreffende DataGridRow.	• EditItemIndex wordt op -1 gezet.	• Betreffende sectie-node wordt uit XML-Document verwijderd. • DataGrid wordt gebon-den aan XML-document.
dgDocument_EditCommand	Klikken op de knop 'Bewerken' in de betreffende DatatGridRow.	• EditItemIndex wordt op set huidige item-index gezet. • ContentTemplate kolom wordt geladen in EditItem-modus.	• Schermelementen in ContentTemplate kolom worden gebonden aan XML-Document.
dgDocument_UpdateCommand	Klikken op de knop 'Bewerken' in de betreffende DataGridRow.	• EditItemIndex wordt op -1 gezet. • ContentTemplate kolom wordt geladen in Item-modus.	• Huidige waardes van schermelementen in ContentTemplate kolom worden opgeslagen in XML-document. • DataGrid wordt gebonden aan XML-document.
Opslaan_Click	Klikken op de knop 'Opslaan'.	• DocumentService maakt document actief voor bewerking.	• XML-document wordt gestuurd naar de DocumentService.

Figuur 7: Events veroorzaakt door acties op schermelementen

Pilotontwikkelplan pilot 2: content-template-editor

Afstudeeropdracht

Student:	Dhr. M. Verheul
Examinatoren:	Dhr. J.J. van Beijnum
	Dhr. W. van Vliet
afstudeerbedrijf:	Marbel Software B.V. te Noordwijkerhout
bedrijfsmentor:	Dhr. Ing. G. Scheeres
datum:	10 juni 2005
afstudeerrichting:	Informatica en Informatiekunde: VIA
afstudeerblok	2005-1.1
versie:	1.0

Inhoudsopgave

1. Inleiding

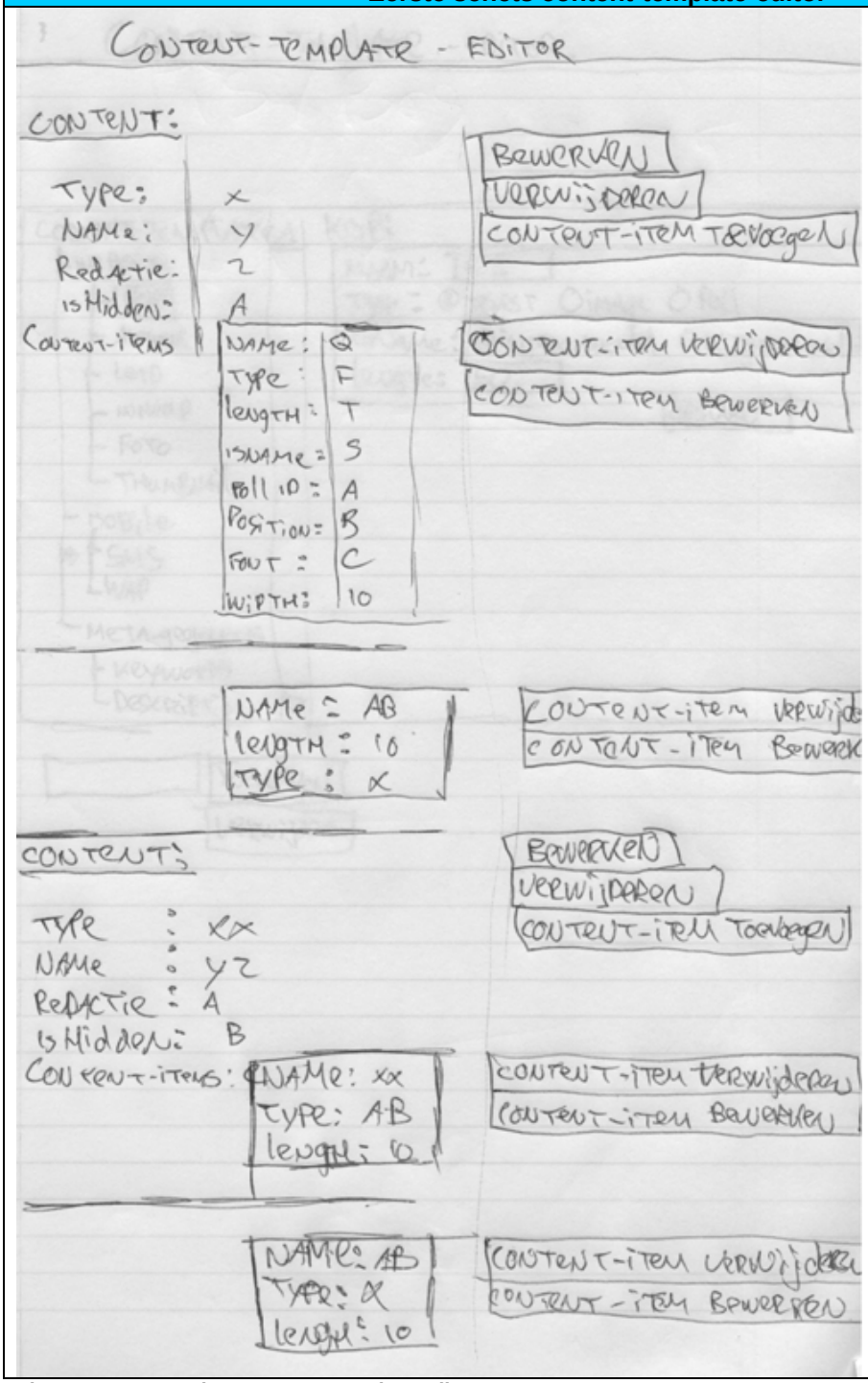
Dit document is een product van de tweede fase binnen de ontwikkelmethodiek IAD, de pilotontwikkeling. Het is geschreven in het kader van het afstudeerproject bij Marbel software. Dit pilotontwikkelplan gaat over de tweede pilot die ontwikkeld gaat worden, de content-template-editor. In hoofdstuk 2 wordt de functionele structuur van de pilot beschreven. U leest hier onder andere welke schermen er ontwikkeld gaan worden, wat de functionaliteit per scherm is en wat de mogelijke acties zijn. Tot slot wordt in hoofdstuk 3 de technische structuur van de pilot beschreven.

2. Globaal functionele structuur pilot

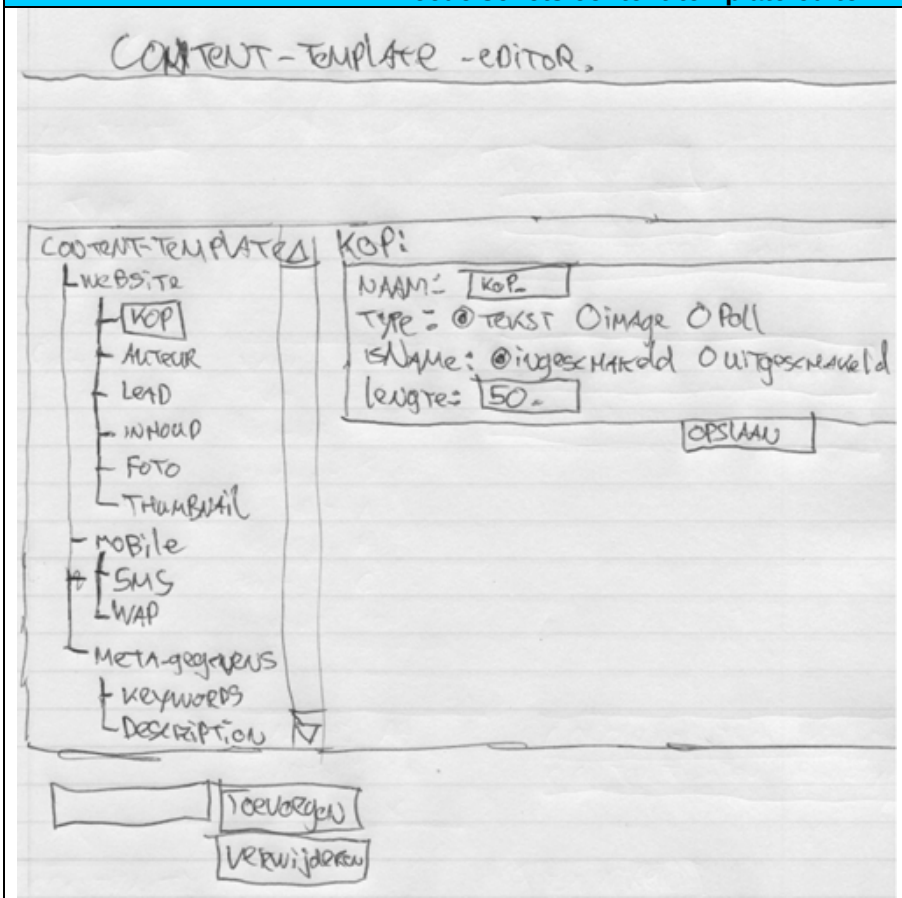
In dit hoofdstuk wordt de functionele structuur van de pilot content-template-editor beschreven. In de definitiestudie is in de systeemeisen al globaal beschreven aan welke eisen de content-template-editor moet voldoen. Deze zullen in dit hoofdstuk worden uitgewerkt.

2.1. Schermontwerpen

De volgende schetsen geven een eerste indicatie van hoe de content-template-editor eruit gaat zien.

Eerste schets content-template-editor	
	De content-template-editor is opgebouwd uit twee niveau's. In het eerste niveau worden de attributen van de content-elementen weergegeven. In het tweede niveau worden de attributen va van de contentitem-elementen weergegeven. Een content- of contentitemelement is aan te passen door op een van de knoppen rechts ervan te klikken.

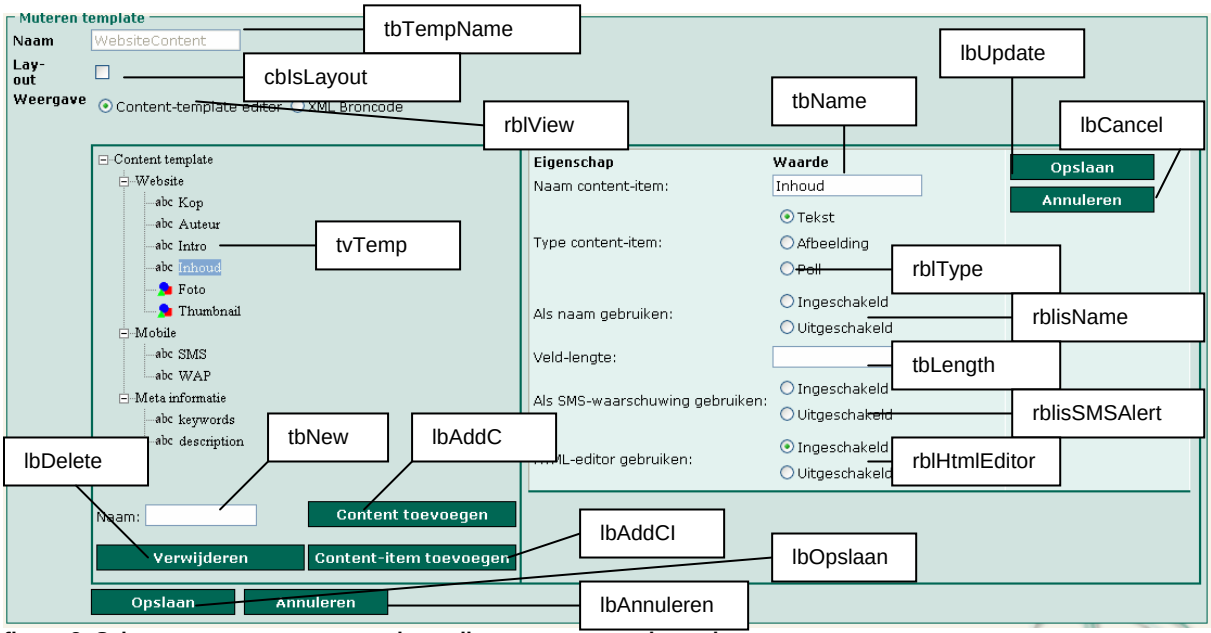
Figuur 1: Eerste schets content-template-editor

Tweede schets content-template-editor	
	Deze tweede schets is een stuk overzichtelijker dan de vorige. Bij dit ontwerp wordt een treeview-control als selectie-mechanisme gebruikt. In het eerste niveau van deze boomstrtuctuur staan de content-elementen en in het tweede niveau de contentitem-elementen. Als in een boom een item wordt geselecteerd, worden de eigen-schappen weergegeven rechts van de boom. Ook verschijnt er een knop ‘bewerken’, die ervoor zorgt dat de eigenschappen bewerkt kunnen worden, d.m.v. een textbox of radiobutton. Content(item)-elementen kunnen worden toegevoegd of verwijderd onderaan de pagina.

Figuur 2: Tweede schets content-template-editor

2.2. Digitaal ontwerp content-template-editor

Op basis van de schetsen van paragraaf 2.1 en door middel van prototyping is het volgende digitale ontwerp gemaakt voor de content-template-editor.



figuur 3: Schermontwerp content-template-editor met namen schermelementen

De editor moet voorzien worden van ingebouwde help-functionaliteit. Het is op dit moment nog niet bekend op welke wijze deze functionaliteit gerealiseerd gaat worden.

3. Globaal-technische structuur pilot

In dit hoofdstuk wordt de technische structuur van de tweede pilot (content-template-editor) beschreven.

3.1. Beschrijving content-template-definitie

Het figuur hieronder is een Xml schema dat aangeeft wat de vereiste structuur van een content-template-xmldocument is. De content-template-editor moet xmldocumenten kunnen genereren die voldoen aan deze definitie.

```
<?xml version="1.0" ?>
<xs:schema id="contentcontainer" targetNamespace="http://tempuri.org/~vsE2E.xsd"
xmlns:mstns="http://tempuri.org/~vsE2E.xsd" xmlns="http://tempuri.org/~vsE2E.xsd"
xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeFormDefault="qualified"
elementFormDefault="qualified">
 <xs:element name="contentcontainer">
  <xs:complexType>
   <xs:choice maxOccurs="unbounded">
    <xs:element name="contentcollection">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="content" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
         <xs:sequence>
          <xs:element name="contentitemcollection" minOccurs="0" maxOccurs="unbounded">
           <xs:complexType>
            <xs:sequence>
             <xs:element name="contentitem" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
               <xs:sequence>
                <xs:element name="imageid" type="xs:integer" minOccurs="0" />
                <xs:element name="question" type="xs:string" minOccurs="0" />
                <xs:element name="tekst" minOccurs="0" maxOccurs="unbounded">
                 <xs:complexType>
                  <xs:sequence>
                   <xs:element name="cdata-section" type="xs:string" minOccurs="0" />
                  </xs:sequence>
                 </xs:complexType>
                </xs:element>
                <xs:element name="choice" minOccurs="0" maxOccurs="unbounded">
                 <xs:complexType>
                  <xs:attribute name="value" form="unqualified" type="xs:string" />
                 </xs:complexType>
                </xs:element>
               </xs:sequence>
               <xs:attribute name="type" form="unqualified" use="optional">
                <xs:simpleType>
                 <xs:restriction base="xs:string">
                  <xs:enumeration value="tekst" />
                  <xs:enumeration value="image" />
                  <xs:enumeration value="poll" />
                 </xs:restriction>
                </xs:simpleType>
               </xs:attribute>
               <xs:attribute name="name" form="unqualified" type="xs:string" />
               <xs:attribute name="length" form="unqualified" type="xs:integer" />
               <xs:attribute name="isName" form="unqualified" type="xs:boolean" />
               <xs:attribute name="htmlEditor" form="unqualified" type="xs:boolean" />
               <xs:attribute name="isSMSAlert" form="unqualified" type="xs:boolean" />
               <xs:attribute name="position" form="unqualified" type="xs:string" />
               <xs:attribute name="font" form="unqualified" type="xs:string" />
               <xs:attribute name="width" form="unqualified" type="xs:integer" />
               <xs:attribute name="height" form="unqualified" type="xs:integer" />
               <xs:attribute name="isPoll" form="unqualified" type="xs:boolean" />
               <xs:attribute name="fontheight" form="unqualified" type="xs:integer" />
               <xs:attribute name="stickcorpulency" form="unqualified" type="xs:integer" />
               <xs:attribute name="bgcolor" form="unqualified" type="xs:string" />
               <xs:attribute name="startingpoint" form="unqualified" type="xs:integer" />
               <xs:attribute name="offsetstick" form="unqualified" type="xs:integer" />
               <xs:attribute name="showVoteCount" form="unqualified" type="xs:boolean" />
```

```
              </xs:complexType>
             </xs:element>
            </xs:sequence>
           </xs:complexType>
          </xs:element>
         </xs:sequence>
         <xs:attribute name="type" form="unqualified" use="optional">
          <xs:simpleType>
           <xs:restriction base="xs:string">
            <xs:enumeration value="content" />
            <xs:enumeration value="meta" />
           </xs:restriction>
          </xs:simpleType>
         </xs:attribute>
         <xs:attribute name="name" form="unqualified" type="xs:string" />
         <xs:attribute name="redactie" form="unqualified" type="xs:string" />
         <xs:attribute name="isHidden" form="unqualified" type="boolean" />
        </xs:complexType>
       </xs:element>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
   </xs:choice>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

figuur 4: Xml schema content-template-xmldocument

3.2. Beschrijving events

De onderstaande tabel beschrijft de acties die geprogrammeerd moeten worden om de events veroorzaakt door de schermelementen af te handelen.

Events voor Template-editor			
Event	**User action**	**System response**	**Database response**
Page_Load	n.v.t.	• De pagina wordt geladen en de scherm-elementen geïnitialiseerd.	• TemplateService haalt XML-document op. • tvTemp wordt gebonden aan XML-document.
tvTemp_Selected-IndexChanged	Klikken op een item in Treeview tvTemp.	• Bepalen of er op content- of contentitem-element is geklikt.	• dgC(I) binden aan betreffende node uit XML-Document.
lbAddC_Click	Klikken op de knop 'Content toevoegen'.	• Nieuwe content-node maken op basis van naam in tbNew en toevoegen aan.	• Content-element toevoegen aan XML-document. • tvTemp wordt gebonden aan XML-document.
lbAddCI_Click	Klikken op de knop 'Content-item toevoegen'	• Nieuwe contentitem-node maken op basis van naam in tbNew en toevoegen aan.	• Contentitem-element toevoegen aan XML-document. • tvTemp wordt gebonden aan XML-document.
lbDelete_Click	Klikken op de knop 'verwijderen'.	• Bepalen of er een content- of contentitem-node is geselecteerd.	• Betreffende node verwijderen uit XML-document.
rblType_Selected-IndexChanged	Klikken op rblType als er een content-element is geselecteerd in tvTemp.	• Afhankelijk van de keuze andere schermelementen tonen.	n.v.t.
lbUpdate_Click	Klikken op de knop 'Opslaan'.	• Huidig DataGridItem in Item-modus zetten. • Huidige waarden uitlezen van schermelementen. • XmlNode maken met attributen van huidige waarden.	• Huidige node verwijderen uit XML-document. • Nieuwe node toevoegen aan XML-document. • tvTemp wordt gebonden aan XML-document.
lbCancel_Click	Klikken op de knop 'Annuleren' in de dgC(I).	• Huidig DataGridItem in Item-modus zetten.	• tvTemp wordt gebonden aan XML-document.
lbEdit_Click	Klikken op de knop 'bewerken'.	• Huidig DataGrid-Item in Edititem-modus zetten.	• tvTemp wordt gebonden aan XML-document.
rblView_Selected-IndexChanged	Klikken op de radiobuttonlist rblView.	• Afhankelijk van keuze datagrid en treeview of textbox met xml –bron tonen.	• tvTemp of Textbox wordt gebonden aan XML-document.

Figuur 5: Events veroorzaakt door acties op schermelementen

Pilotontwikkelplan pilot 3: CSS-editor

Afstudeeropdracht

Student:	Dhr. M. Verheul
Examinatoren:	Dhr. J.J. van Beijnum
	Dhr. W. van Vliet
afstudeerbedrijf:	Marbel Software B.V. te Noordwijkerhout
bedrijfsmentor:	Dhr. Ing. G. Scheeres
datum:	10 juni 2005
afstudeerrichting:	Informatica en Informatiekunde: VIA
afstudeerblok	2005-1.1
versie:	1.0

Inhoudsopgave

1. Inleiding

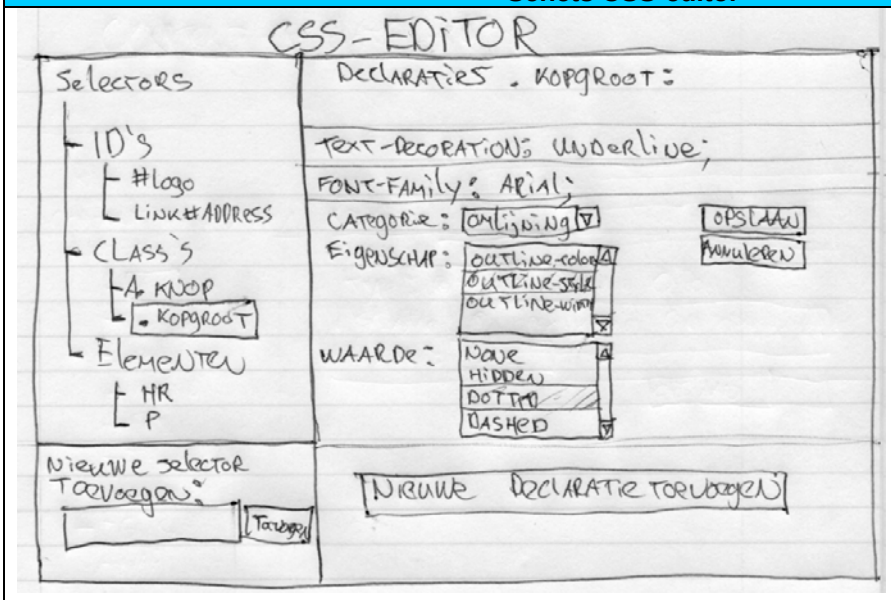
Dit document is een product van de tweede fase binnen de ontwikkelmethodiek IAD, de pilotontwikkeling. Het is geschreven in het kader van het afstudeerproject bij Marbel software. Dit pilotontwikkelplan gaat over de tweede pilot die ontwikkeld gaat worden, de content-template-editor. In hoofdstuk 2 wordt de functionele structuur van de pilot beschreven. U leest hier onder andere welke schermen er ontwikkeld gaan worden, wat de functionaliteit per scherm is en wat de mogelijke acties zijn. Tot slot wordt in hoofdstuk 3 de technische structuur van de pilot beschreven.

2. Globaal functionele structuur pilot

In dit hoofdstuk wordt de functionele structuur van de pilot CSS-editor beschreven. In de definitiestudie is in de systeemeisen al globaal beschreven aan welke eisen de CSS-editor moet voldoen. Deze zullen in dit hoofdstuk worden uitgewerkt.

2.1. Schermontwerpen

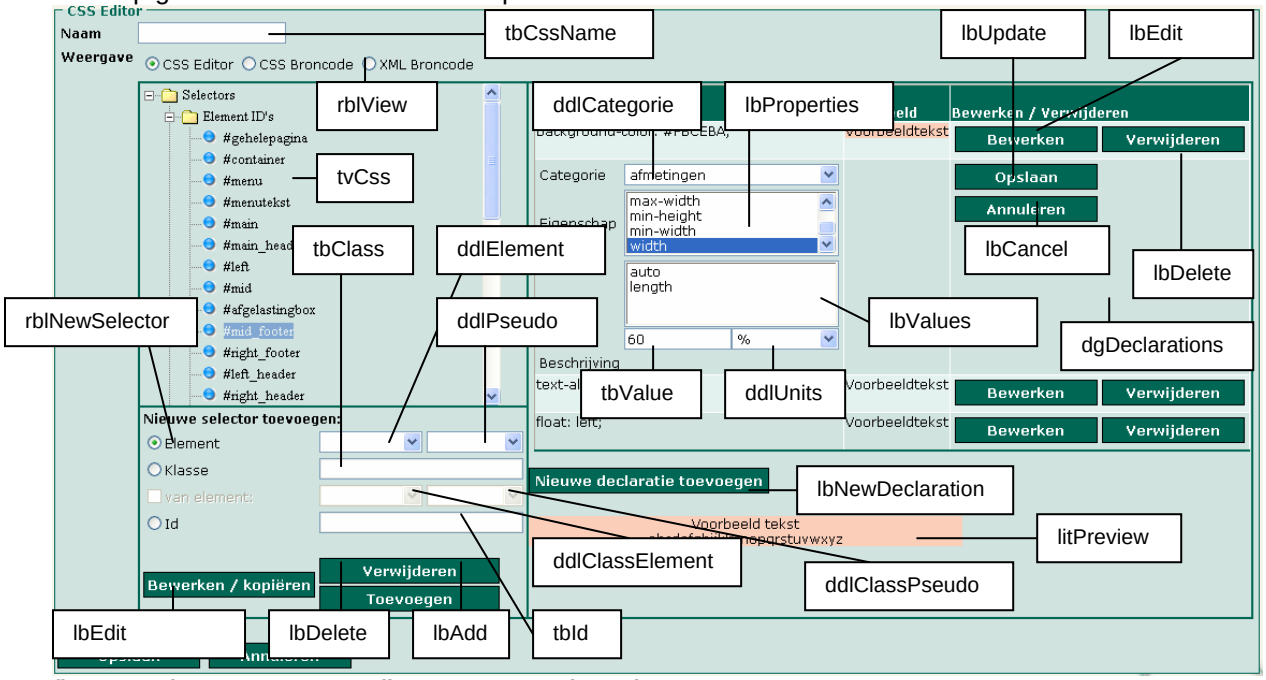
De volgende schets geeft een eerste indicatie van hoe de CSS-editor eruit gaat zien.

Schets CSS-editor	
	De linkerkant van de CSS-editor is een boom-structuur te zien waarin de verschillende ID-, Class- en elementselectors worden weergegeven. Onder de boom kunnen nieuwe selectors worden toegevoegd. Als in de boomstructuur een item is geselecteerd, verschijnt in het rechter gedeelte een datagrid waarin alle aan dat item toegekende eigenschappen worden weergegeven. Een item toevoegen of bewerken kan door middel van een dropdownlistbox waarin een categorie geselecteerd kan worden. Als dat gedaan is, verschijnen de eigenschappen van die categorie in een listbox daaronder. De waarde die voor die eigenschap ingesteld moet worden, kan worden gekozen uit nog een listbox. Indien er een getal bij die waarde gegeven moet worden, verschijnt een textbox.

Figuur 1: Schets CSS–editor

2.2. Digitaal ontwerp CSS-editor

Op basis van de schetsen van paragraaf 2.1 en door middel van prototyping is het volgende digitale ontwerp gemaakt voor de content-template-editor.



figuur 2: Schermontwerp CSS-editor met namen schermelementen

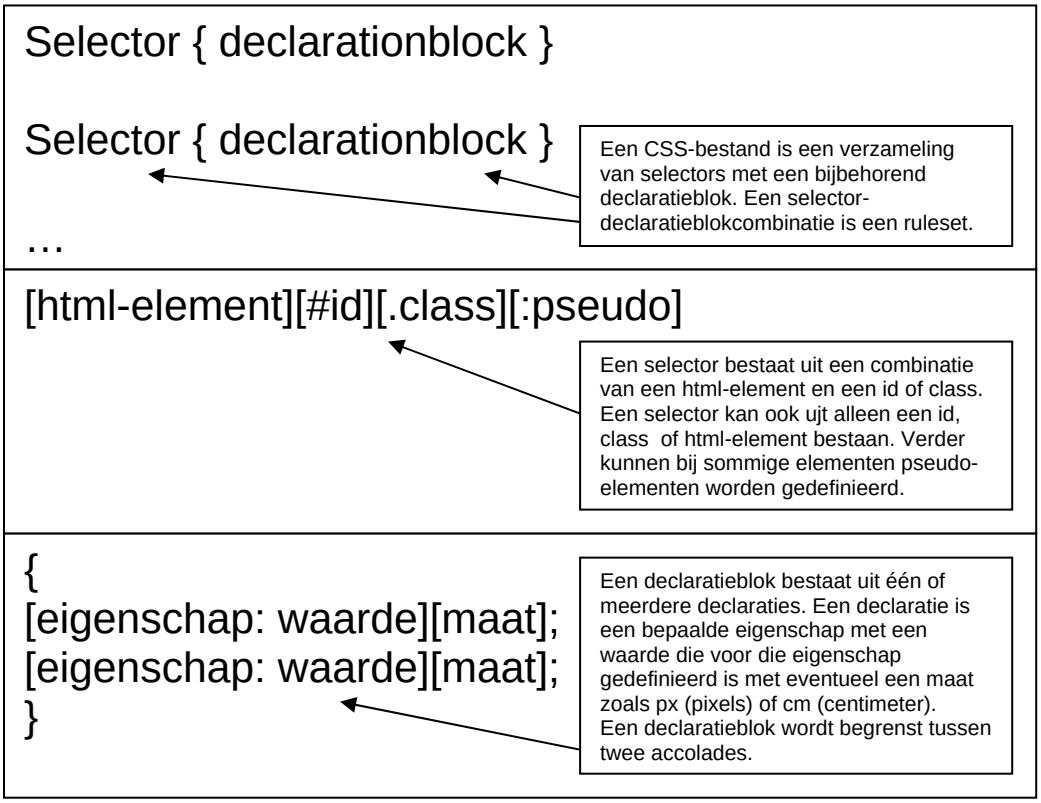
De editor moet voorzien worden van ingebouwde help-functionaliteit. Het is op dit moment nog niet bekend op welke wijze deze functionaliteit gerealiseerd gaat worden.

3. Globaal-technische structuur pilot

In dit hoofdstuk wordt de technische structuur van de derde pilot (CSS-editor) beschreven.

3.1. Structuur CSS-bestanden

CSS-documenten zijn tekstbestanden die opgebouwd zijn volgens een aantal richtlijnen. Deze richtlijnen zijn vastgelegd in de CSS-specificatie, die opgesteld is door het World Wide Web-consortium.
Het bewerken van op tekst-gebaseerde documenten is een complexe aangelegenheid, omdat er rekening gehouden moet worden met allerlei kleine afwijkingen (zoals extra spaties en 'enters'). Daarom zal voor de CSS-editor in Spider een XML-structuur ontwikkeld worden waarin de CSS-documenten worden opgeslagen. Bij het interpreteren en muteren van XML-bestanden hoeft geen rekening gehouden te worden met deze afwijkingen.
De structuur van een CSS-bestand wordt toegelicht in het volgende figuur.



figuur 3: Vereenvoudigde syntax CSS-document

De syntax zoals die gespecificeerd is in bovenstaand figuur is uitgangspunt geweest voor het opstellen van het CSS-XML-formaat. Geavanceerde css-functionaliteit zoals @-blocks (het aanroepen van een ander CSS-bestand vanuit een CSS-bestand), child-element selectors en adjacent sibling element-selectors (conditionele selector voor elementen op vergelijkbaar niveau) wordt niet meegenomen bij het opstellen van het CSS-XML-formaat.
Het volgende figuur is een voorbeeld van het XML-formaat dat ontwikkeld is voor de opslag van CSS-documenten.

```
<stylesheet>
   <rulesets>
      <ruleset>
         <selectors>
            <selector>
               <element />
               <class />
               <id>gehelepagina</id>
               <pseudo />
            </selector>
         </selectors>
         <declarationblock>
            <declaration>
               <property>width</property>
               <value>100</value>
               <unit>%</unit>
            </declaration>
            <declaration>
               <property>height</property>
               <value>30</value>
               <unit>%</unit>
            </declaration>
            <declaration>
               <property>overflow</property>
               <value>auto</value>
               <unit />
            </declaration>
         </declarationblock>
      </ruleset>
      <ruleset>
         <selectors>
            <selector>
               <element />
               <class />
               <id>container</id>
               <pseudo />
            </selector>
         </selectors>
         <declarationblock>
            <declaration>
               <property>width</property>
               <value>100</value>
               <unit>%</unit>
            </declaration>
         </declarationblock>
      </ruleset>
   </rulesets>
</stylesheet>
```

figuur 4: Voorbeeld van CSS-XML-bestand

3.2. XML -> CSS Transformatie

Omdat een webbrowser een CSS-document in tekst-formaat toegestuurd dient te krijgen, moet er een conversieslag plaatsvinden bij het opvragen van een CSS-document. Deze conversieslag gaat plaatsvinden binnen de pagina ‘css.aspx’. Deze pagina zorgt er momenteel voor dat een CSS-document uit de database wordt opgehaald en als tekst wordt doorgestuurd naar de browser.
Een extra taak voor deze pagina wordt het converteren van een pagina in het CSS-XML-formaat naar CSS. Voor deze conversieslag gaat XSLT gebruikt worden. XSLT is een techniek die speciaal ontwikkeld is om XML te transformeren naar een andere XML-indeling, HTML of platte tekst.
Het volgende figuur is het XSLT-document dat een CSS-XML-document kan transformeren naar CSS.

```
<?xml version="1.0"?>
   <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
   <xsl:output method="text" indent="no"/>

   <xsl:template match="rulesets">
      <xsl:apply-templates select="ruleset"/>
   </xsl:template>

   <xsl:template match="selector">
      <xsl:if test="element != ''"><xsl:value-of select="element"/></xsl:if>
      <xsl:if test="class != ''">.<xsl:value-of select="class"/></xsl:if>
      <xsl:if test="id != ''">#<xsl:value-of select="id"/></xsl:if>
      <xsl:if test="pseudo != ''">:<xsl:value-of select="pseudo"/></xsl:if>
      <xsl:if test="position() &lt; last()">, </xsl:if>
   </xsl:template>

   <xsl:template match="declarationblock"> {<xsl:text>
      </xsl:text>
      <xsl:apply-templates select="declaration"/><xsl:text>}
      </xsl:text>
   </xsl:template>

   <xsl:template match="declaration">
      <xsl:if test="property"><xsl:value-of select="property"/></xsl:if>
      <xsl:if test="value"><xsl:if test="unit">: <xsl:value-of
      select="value"/><xsl:value-of select="unit"/>;<xsl:text>
      </xsl:text>
      </xsl:if></xsl:if>
      <xsl:if test="value"><xsl:if test="boolean(unit) = false">: <xsl:value-of
      select="value"/>;<xsl:text>
      </xsl:text>
      </xsl:if></xsl:if>
   </xsl:template>
</xsl:stylesheet>
```

Als deze attributen niet leeg zijn, geef de waardes dan weer met het corresponderende symbool ervoor.

Als het niet de laatste selector is, een comma weergeven.

Een declaratieblok begint en eindigt met een accolade.

daartussen de declaration-template toepassen.

Declaratieregels opbouwen, indien unit is opgegeven, weergeven.

figuur 5: XSLT-document voor transformatie XML naar CSS

Wanneer het XSLT-document uit bovenstaand figuur gebruikt wordt om het XML-document uit figuur 4 te transformeren, is het CSS-document in het volgende figuur het resultaat.

```
#gehelepagina {
width: 100%;
height: 30%;
overflow: auto;
}

#container {
width: 100%;
}
```

figuur 6: Resultaat XSLT-Transformatie

3.3. CSS-definitie

Alle mogelijke CSS-eigenschappen met bijbehorende waarden worden vastgelegd in een apart XML-bestand. Het volgende figuur brengt de structuur van dat XML-bestand in beeld.

```
<?xml version="1.0" encoding="UTF-8"?>
<csseditor>
 <categorie naam="achtergrond">
  <property naam="background-color" description="Stelt de achtergrondkleur van een
  element in.">
   <possible type="value" naam="color" />
   <possible type="multi" naam="transparant" />
  </property>
  <property naam="background-image" description="Stelt de achtergrondafbeelding in">
   <possible type="value" naam="url" />
   <possible type="multi" naam="none" />
  </property>
  ...
 </categorie>
 ...
</csseditdor>
```

figuur 7: Structuur XML-bestand met CSS-eigenschappen en categorieën

Dit bestand is op opgesteld op basis van de CSS-reference van de website www.w3schools.com. Het bestand zal gebruikt worden om de schermelementen in de CSS-editor dynamisch van waarden te voorzien.

3.4. Beschrijving events

De onderstaande tabel beschrijft de acties die geprogrammeerd moeten worden om de events veroorzaakt door de schermelementen af te handelen.

Events voor CSS-editor			
Event	**User action**	**System response**	**Database response**
Page_Load		• De pagina wordt geladen en de scherm-elementen geïnitialiseerd.	• CSSService haalt CSS-XML-document op. • tvCss wordt gebonden aan XML-document.
tvCss_Selected IndexChange	De gebruiker selecteert een selector in boomstructuur.	• Bepalen wat voor soort	• dgDeclarations wordt gebonden aan de declarations-XMLNode, horend bij de in tvCSS geselecteerde selector.
rblNewSelector_ SelectedIndexChange	De gebruiker klikt op één van de radiobuttons bij ‘Nieuwe selector toevoegen’.	• Afhankelijk van de keuze moeten er schermelementen enabled/disabled worden.	
rblView_ SelectedIndexChange	De gebruiker klikt op één van de radiobuttons bij ‘Weergave’.	• Afhankelijk van de keuze verandert de weergave in ‘CSS editor’, ‘CSS Broncode’ of ‘XML Broncode’.	• Afhankelijk van de inhoud wordt dgDeclarations gebonden aan datagrid
lbAdd_Click	De gebruiker klikt op de knop ‘toevoegen’ bij ‘nieuwe selector toevoegen’.	• Afhankelijk van de selctie in rblNewSelector wordt een selector-node gemaakt met element-, id- en/of class-element.	• Nieuwe selector-node wordt toegevoegd aan XmlDocument. • tvCss wordt gebonden aan XmlDocument.
lbDelete_Click	De gebruiker klikt op de knop ‘Verwijderen’.		De Xml-node horend bij de in tvCss geselecteerde selector wordt verwijderd uit het XmlDocument.
lbEdit_Click	De gebruiker klikt op de knop ‘bewerken’.	• Er verschijnen knoppen waarmee de naam van de in tvCss geselecteerde selector gewijzigd kan worden.	
lbNewDeclaration	De gebruiker klikt op de knop ‘Nieuwe declaratie toevoegen’.	• EditItemIndex van dgDeclarations wordt op ItemIndex van de nieuw toegevoegde declaration-node gezet.	• Er wordt een nieuwe dgDeclaration-XmlNode toegevoegd aan het XmlDocument. • dgDeclarations wordt opnieuw gebonden aan het XmlDocument.
lbEdit_Click (dgDeclarations child-control)	De gebruiker klikt op de knop ‘bewerken’ in een regel van dgDeclarations.	• EditItemIndex van dgDeclarations wordt op de huidige ItemIndex gezet.	• dgDeclarations wordt gebonden aan XmlDocument.
lbDelete_Click (dgDeclarations child-control)	De gebruiker klikt op de knop ‘Verwijderen’ in een regel van	• EditItemIndex van dgDeclarations wordt op -1 gezet.	• Huidige declaration-node wordt verwijderd van XmlDocument.

Events voor CSS-editor			
	dgDeclarations.		• dgDeclarations wordt gebonden aan XmlDocument.
lbUpdate_Click (dgDeclarations child-control)	De gebruiker klikt op de knop 'Update' in een regel van dgDeclarations.	• EditItemIndex van dgDeclarations wordt op -1 gezet. • Declaration-XmlNode wordt gemaakt en gevuld met de bijgewerkte waarden van huidige regel.	• Nieuwe declaration-node wordt toegevoegd aan XmlDocument. • dgDeclarations wordt gebonden aan XmlDocument.
lbCancel_Click (dgDeclarations child-control)	De gebruiker klikt op de knop 'Annuleren' in een regel van dgDeclarations.	• EditItemIndex van dgDeclarations wordt op -1 gezet.	• dgDeclarations wordt gebonden aan XmlDocument.
ddlCategorie_ SelectedIndexChange (dgDeclarations child-control)	De gebruiker selecteert een item in ddlCategories.	• Afhankelijk van de keuze wordt lbProperties gevuld met eigenschappen.	
lbProperties_ SelectedIndexChange (dgDeclarations child-control)	De gebruiker selecteert een item in lbProperties.	• Afhankelijk van de keuze wordt lbValues gevuld met waarden.	
lbValues_ SelectedIndexChange (dgDeclarations child-control)	De gebruiker selecteert een item in lbValues.	• Afhankelijk van de keuze wordt tbValue en ddlUnits laten zien.	
Opslaan_Click	De gebruiker klikt op de knop 'Opslaan' onder in het scherm.	• De gebruiker wordt doorgestuurd naar de menu-pagina.	• Het XmlDocument uit de session wordt opgehaald en naar de CSSService gestuurd om opgeslagen te worden.
Annuleren_Click	De gebruiker klikt op de knop 'Annuleren' onder in het scherm.	• De gebruiker wordt doorgestuurd naar de menu-pagina.	

Figuur 8: Events veroorzaakt door acties op schermelementen

Inhoudsopgave

Vereenvoudiging programmacode ITemplate-klasse bij documenteditor

```
public class DataGridContentTemplate : ITemplate
 {
//Globale variabelen declararen..
  ListItemType templateType;
  string strToken;
//Constructorcode: code wordt uitgevoerd bij het
//maken van een nieuw object van deze klasse.
  public DataGridContentTemplate(ListItemType type, string token)
  {
//Globale vairabelen vullen..
   templateType = type;
   strToken = token;
  }

  public void InstantiateIn(System.Web.UI.Control container)
  {
//Deze code wordt uitgevoerd als de templatekolom wordt aangeroepen..
   Literal l = new Literal();

//Afhankelijk van het soort templatekolom, de kolom vullen met de benodigde controls
   switch(templateType)
   {
    case ListItemType.Header:
     l.Text = "content eigenschappen";
     container.Controls.Add(l);
     break;

    case ListItemType.Item:
//In de Item-weergave hoeft alleen maar de waarde worden laten zien. In dit voorbeeld: alleen
de categorie.
     Label lCat = new Label();
     lCat.DataBinding += new EventHandler(lCat_DataBinding);
     container.Controls.Add(lCat);
     break;

    case ListItemType.EditItem:
//In de EditItem-weergave moeten ook controls geboden worden om de waarden te muteren.
//Plaats deze controls (2x listbox en 2x linkbutton) op het scherm, koppel ze
// aan de cssstyle 'button', Koppel de Click-events van de knoppen en de
//databinding-events van de listboxen aan een eventHandler.
     ListBox lbCat = new ListBox();
     ListBox lbAllCat = new ListBox();
     LinkButton bCatAdd = new LinkButton();
     bCatAdd.Text = "<--";
     bCatAdd.Width = Unit.Percentage(100);
     bCatAdd.CssClass = "button";
     LinkButton bCatRemove = new LinkButton();
     bCatRemove.Text = "-->";
     bCatRemove.Width = Unit.Percentage(100);
     bCatRemove.CssClass = "button";
     lbAllCat.DataBinding += new EventHandler(lbAllCat_DataBinding);
     lbAllCat.Width = Unit.Percentage(100);
     lbAllCat.ID = "lbAllCat";
     lbCat.DataBinding += new EventHandler(lbCat_DataBinding);
     lbCat.Width = Unit.Percentage(100);
     lbCat.ID = "lbCat";
     bCatAdd.Click += new EventHandler(bCatAdd_Click);
     bCatRemove.Click += new EventHandler(bCatRemove_Click);
     container.Controls.Add(lbCat);
     container.Controls.Add(bCatAdd);
     container.Controls.Add(bCatRemove);
     container.Controls.Add(lbAllCat);
     break;
   }
  }

  private void lCat_DataBinding(object sender, System.EventArgs e)
  {//Voor normale weergave een label gebruiken. Variabelen initialiseren.
   Label lCat = (Label) sender;
   DataGridItem container = (DataGridItem) lCat.NamingContainer;
   string strWaarden;
   //De DataItem-eigenschap typecasten als XmlNode…
```

```
    if (((System.Xml.XmlNode)container.DataItem).Attributes["category"] == null)
    {
     strWaarden = "";
    }
    else
    {//Waarden-string in label laten zien.
     strWaarden = ((System.Xml.XmlNode)container.DataItem).Attributes["category"].Value;
    }
    lCat.Text = strWaarden;
   }

   private void lbCat_DataBinding(object sender, System.EventArgs e)
   {//Voor edititem-weergave moet onder andere deze listbox gebruikt worden.
    ListBox lbCat =  (ListBox) sender;
    DataGridItem container = (DataGridItem) lbCat.NamingContainer;
    string strWaarden;
    //DataItem-eigenschap op dezelfde manier typecasten als hierboven.
    if (((System.Xml.XmlNode)container.DataItem).Attributes["category"] == null)
    {
     strWaarden = "";
    }
    else
    {
     strWaarden = ((System.Xml.XmlNode)container.DataItem).Attributes["category"].Value;
    }
 //Als er waarden in de string zitten..
    if (strWaarden.Length > 0)
    {
 //De string splitsen waar comma's zitten en alle items doorlopen.
     string[] ArrayWaarden = strWaarden.Split(",".ToCharArray());
     lbCat.Items.Clear();
     foreach (string waarde in ArrayWaarden)
     {
     //Voor elk item een ListItem aanmaken en deze toevoegen aan de ListBox
      ListItem liNew = new ListItem(waarde, waarde);
      lbCat.Items.Add(liNew);
     }
    }
    else
    {//Als er geen waarden in de string zitten, listbox legen voor de zekerheid.
     lbCat.Items.Clear();
    }
   }

   private void bCatAdd_Click(object sender, System.EventArgs e)
   {//Als er op de knop 'toevoegen' is gedrukt.
    //Als er een categorie in de Listbox lbAllCat is geselecteerd..
    if (lbAllCat.SelectedIndex != -1)
    {//Een Nieuw ListItem aanmaken met de geselecteerde waarde.
     ListItem liNew = new ListItem(lbAllCat.SelectedItem.Text,
     lbAllCat.SelectedItem.Value);
     //Als de listbox het item nog niet bevat..,
     if (lbCat.Items.Contains(liNew) != true)
     {//Het item toevoegen.
      lbCat.Items.Add(liNew);
     }
    }
    //DataGridItem maken voor de container van deze knop.
    DataGridItem container = (DataGridItem) bCatAdd.NamingContainer;
 //ListBox met subcategorien definiëren..
    ListBox lbAllSubCat = (ListBox) container.FindControl("lbAllSubCat");
    //Nieuwe instantie van Category WebService..
    CategoryService _CS = new CategoryService();
    lbAllSubCat.Items.Clear();
    //Doorloop alle geselecteerde categoriën..
    foreach (ListItem liDo in lbCat.Items)
    {//En maak voor elke categorie een dataset aan met daarin (via de categoryservice)
     //de subcategoriën die bij die categorie horen..
     DataSet dsSubCats = _CS.GetAllSelectedSubCategory(liDo.Value.Trim(), strToken);
     //Doorloop daarna alle datarows van die dataset..
     foreach (DataRow drSubCat in dsSubCats.Tables[0].Rows)
     {
     //..en maak een ListItem aan met de naam van de subcategoriën.
      ListItem liNew = new ListItem(drSubCat.ItemArray[1].ToString(),
```

```
    drSubCat.ItemArray[1].ToString());
    //en voeg die als die nog niet voor kwam in de ListBox..
    if (!lbAllSubCat.Items.Contains(liNew))
    {
     //toe!
     lbAllSubCat.Items.Add(liNew);
    }
   }
  }
 }
```

Programmacode voor het vullen van de treeview bij de template-editor

```
 private void BuildTree()
 {
  //Eerst de TreeView legen..
  tvTemp.Nodes.Clear();
  //Haal XmlDocument op uit de sessie..
  _XD = (XmlDocument) Session["te.tempDoc"];
  //Rootnode maken waar de hele boom aan komt te hangen..
  TreeNode tnRoot = new TreeNode();
  //Tekst invullen..
  tnRoot.Text = "Content template";
  //Root-node toevoegen..
  tvTemp.Nodes.Add(tnRoot);
  //Doorloop alle content-elementen in het XML-bestand:
  foreach (XmlNode xnContent in _XD.SelectNodes("contentcontainer/contentcollection/content"))
  {
   //Maak voor elk content-element een treenode aan.
   TreeNode tnContent = new TreeNode();
   //Geef de treenode de naam van het content-element.
   tnContent.Text = xnContent.Attributes["name"].Value;

   //Doorloop alle contentitem-elementen die horen bij dit content-element.
   foreach (XmlNode xnContentItem in
xnContent.SelectNodes("contentitemcollection/contentitem"))
   {
    //Maak een nieuwe contentitem-treenode aan.
    TreeNode tnContentItem = new TreeNode();
    //Wanneer het contentitem-element van het type tekst is..
    if (xnContentItem.Attributes["type"].Value == "tekst")
    {
     //Gebruik dan het volgende pictogram..
     tnContentItem.ImageUrl = Server.MapPath("images/text.gif");
    }
    //Wanneer het contentitem-element van het type image is..
    if (xnContentItem.Attributes["type"].Value == "image")
    {
     //Gebruik dan het volgende pictogram..
     tnContentItem.ImageUrl = Server.MapPath("images/image.gif");
    }
    //Wanneer het contentitem-element van het type poll is..
    if (xnContentItem.Attributes["type"].Value == "poll")
    {
     //Gebruik dan het volgende pictogram..
     tnContentItem.ImageUrl = Server.MapPath("images/vote.gif");
    }
    //Geef de contentitem-treenode de naam van het bijbehorende XML-element.
    tnContentItem.Text = xnContentItem.Attributes["name"].Value;
    //Voeg de contentitem-treenode toe aan de content-treenode..
    tnContent.Nodes.Add(tnContentItem);
   }
   //en voeg de contenttreenode toe aan de root treenode.
   tnRoot.Nodes.Add(tnContent);
  }
  //klap de boom helemaal uit.
  tvTemp.ExpandLevel = 3;
 }
```

Programmaroutine XML naar CSS

```
public string getcss(XmlDocument xdCss, string xsltlocation)
  { //Nieuwe XSL-T transform aanmaken...
   XslTransform xslt = new XslTransform();
   //XSL-T laden.
   xslt.Load(xsltlocation);
   string output;
   StringWriter writer = new StringWriter();
   //Transform uitvoeren naar stringwriter...
   xslt.Transform(xdCss, null, writer, null);
   //Stringwriter wegschrijven naar string (uitvoer van de functie)
   output = writer.ToString();
   return output;
  }
```

Programmaroutine CSS naar XML

```
  public XmlDocument getxd(string cssstring, string cssdeflocation)
  {
   //Allereerst de CSS-string ontdoen van commentaar d.m.v. een regular expression.
   cssstring = System.Text.RegularExpressions.Regex.
   Replace(cssstring, "/\\*.*?\\*/", "");
   //XML-string klaarmaken.
   string strXml = "";
   //XML-document klaarzetten met CSS-specificatie. Wordt later gebruikt.
   XmlDocument xdCssDef = new XmlDocument();
   xdCssDef.Load(cssdeflocation);
   XmlNodeList xnlUnits = xdCssDef.SelectNodes("csseditor/cssunits/measurements/unit");
   //XML-string begint met stylesheet- en rulesets-element.
   strXml += "<stylesheet>";
   strXml += "<rulesets>";
   //stringarray met rulesets maken door CSS-string te splitsen bij elke '}'.
   string[] rulesets = cssstring.Trim().Split("}".ToCharArray());
   //Elk onderdeel van die stringarray loopen...
   foreach (string ruleset in rulesets)
   {
    if (ruleset.IndexOf("{") > 0) //Alleen de juiste onderdelen loopen.
    {
     strXml += "<ruleset>";
     //Het selectorgedeelte ontdoen van de rechter accolade...
     string selectors = ruleset.Substring(0, ruleset.IndexOf("{"));
     strXml += "<selectors>";
     //Loop maken tussen voor alle mogelijke selectors in de selector-string.
     foreach (string selector in selectors.Trim().Split(",".ToCharArray()))
     {
      //variabelen klaarzetten voor de onderdelen van de selector.
      string _element = "";
      string _class = "";
      string _id = "";
      string _pseudo = "";
      strXml += "<selector>";
      //standaard zit er geen pseudo-element in..
      bool gotpseudo = false;
      //De volgende code kan tot een exceptie leiden...
      //Niets doen met die exceptie..
      try
      {
       if (selector.IndexOf(":") > 0) //Als er een dubbele punt in de string zit...
       {
        //...zet hem dan apart...
        _pseudo = selector.Substring(selector.IndexOf(":") + 1);
        // en onthoud dat ie erin zit.
        gotpseudo = true;
       }
      }
      catch{}
      //Als er GEEN pseudo-element in zit:
      if (!gotpseudo)
      {   // en er zit een punt in de selectorstring
       if (selector.IndexOf(".") > 0)
       { //zet de elementen dan in hun eigen variabele..
        _element = selector.Substring(0, selector.IndexOf("."));
        _class = selector.Substring(selector.IndexOf(".") + 1);
        _id = "";
```

```
 }
 //begint de selector met een punt?
 if (selector.IndexOf(".") == 0)
 { //Dan is er geen element.
  _element = "";
  _class = selector.Substring(selector.IndexOf(".") + 1);
  _id = "";
 }// Zit er een hekje in de string?
 if (selector.IndexOf("#") > 0)
 {   //Dan is er een element en een id die gedefinieerd moet worden.
  _element = selector.Substring(0, selector.IndexOf("#"));
  _id = selector.Substring(selector.IndexOf("#") + 1);
  _class = "";
 }// Begint de string met dat hekje?
 if (selector.IndexOf("#") == 0)
 { //Dan is er geen element...
  _element = "";
  _id = selector.Substring(selector.IndexOf("#") + 1);
  _class = "";
 }//Zit er geen punt en geen hekje in de string?
 if ((selector.IndexOf(".") < 0)&&(selector.IndexOf("#") < 0))
 {   //Dan is de selector een en al element.
  _element = selector;
  _id = "";
  _class = "";
 }
}
// Als er WEL een pseudo is...
else
{   // Dan moet ie eraf worden gehaald bij het splitsen.
 if (selector.IndexOf(".") > 0)
 {
  _element = selector.Substring(0, selector.IndexOf("."));
  _class = selector.Substring(selector.IndexOf(".") + 1, (selector.IndexOf(":") –
  selector.IndexOf("."))-1);
  _id = "";
 }
 //en hier..
 if (selector.IndexOf(".") == 0)
 {
  _element = "";
  _class = selector.Substring(selector.IndexOf(".") + 1, (selector.IndexOf(":") –
  selector.IndexOf("."))-1);
  _id = "";
 }//en hier...
 if (selector.IndexOf("#") > 0)
 {
  _element = selector.Substring(0, selector.IndexOf("#"));
  _id = selector.Substring(selector.IndexOf("#") + 1, (selector.IndexOf(":") –
  selector.IndexOf("."))-1);
  _class = "";
 }//en hier...
 if (selector.IndexOf("#") == 0)
 {
  _element = "";
  _id = selector.Substring(selector.IndexOf("#") + 1, (selector.IndexOf(":") –
  selector.IndexOf("."))-1);
  _class = "";
 }// en hier...
 if ((selector.IndexOf(".") < 0)&&(selector.IndexOf("#") < 0))
 {
  _element = selector.Substring(0, selector.IndexOf(":"));
  _id = "";
  _class = "";
 }
}
// Nu alle variabelen juist gevuld zijn, kan de XML-string verder worden opgebouwd.
strXml += "<element>";
strXml += _element;
strXml += "</element>";
strXml += "<class>";
strXml += _class;
strXml += "</class>";
strXml += "<id>";
```

```
      strXml += _id;
      strXml += "</id>";
      strXml += "<pseudo>";
      strXml += _pseudo;
      strXml += "</pseudo>";
      strXml += "</selector>";
     }
     strXml += "</selectors>";
     //Elk selectors-geeelte heeft een declarationblock.
     //Dat is te vinden aan de rechterkant van de linker accolade...
     string declarations = ruleset.Substring(ruleset.IndexOf("{") + 1);
     strXml += "<declarationblock>";
     //Maak een loop voor elke declaratie.
     foreach (string declaration in declarations.Split(";".ToCharArray()))
     {
      if (declaration.IndexOf(":") > 0)
      {
       strXml += "<declaration>";
       //Elke declaratie bestaat uit een property
       //(alles aan de linkerkant van de dubbele punt afgezien van spaties)
       string _property = declaration.Substring(0, declaration.IndexOf(":")).Trim();
       strXml += "<property>";
       strXml += _property;
       strXml += "</property>";
       strXml += "<value>";
       //en een value (alles aan de rechterkant van de : aftgezien van spaties)
       string _value = declaration.Substring(declaration.IndexOf(":") + 1).Trim();
       //en een unit (meeteenheid)
       string _unit = "";
       //Die meeteenheid moet herkend worden..
       //Gebruik daarvoor de eerder klaargezette XmlNodeList
       foreach (XmlNode xnUnit in xnlUnits)
       {
        if ((_value.Length > 0)&&(_value.IndexOf(xnUnit.Attributes["naam"].Value) ==
        (_value.Length - xnUnit.Attributes["naam"].Value.Length)))
        {
         try
          //Als de waarde gedefinieerd is en die eindigd niet
          //op een van de verschillende units...
         { //Dan levert de volgende code geen exceptie op...
          _value = Int32.Parse(_value).ToString();
         }
         catch
         {
          //Anders is de waarde niet alleen een getal
          //en moet de unit apart worden opgeslagen.
          _unit = _value.Substring(_value.IndexOf(xnUnit.Attributes["naam"].Value));
          _value = _value.Substring(0, _value.IndexOf(xnUnit.Attributes["naam"].Value));
         }
        }
       }
       //Variabelen aan string toevoegen en elementen sluiten...
       strXml += _value;
       strXml += "</value>";
       strXml += "<unit>";
       strXml += _unit;
       strXml += "</unit>";
       strXml += "</declaration>";
      }
     }
     strXml += "</declarationblock>";
     strXml += "</ruleset>";
    }
   }
   strXml += "</rulesets>";
   strXml += "</stylesheet>";
   XmlDocument xdCss = new XmlDocument();
   //Geen XML-string teruggeven maar een XmlDocument.
   xdCss.LoadXml(strXml);
   return xdCss;
  }
```

Toelichting

Het doel van dit document is het vastleggen van de uitkomsten van de functionele test van vernieuwde delen van het systeem 'Spider'. Deze delen zijn ontwikkeld in het kader van de afstudeeropdracht 'Verbeteren van de usability van Spider-CMS' uitgevoerd door Michiel Verheul. De tests zijn uitgevoerd door bij iedere editor steeds een document aan te maken waarbij alle functionaliteit van de editor gebruikt werd.
Verder moet via dit document ook formeel het acceptatieoordeel van de opdrachtgever worden vastgelegd. Dit acceptatieoordeel vindt plaats op basis van eisen die gesteld zijn in het onderdeel 'ontwikkelscenario' van het rapport 'definitiestudie'.
Ook de uitkomsten van de functionele test wordt hierbij meegenomen. In het geval van een positief acceptatieoordeel kunnen de delen van het systeem worden geïntegreerd in een nieuwe release. Deze release kan dan worden ingevoerd bij de verschillende afnemers van het systeem.
Wanneer er over delen van het systeem geen positief acceptatieoordeel kan worden gegeven, dienen er afspraken gemaakt te worden die alsnog tot acceptatie van deze delen moeten leiden.

Basissysteemeisen

Nr.	BS.01
Omschrijving	Gebruikers (gebruikersgroep site-beheerders) moeten wijzingen kunnen aanbrengen in de opmaak van een website zonder dat ze kennis hoeven te hebben van CSS.
Systeem voldoet aan eis	Ja, voor het toevoegen en aanpassen van opmaakeigenschappen hoeven gebruikers geen CSS-eigenschappen te kennen.
Opmerkingen	De gebruikers dienen wel op de hoogte te zijn van de mogelijkheden van CSS en de wijze waarop ze die kunnen toepassen op de website. Dit wordt uitgelegd in het helpgedeelte van de CSS-editor.
Correctie nodig	-

Nr.	BS.02
Omschrijving	Gebruikers (gebruikersgroep site-beheerders) moeten nieuwe pagina's kunnen toevoegen aan een website zonder dat ze kennis hoeven te hebben van XML dat gebruikt wordt binnen Spider om pagina's te definiëren.
Systeem voldoet aan eis	Ja, er hoeft geen XML gebruikt te worden om nieuwe pagina's te definiëren bestaande pagina's aan te passen.
Opmerkingen	Gebruikers dienen wel op de hoogte te zijn van de mogelijkheden van documenten binnen Spider. Deze worden besproken het helpgedeelte van de documenteditor.
Correctie nodig	-

Nr.	BS.03
Omschrijving	Gebruikers (gebruikersgroep site-beheerders) moeten content-templates kunnen toevoegen aan een website zonder dat ze kennis te hoeven hebben van XML dat gebruikt wordt om nieuwe templates te definiëren binnen Spider.
Systeem voldoet aan eis	Ja, er hoeft geen XML gebruikt te worden om nieuwe content-templates te definiëren of bestaande content-templates aan te passen.
Opmerkingen	De gebruiker dient wel op de hoogte te zijn van de algehele werking van Spider en de structuur van content. Dit wordt uitgelegd in het ingebouwde helpgedeelte.
Correctie nodig	-

Interface-eisen

Nr.	IF.01
Omschrijving	De gebruikersinterfaces moeten goed worden weergegeven in Microsoft Internet Explorer 5.5 of hoger.
Systeem voldoet aan eis	Ja, de pagina's worden correct weergegeven.
Opmerkingen	-
Correctie nodig	-

Nr.	IF.02
Omschrijving	De gebruikersinterfaces moeten aansluiten bij de huidige vormgeving van Spider.
Systeem voldoet aan eis	Ja, de ontwikkelde gebruikersinterfaces maken gebruik van dezelfde stylesheet als die van de andere interfaces binnen Spider.
Opmerkingen	-
Correctie nodig	Bij elk van de ontwikkelde pilots (en bij andere onderdelen binnen Spider) dient er verticaal gescrold te worden als de inhoud niet meer op het scherm past. In de nabije toekomst zullen de buitenste schuifbalken worden weggewerkt zodat alle ook de 'opslaan'- en 'annuleren'-knoppen op het scherm passen.

Nr.	IF.03
Omschrijving	De gebruikersinterfaces moeten te gebruiken zijn zonder dat er aanvullende training voor nodig is.
Systeem voldoet aan eis	Ja
Opmerkingen	-
Correctie nodig	-

Nr.	IF.04
Omschrijving	De gebruikersinterfaces moeten worden uitgerust met ingebouwde helpfunctionaliteit.
Systeem voldoet aan eis	Ja
Opmerkingen	-
Correctie nodig	-

Nr.	IF.05
Omschrijving	Ervaren gebruikers moeten de mogelijkheid hebben om de gebruikersinterfaces uit te kunnen schakelen.
Systeem voldoet aan eis	Ja, de gebruikersinterfaces zijn uit te schakelen.
Opmerkingen	Het zou fijn zijn als gebruikers de mogelijkheid hebben om deze instelling op te slaan in het gebruikersprofiel.
Correctie nodig	Bovenstaande functionaliteit meenemen in een volgende release.

Nr.	IF.06
Omschrijving	De gebruikersinterfaces moeten een correcte invoer van de gebruikers afdwingen.
Systeem voldoet aan eis	Ja, invoer van verplichte velden wordt afgedwongen, foutieve invoer wordt voorkomen door middel van validatie.
Opmerkingen	-
Correctie nodig	-

Integriteitseisen

Nr.	IG.01
Omschrijving	De gegevens die worden ingebracht in Spider dienen centraal te worden opgeslagen zodat ze op elk moment geraadpleegd kunnen worden en gemakkelijk kunnen worden gebackupt.
Systeem voldoet aan eis	Ja, dit wordt echter standaard geregeld binnen Spider.
Opmerkingen	-
Correctie nodig	-

Nr.	IG.02
Omschrijving	Er moeten meerdere versies van de ingebrachte gegevens bewaard worden, zodat in geval van een vergissing door de gebruiker, een vorige versie van de gegevens kunnen worden herroepen.
Systeem voldoet aan eis	Bij de documenteditor en template-editor wordt dit wel gedaan. De CSS-editor beschikt echter nog niet over een versie-managementsysteem.
Opmerkingen	-
Correctie nodig	Bij een eventuele volgende release kan een XML-DAL ontwikkeld worden voor de opslag van CSS-bestanden.

Performance-eisen

Nr.	PF.01
Omschrijving	De responsetijd van het systeem op acties van de gebruiker dient te worden beperkt tot maximaal 1 seconde als het gaat om het bewerken van kleine hoeveelheden informatie. (tot 5000 karakters) bij een verbindingssnelheid van 100 kbps.
Systeem voldoet aan eis	Niet kunnen vaststellen.
Opmerkingen	Deze eis is niet getest omdat het systeem nog niet bij een klant is ingevoerd.
Correctie nodig	-

Nr.	PF.02
Omschrijving	De te ontwikkelen gebruikersinterfaces dienen geladen te zijn in maximaal 1 seconde bij een verbindingssnelheid van 100 kbps.
Systeem voldoet aan eis	Niet kunnen vaststellen.
Opmerkingen	Deze eis is niet getest omdat het systeem nog niet bij een klant is ingevoerd.
Correctie nodig	-

Operationele eisen

Nr.	OP.01
Omschrijving	Spider wordt ontwikkeld en beheerd door Marbel Software. De nieuwe functionaliteit die in het kader van dit project wordt ontwikkeld, zal worden meegenomen bij de invoering van een nieuwe release van Spider bij een klant.
Systeem voldoet aan eis	Ja, de ontwikkelde pilots kunnen worden geïntegreerd in de volgende release.
Opmerkingen	-
Correctie nodig	-

Nr.	OP.02
Omschrijving	De nieuwe functionaliteit die ontwikkeld gaat worden, moet beheerd kunnen worden door Marbel Software.
Systeem voldoet aan eis	Ja, de functionaliteit is ontwikkeld binnen dezelfde technische structuur als de rest van het systeem.
Opmerkingen	-
Correctie nodig	-

Nr.	OP.03
Omschrijving	Als een nieuwe versie van Spider met de in dit project ontwikkelde functionaliteit bij een klant ingevoerd gaat worden, moeten de gebruikers goed met de veranderingen overweg kunnen. Daarom zal er een uitgebreide beschrijving worden gemaakt waarin de veranderingen worden toegelicht.
Systeem voldoet aan eis	Nee, deze beschrijving is momenteel nog niet gemaakt.
Opmerkingen	-
Correctie nodig	Indien gewenst moet er een document komen waarin alle wijzigingen worden opgesomd.

Acceptatie

Tijdens de fase definitiestudie van dit project zijn bij het bepalen van het ontwikkelscenario een aantal acceptatiecriteria opgesteld. Op basis van deze criteria moet de opdrachtgever nu bepalen of hij akkoord gaat met de ontwikkelde pilots. Dit kan worden aangegeven in de onderstaande tabel.

Criterium	Pilot	Akkoord
De functionaliteit van de pilot komt overeen met de systeemeisen.	1. Documenteditor	JA / NEE
	2. Template-editor	JA / NEE
	3. CSS-editor	JA / NEE
Er is voldaan aan de integriteits- en interface-eisen.	1. Documenteditor	JA / NEE
	2. Template-editor	JA / NEE
	3. CSS-editor	JA / NEE
De systeemdocumentatie is op orde.	1. Documenteditor	JA / NEE
	2. Template-editor	JA / NEE
	3. CSS-editor	JA / NEE
Het systeem kan worden beheerd door Marbel Software.	Alle pilots	JA / NEE

Wanneer de opdrachtgever zich in voldoende mate kan vinden met de bovenstaande criteria, kan hij overgaan tot een definitief acceptatieoordeel.

Openstaande punten

- Schermen compacter maken zodat niet meer verticaal hoeft te worden gescrolld.
- Mogelijkheid om GUI's uit te schakelen integreren in gebruikersprofiel
- XML-DAL ontwikkelen voor CSS-editor
- Broncode voor CSS-editor documenteren

Definitief acceptatieoordeel opdrachtgever