Technische Universiteit Delft

Bachelor eindproject

Faculteit: Electrotechniek, Wiskunde en Informatica

Sectie: Web Information Systems

DENC Docs

Studenten:
Martijn Berger (1123076)
Michael Croes (1265180)

Begeleiders:
Bernard Sodoyer (Technische Universiteit Delft)
Sjef Ruhe (DENC)

versie 0.3
8 mei 2010
IN3405

Voorwoord

Dit werk is het eindrapport van het Bachelor eindproject dat op zijn beurt weer het sluitstuk is van de bacheloropleiding Technische Informatica aan de Technische Universiteit Delft. Ter afsluiting van de driejarige bacheloropleiding is een kwartaal gereserveerd om een zelfstandig project uit te voeren. In deze periode is het de bedoeling dat studenten zelfstandig het hele software ontwikkeltraject doorlopen van de "tekentafel", tot de daadwerkelijke implementatie, en zo gaandeweg leren de opgedane kennis in de praktijk te brengen.
Voorliggend project werd uitgevoerd bij DENC Netherlands. DENC is een multidiciplinaire dienstverlener, met name actief in de bouwsector, die zich vooral toelegt op het bieden van totaaloplossingen voor huisvestingsvraagstukken. Onze begeleider vanuit DENC was ir. J.J.M. Ruhe (Sjef). Als werktuigbouwkundig ingenieur richt hij zich op logistiek, procesverbetering en innovatie. Zo is hij onder meer betrokken geweest bij diverse projecten die tot doel hadden het optimaliseren van stromen van producten bij onder andere Schiphol, Flora Holland en Centraal Boekhuis. Daarbij viel hem op dat zelfs instellingen van dat kaliber de technische documenten, nodig voor hun projecten, moeilijk boven water kregen. Zo kwam hij op het idee voor dit project. Vanuit de Technische Universiteit Delft werden wij begeleid door ir. B.R. Sodoyer, van wiens jarenlange ervaring op het gebied van webtechnologie en begeleiding van soortgelijke projecten we hebben geprofiteerd. Wij danken hen hartelijk voor de hulp en het advies dat wij van hen ontvangen hebben.

Inhoudsopgave

Hoofdstuk 1

Inleiding

Bij het ontplooien van projectmatige activiteiten gaat het vrijwel altijd in essentie om het beheren, rangschikken en organiseren van informatiestromen. Vaak wordt de aanwezige informatie onvoldoende benut doordat deze niet op de juiste wijze (plaats,codering,etc) is opgeslagen. Optimalisering van informatiebeheer maakt niet alleen informatie toegankelijker voor buitenstaanders, maar kan er ook voor zorgen dat gegevens gemakkelijk tegen elkaar kunnen worden afgezet en zodoende niet eerder ontdekte dwarsverbanden aan het licht komen.
Een goed informatiebeheersysteem is kortom onontbeerlijk voor elk bedrijf of instelling. Dit geldt vooral wanneer projectmatig wordt gewerkt, vanwege de vaak grote hoeveelheid specifieke gegevens die ordelijk met elkaar moeten worden verbonden.
Voor u ligt onze (bescheiden) bijdrage aan de zoektocht naar het optimale informatiebeheersysteem. Wij hebben ons daarbij gericht op de ordening van projectdocumenten, in het bijzonder technische tekeningen. Dit sloot het beste aan bij de behoeften en wensen van DENC en paste binnen de tijd die voor dit project is gereserveerd.

Probleemstelling

De centrale probleemstelling van ons project is kort gezegd dan ook de volgende: Hoe kunnen technische tekeningen en generieke documenten zoals tekstdocumenten, afbeeldingen en PDF documenten voor alle stakeholders op een optimale manier opgeslagen worden? Essentieel daarbij is dat de informatie gedurende een lange of zeer lange periode van tijd beschikbaar moet blijven.

Opzet verslag

Ons verslag is als volgt opgezet: de uitvoering wordt van week tot week besproken. Ontwerpkeuzes worden gespecificeerd en toegelicht. Vervolgens trekken wij daar een aantal conclusies uit, formuleren we enkele aanbevelingen en uitbreidingsmogelijkheden. In hoofdstuk 4 wordt het project geëvalueerd, waarna we een overzicht geven van onze bronnen. Een groot deel van de documenten die het ontwerp beschrijven zijn te vinden in de appendices zoals aangegeven in de inhoudsopgave.

Hoofdstuk 2

Implementatie

2.1 Week 1

Het doel van deze week was om de specificaties van het project op te stellen. Door overleg met de opdrachtgever was van voor af aan duidelijk wat het einddoel van ons project moest zijn, namelijk een systeem te maken dat geschikt is om documenten te beheren. Nevendoelstellingen waren onder meer gebruiksgemak, uitbreidbaarheid en onderhoudbaarheid.

Het opstellen van de specificaties was in sommige gevallen vrij ingewikkeld, want wat is een eenvoudige gebruikersinterface? Wat is een betrouwbare manier om bestanden te plaatsen? Waar ligt de grens van integratie met de desktop?

Andere specificaties lagen meer voor de hand, eenvoudigweg omdat de keuzemogelijkheden beperkt waren. Het geheel moest namelijk passen binnen de bestaande infrastructuur, wat eigenlijk enkel betekende dat er gebruik gemaakt moest gaan worden van de bestaande servers en de bestaande inloggegevens van DENC medewerkers.

Tot slot waren er nog specificaties met betrekking tot de bestanden. Omdat we te maken hadden met AutoCAD tekeningen en dit formaat niet openbaar is en er geen goede documentatie over beschikbaar is , moesten wij uitgebreid onderzoek doen naar de mogelijkheden om in dit formaat te lezen en/of schrijven.

De specificaties zoals we die hebben vastgelegd zijn te vinden bij het Architectural design document in de bijlagen.

2.2 Week 2

Deze week stond in het teken van de verdere oriëntatie en evaluatie van tools die we zouden gaan gebruiken bij de ontwikkeling.

2.2.1 AutoCAD tekeningen

We zouden zoals gezegd met name te maken krijgen met AutoCAD tekeningen. Daarom was het belangrijk om uit te zoeken wat we als mogelijkheden hadden om tekeningen in dit formaat te beheren.

We wilden op z'n minst de mogelijkheid hebben om previews te maken van deze tekeningen, maar indien mogelijk lagen er ook nog zaken als tekstextractie en conversie naar PDF in de planning. Al snel hadden we enkele opties gevonden.

Er is een programma genaamd cad2svg van Renaud Bompuis [**?**] dat het mogelijk maakt om AutoCAD tekeningen om te zetten naar SVG bestanden. SVG bestanden bestaan uit een vector representatie van het object, moderne browsers kunnen deze direct weergeven in de browser. Helaas is dit niet het geval voor alle browsers, dus er zou nog minstens één extra stap nodig zijn als we dit programma zouden willen gebruiken voor ons beheer systeem. Gelukkig is SVG vastgelegd in een open standaard, wat in dit geval betekent dat er genoeg implementaties zijn voor het omzetten van SVG naar andere formaten.

De andere optie om handelingen met AutoCAD tekeningen te verrichten was de aanschaf van de DWGDirect bibliotheek van de OpenDWG Alliance [**?**]. Deze bibliotheek geeft toegang tot de data in AutoCAD tekeningen door middel van een zeer ingewikkelde API. Kiezen voor deze bibliotheek zou echter wel als gevolg hebben dat we zelf ook de converters moeten implementeren, wat op zichzelf al heel wat werk kan zijn. De logische keuze was dan ook om in eerste instantie gebruik te maken van cad2svg in combinatie met ImageMagick en de Batik Rasterizer.

2.2.2 Andere bestanden

Omdat we ook andere bestanden een zinnige grafische representatie wilden geven hebben we gezocht naar een uniforme manier om bestanden om te zetten naar afbeeldingen. Onze insteek was dat het handig was om regels op te stellen voor de bronbestanden. We hebben besloten om dit te doen aan de hand van het mime-type van het bestand, zodat we voor verschillende typen bestanden verschillende regels voor het omzetten konden maken. Toen we dit besloten hadden, bedachten we dat dit ook gebruikt kon worden om andere afgeleiden van bestanden te maken naast afbeeldingen. Dit kan bijvoorbeeld ook gebruikt worden om een PDF van een AutoCAD tekening te maken of een PDF van een Word document.

2.2.3 PHP framework

De DENC servers beschikken reeds over PHP installaties, dus de meest voor de hand liggende programmeertaal was PHP. Zelf hadden we al besloten dat we object-geörienteerd wilden programmeren en uiteraard behoort dit ook tot de mogelijkheden met PHP. Er zijn voor PHP een heleboel frameworks beschikbaar die al een grote hoeveelheid basisfunctionaliteit leveren zodat je deze niet zelf hoeft te schrijven. We hebben onder andere Zend Framework, CodeIgniter en het zelf schrijven van een framework bekeken als opties. Zend Framework is hier het meest compleet, CodeIgniter zit tussen zelf schrijven en Zend Framework in . Zelf schrijven zou in ons geval hebben betekend dat we het merendeel van de tijd zouden hebben besteed aan het ontwikkelen van een basis en niet ons product in 10 weken hadden kunnen maken. Er zijn echter ook nadelen aan het gebruik van een kant en klaar framework: de flexibiliteit is logischerwijs beperkter dan bij het zelf schrijven.

2.2.4 Client server interactie op website

Om een gebruiksvriendelijke website te maken is het nodig om na te denken over de geboden ervaring aan de gebruiker. Op dat moment hadden wij de keuze uit drie technieken om de gebruikersinterface te bouwen.

- (X)HTML (zonder scripting)
- (X)HTML + JavaScript
- Flash

Wij zijn geen grote fans van Flash omdat het niet open is, er maar een implementatie van is en het lang niet op alle platforms beschikbaar is. Daarnaast is het veel te veel werk om een HTML fallback te maken voor een applicatie geschreven in Flash. Flash viel vanwege deze bezwaren af. Omdat (X)HTML als basis zou dienen voor onze applicatie besloten wij om waar mogelijk JavaScript te gebruiken om bepaalde acties te vereenvoudigen. We hebben dus gekozen om gebruik te maken van progressive enhancement en bieden zodoende de gebruiker waar mogelijk een mooiere of interactievere ervaring.

Naast de keuze voor JavaScript is, om een goede applicatie te maken, de keuze voor een JavaScript framework van groot belang. Er is tegenwoordig een groot aantal JavaScript frameworks beschikbaar, waaronder jQuery, MooTools, Prototype, script.aculo.us en Dojo. Het was voor ons moeilijk om hier een keuze uit te maken. Uit de websites en commentaren op internet, werd ons niet duidelijk welke de beste keus zou zijn. Uiteindelijk hebben wij in eerste instantie voor Dojo gekozen omdat dit de standaard leek te zijn bij het Zend Framework.

2.3 Week 3

Deze week hebben we een aantal essentiële keuzes gemaakt. We hebben allereerst besloten om Zend Framework te gebruiken zodat we daarmee een heleboel basisfunctionaliteit beschikbaar zouden hebben. Daarnaast hebben we zoals gezegd gekozen om Dojo te gebruiken als JavaScript framework, omdat Zend integratie met Dojo biedt.

Deze week heeft verder in het teken gestaan van een nadere kennismaking met de door ons gekozen tools. We hebben deze week Subversion opgezet om onze code te plaatsen en zijn begonnen met het opzetten van een omgeving voor Zend Framework. Daarnaast hebben we de server ingericht zodat we eenvoudig onze applicatie online konden testen en waar nodig hebben we de juiste software op onze werkstations geïnstalleerd.

2.4 Week 4

Het eerste wat we deze week gedaan hebben is het opzetten van Trac. Trac maakt het mogelijk om eenvoudig te zien wat er veranderd is aan een subversion repository en biedt daarnaast mogelijkheden voor bug-tracking en een wiki. Na het opzetten van Trac zijn we begonnen met het invullen van

informatie in de Wiki. De meeste informatie betrof implementatiespecifieke eisen en een eerste opzet van de database.

De eerste code in Subversion bevatte direct een layout die voldeed aan onze eisen om eenvoudig te kunnen testen. Omdat authenticatie en autorisatie erg belangrijk zijn binnen dit project was de eerstvolgende stap de mogelijkheid om in te loggen. Nadat we een werkend inlogformulier hadden gemaakt, is er een ACL toegevoegd om ook te voorzien in de autorisatie.

Hierna zijn we begonnen met een eerste implementatie van de document browser. Deze was al snel in staat om documenten uit het model weer te geven en door de mappen heen te bladeren. Het model is uitgebreid met een ACL om te zorgen dat niet iedereen bij alle documenten kon en dit vormde de eerste basis voor ons documentbeheersysteem.

De rest van deze week zijn we bezig geweest met het uitbreiden van ons basis bestandsbeheer. In eerste instantie maakten we nog geen gebruik van icoontjes om de mappen en bestanden weer te geven, maar dit hebben we toegevoegd om het geheel iets gebruiksvriendelijker te maken. Daarnaast hebben we meer acties toegevoegd zoals het wijzigen van bestandsgegevens (naam, omschrijving). Tot slot hebben we de belangrijkste functionaliteiten geïmplementeerd: downloaden en uploaden.

2.5 Week 5

Uit het overleg naar aanleiding van de werkzaamheden van week 4 kwam naar voren dat we moesten zorgen dat enkele onderdelen interactiever zouden worden. Zo is bij het downloaden van een bestand wel duidelijk hoe lang het nog gaat duren, bij het uploaden van een bestand is het slechts in enkele gevallen duidelijk.[1] Dit probleem bleek niet heel eenvoudig op te lossen. Zend leek enige ondersteuning te bieden voor een zogenaamde progress bar bij het uploaden van bestanden en er zijn alternatieve methoden zoals het gebruik van een Flash uploader om bestanden te uploaden. Omdat de flash ondersteuning erg slecht is op de meeste platformen [2]

en soms zelfs totaal ontbreekt [3].

We kozen ervoor om door middel van native ondersteuning in de meeste browsers de voortgang van het uploaden weer te geven.

Na het aanmaken van de juiste objecten in de PHP code om de gebruiker de voortgang van een upload te tonen, lukte het ons niet om daadwerkelijk een voortgang zichtbaar te maken. Sterker nog, het uploaden van bestanden werd onmogelijk gemaakt door de toegevoegde code. Na een zoektocht op internet bleek dat de integratie tussen Zend en Dojo toch niet het niveau had dat we verwachtten. We hebben gekeken of het een probleem was dat we zelf op zouden kunnen lossen en belangrijker nog, of het een probleem was dat we wilden oplossen. Uit alle informatie die we op internet tegenkwamen bleek dat het een verstandigere keuze was om een andere javascript bibliotheek te gebruiken.

[1] Google Chrome zien hoeveel procent van het bestand al geüpload is. Browsers als Firefox proberen in de zogenaamde progress bar ook een indicatie van het proces te geven, maar dit is vaak niet duidelijk.

[2] In Windows is er redelijke ondersteuning in Internet Explorer, bij alle andere browsers zijn er verschillende problemen bekend. In Linux sluit Flash de browser volledig buiten als er een proces bezig is, totdat Flash klaar is. Dit zorgt er voor dat statusinformatie ook niet zichtbaar is tot het uploaden is voltooid.

[3] De iPhone heeft geen ondersteuning voor Flash, Apple hecht hier geen waarde aan en is niet van plan om Adobe tegemoet te komen om ze te helpen bij het ontwikkelen van een Flash implementatie voor de iPhone.

Er zijn genoeg javascript bibliotheken beschikbaar, maar jQuery leek voor ons toch de meest bruikbare. Naast de basisfunctionaliteit in jQuery is er ook nog veel extra functionaliteit beschikbaar door middel van plugins. Zo is er een plugin beschikbaar die het uploaden van bestanden afhandelt, waarna we zelf alleen nog maar code hoefden toe te voegen voor het weergeven van de voortgang van het uploaden.

Nadat we zichtbaar voortgang hadden bij het uploaden van bestanden zijn we verder gegaan met het verbeteren van de weergave van de bestandsbrowser. Zo hebben we specifieke iconen toegevoegd voor een aantal bestandstypen en een preview vak waarin afbeeldingen al zichtbaar waren zonder dat het nodig was om een bestand te downloaden.

De tweede helft van de week hebben we de data modellen verbeterd. Er was nog geen controle op dubbele bestandsnamen en bestandsnamen mochten ook nog leestekens bevatten. Deze controles hebben we toegevoegd in de modellen en daarna hebben we de interface aangepast zodat fouten direct zichtbaar zouden zijn.

Dit wees ons direct op een volgend probleem. Als er na het uploaden van een bestand een fout in de bestandsnaam zat, dan moest het bestand opnieuw worden geüpload. Dit probleem hebben we opgelost door eerst te controleren of alle ingevulde data wel geldig was, voordat de gebruiker daadwerkelijk het bestand kon uploaden.

Deze oplossing gaat er wel van uit dat de gebruiker javascript ingeschakeld heeft staan in zijn browser, maar na overleg hebben we besloten dat we daar van uit mogen gaan. We hebben een melding toegevoegd aan de website indien javascript niet was ingeschakeld, met de mededeling dat enkele onderdelen op de website afhankelijk zijn van javascript en mogelijk niet goed zullen werken als javascript niet beschikbaar is.

Tot slot hebben we deze week de Advanced PHP Cache geïnstalleerd om de snelheid van de applicatie te bevorderen. APC zorgt er voor dat een PHP bestand niet elke keer opnieuw omgezet hoeft te worden naar bytecode, maar ook uit het geheugen kan worden geladen als het niet aangepast is. Daarnaast zijn er nog andere mogelijkheden met APC die we eventueel wilden gaan gebruiken in de toekomst.

2.6 Week 6

Omdat we in de voorgaande week al previews konden weergeven voor afbeeldingbestanden werd het nu ook prioriteit om dit bij andere bestanden weer te kunnen geven. Dit betekende dat we buiten de website zelf ook een applicatie nodig hadden om het mogelijk te maken previews te genereren. We hebben besloten om dit simpel, maar toch generiek te houden.

De implementatie bestaat uit een klein Python script dat de server vraagt welke taken er uitgevoerd moeten worden en deze daarna uitvoert. De taken zijn commando's zoals ze ingevuld worden in een shell, met twee placeholders voor de invoernaam en de uitvoernaam. Door een aantal taken achter elkaar uit te voeren was het ook mogelijk om ingewikkelde conversies uit te voeren, zoals van DWG naar PNG met als tussenvorm een SVG bestand.

Het werd nu ook belangrijk om een aantal functies duidelijk af te splitsen van de code die te maken had met bestandsbeheer. De code voor het opvragen van een preview werd in een nieuwe

module geplaatst. Daarnaast hebben we met behulp van een zogenaamde ViewHelper een functie gemaakt die het mogelijk maakt om makkelijk het juiste pictogram voor een bestand op te zoeken, i wat later ook de mogelijkheid kon bieden om direct de preview weer te geven.

De rest van deze week heeft in het teken gestaan van het verbeteren van de kwaliteit van de code. We gebruikten in eerste instantie het standaard Zend_Db_Table_Row object voor alle data die uit de database kwam. Het werd echter al snel duidelijk dat een aangepast Row object voordeel zou bieden, dus het was nu tijd om aangepaste i Row objecten te maken voor vrijwel alle klassen die te maken hadden met bestandsbeheer.

Daarnaast hadden we een losse klasse per database tabel, terwijl dit juist minder nuttig was. Uit de code was inmiddels duidelijk geworden dat er een betere optie was, namelijk gebruik maken van een Zend_Db_Table_Definition. Het omzetten kostte wel wat tijd, maar zorgde er voor dat we minder klassen hadden wat de snelheid van onze applicatie weer ten goede zo moeten komen.

Tot slot hebben we de code aangepast aan nieuwe functionaliteit die inmiddels in Zend Framework beschikbaar was, zoals verbeterde integratie met LDAP. Hier kwamen we ook nog wat fouten tegen in de implementatie in Zend, die we upstream hebben gemeld. Dit was een leuke ervaring, omdat zowel wij als de ontwikkelaars van de code van Zend er wat van konden leren.

2.7 Week 7

Deze week stond in het teken van beheer. Tot nu toe werden alle beheertaken direct op de database uitgevoerd. Met name omdat het web stateless is en de manier waarop PHP gebruikt wordt ook, levert dit geen problemen op. Aan de andere kan is het uiteindelijk wel nodig dat alle beheerstaken ook uitgevoerd kunnen worden door mensen zonder technische kennis, wat op dat moment nog wel een probleem was.

We hebben deze week interfaces gemaakt voor het beheer van gebruikers, rechten en rollen. Gelukkig mogen we bij het maken van deze interfaces meer verwachten van de gebruiker dan bij de interfaces die voor eindgebruikers gemaakt moeten worden, hetgeen deze taak een stuk makkelijker maakte.

2.8 Week 8

Deze week hebben we de bestand browser uitgebreid met enkele extra functies. Zo hebben we de optie toegevoegd om een aantal bestanden te dowloaden als een ZIP bestand.

Dit leek ons in eerste instantie erg lastig omdat we niet concreet wisten hoe we dit probleem aan moesten pakken. Het bleek echter relatief simpel. Een zip bestand bestaat uit een header en daarna streams van gecomprimeerde data voor elk bestand. Het is dus ook mogelijk om realtime op de server een zip bestand te streamen. Het enige nadeel is dat het niet mogelijk is om een dergelijke download te stoppen en later voort te zetten, omdat het voor de server niet bekend is waar een bestand begint.

Daarnaast hebben we de optie toegevoegd om bestanden in een alternatief formaat te downloaden. Dit was met name belangrijk voor AutoCAD tekeningen omdat niet al DENC's klanten over een

DWG viewer beschikken. De meeste mensen beschikken wel over een PDF viewer, dus hebben we het mogelijk gemaakt om een AutoCAD bestand ook als PDF te downloaden.

Deze optie is ook eenvoudig toe te voegen voor office documenten, waardoor het mogelijk wordt om deze als PDF te downloaden. We hebben de implementatie hiervan niet gemaakt omdat hier (nog) geen belang bij was binnen DENC. De meeste mensen kunnen ten slotte zonder problemen office documenten openen.

2.9 Week 9

Deze week hebben we met name tijd besteed aan het wegwerken van schoonheidsfoutjes en klein onderhoud, zodat in de toekomst andere mensen eenvoudig kunnen werken met het documentbeheersysteem en uitbreidingen makkelijk toe te voegen zijn.

2.10 Week 10

Deze week was de laatste week van het bachelorproject. We hebben van deze week gebruik gemaakt om het verslag te schrijven. Daarnaast zijn we door het schrijven van het verslag nog wat kleine fouten in implementatie en documentatie tegengekomen. Een aantal van deze fouten hebben we direct opgelost, andere fouten hebben we opgeschreven voor eventuele verdere werkzaamheden.

Hoofdstuk 3

Conclusie

Bij de uitvoering van een project is het van groot belang allereerst het werkveld te begrenzen. Al hebben de (te gebruiken) tools hier invloed op, toch moet men zich daar niet te veel door laten leiden. Ook is goed inlezen van essentieel belang bij het opzetten van een domeinspecifiek project. Pas daarna is het mogelijk daadwerkelijk knopen door te hakken. Sommige keuzes die aan het begin van het project zijn gemaakt, bleken later niet de juiste. Dit ondanks het feit dat ze gezien de op dat moment beschikbare informatie de meest optimale leken. Zo lijkt de implementatie van ACL conceptueel gezien relatief eenvoudig, maar is het in de praktijk toch lastig. Halverwege het project kwamen we er achter dat bij het maken van een web based oplossing, er relatief veel tijd gaat zitten in het overbruggen van de kloof tussen functionaliteit en gebruiksgemak. Een product maken dat uiteindelijk te vermarkten is, vergt een goed oog voor detail. Tot slot is het werken met bestandsformaten die niet openbaar beschreven zijn (in ons geval Autocad) op zijn zachtst gezegd lastig.

Ondanks al deze lessen zijn wij er naar onze mening toch in geslaagd een werkbaar product te maken. Dit in relatief korte tijd en een op manier waardoor het nu daadwerkelijk gebruikt wordt door onze opdrachtgever.

Hieruit concluderen we dan ook dat het project een succes is. We zijn zelf tevreden over het verloop van het project en we zijn blij dat we aan de afronding zijn toegekomen. Tot slot hebben we veel opgestoken van het uitvoeren van het project en daarmee hebben we ook wat onze studie betreft een mijlpaal bereikt.

Uitbreidingsmogelijkheden

In het MosCow document (Appendix:G pag 53) definiëren we al enkele eigenschappen die we graag in het systeem zouden zien. Al deze Would-have's zijn zonder technische problemen te integreren in het bestaande systeem. Dit komt met name door de modulaire opbouw en onze scheiding tussen model, view en controller.

Daarnaast is het mogelijk om het systeem uit te breiden naar een systeem om projectfases mee te beheren. Dit lijkt erg op versiebeheer bij programmeren, alleen is het niet altijd mogelijk om verschillen tussen versies te extraheren. De beschikbaarheid van een dergelijke systeem kan het doorlopen van project-trajecten vergemakkelijken, wat er voor zorgt dat projectmanagers minder tijd nodig hebben voor de formaliteiten en meer tijd beschikbaar hebben voor kwaliteitscontrole.

Tenslotte is het mogelijk om het document beheer te integreren in een ander systeem. De data objecten zijn goed toegankelijk en het systeem is eenvoudig uit te breiden met REST ondersteuning waardoor het ook goed bruikbaar is als een meer generieke storage oplossing. Mocht iemand op een lager niveau aanpassingen willen maken, dan zorgt de goede documentatie er voor dat dit mogelijk is.

Hoofdstuk 4

Evaluatie

In deze sectie geven we een kort overzicht van de belangrijkste ervaringen die we hebben opgedaan tijdens het project. Daaruit zal blijken dat wij een aantal problemen zijn tegengekomen waaruit wij lering kunnen trekken voor de toekomst.

4.1 Zend Framework

Onze eerste ervaring met Zend was dat het ontzettend goed gedocumenteerd was. Helaas waren er zo nu en dan toch momenten dat een blik op de code meer duidelijkheid gaf dan de documentatie. Over het algemeen waren we wel tevreden over de documentatie.

Het gebruik van Zend Framework was ook onze eerste ervaring met een volledig PHP framework. Door hiermee in aanraking te komen, weten we voortaan wat wij ervan kunnen verwachten.

We hebben ook ondervonden dat terwijl wij bezig waren met ontwikkelen er functionaliteit aan Zend werd toegevoegd die we al eerder nodig hadden. In de meeste gevallen zou het geen direct voordeel opleveren om daarna weer over te stappen op de nieuwe Zend functionaliteit, met name omdat we gebonden waren aan een planning. Voor de onderhoudbaarheid van het project is het wel belangrijk dat deze aanpassingen nog gemaakt worden, omdat het in de toekomst zeker tijd kan besparen.

Wij zijn uiteindelijk tot de conclusie gekomen dat een aantal onderdelen van Zend niet zo goed samenwerken als we dat graag zouden willen. Het huidige internet draait om data, maar Zend Framework draait om een opsplitsing in categoriën. Zo is er geen eenvoudige mogelijkheid om een object uit de database te valideren, of automatisch een formulier te laten genereren om het object aan te passen. Daarnaast is het vaak nodig om een object te maken van een klasse, terwijl daarna maar enkele functies van dat object worden aangeroepen. Dit wordt mogelijk veroorzaakt door de manier waarop PHP omgaat met methoden en variabelen in een statische context. Dit maakt het echter niet minder gebruiksvriendelijk Het zou vaak ook makkelijker zijn als klassen eenvoudigere namen kunnen gebruiken.

4.2 jQuery

jQuery heeft ons geweldig geholpen met het maken van een interactieve applicatie. We zijn erg tevreden met de eenvoud waarmee jQuery te gebruiken is. Op het moment dat we een probleem hadden dat niet direct met jQuery opgelost kon worden, was er vaak wel een plugin beschikbaar die precies kon doen wat wij wilden. We begrijpen nu ook heel goed waarom veel grote websites gebruik maken van jQuery.

Het enige nadeel van jQuery is dat het meer een toevoeging is op de basale webpagina. jQuery is niet gemaakt om interactieve web-applicatie te maken, maar meer om een bestaande web-applicatie interactief te maken. In het begin was dit geen probleem, omdat we een applicatie wilden maken die voor iedereen toegankelijk was, dus ook voor mensen zonder javascript. Uiteindelijk wilden we echter toch meer functionaliteit toevoegen die afhankelijk is van javascript ondersteuning bij de gebruiker. Hier hebben we dus naar alternatieve oplossingen moeten zoeken of deze oplossingen zelf moeten maken.

4.3 Opdrachtomschrijving

Bij een project aan de universiteit is de opdracht meestal vantevoren compleet bekend. In dit geval werd de opdracht zo nu en dan aangepast, waardoor extra functionaliteit moest worden toegevoegd.

4.4 Verwerken van werkervaring

Na verloop van tijd zagen we in dat niet alles wat we in het begin van het project hadden gemaakt nog steeds even goed te gebruiken was. In sommige gevallen hebben we dus ook bestaande, werkende code aangepast aan wat we intussen geleerd hadden. Er blijven nog steeds stukken code die we graag opnieuw en beter zouden willen schrijven, en er zijn hele structuren in gebruik waarvoor we graag alternatieven zouden ontdekken en implementeren, maar helaas was daar binnen dit project geen tijd voor.

4.5 Reflectie op de evaluatie

4.5.1 Martijn

Ik ben tevreden over de loop en de uitkomst van het project. Ik heb veel geleerd over het maken van keuzes in het ontwerp proces in het algemeen. Onze keuzes voor het Zend Framework, jQuery, MySQL, PHP en een beetje Python zijn mischien in vergelijking tot Ruby on Rails en GWT relatief conservatief maar ook volwassen.

Integratie en Zend Framework De MVC structuuur heeft ons in een zekere mate gedwongen om onze implementatie te structureren. En het hebben van een startpunt door middel van het framework was ook prettig in de zin dat redelijk snel echt aan de slag konden. Daar tegenover staat dat

net als met alles wat je hergebruikt dat door iemand anders geschreven is je een leerkurve heb om te leren dingen op de correcte manier van uit het perspectief van het framework te doen. Wij zijn met de integratie van LDAP tegen een aantal zaken aangelopen waar de library in eerste instantie allen goede support had voor MS active directory en niet goed functioneerde tegen andere LDAP servers. In dit geval hebben we een aantal tickets aangemaakt en ondertussen zelf onze eigen implementatie gemaakt. Inmiddels is dit probleem verholpen en kunnen wij weer de ingebouwde ondersteuning gebruiken. Dit laat zien hoe belangrijk het is om ondanks het gebruik van de het werk van een derde partij zelf toch over de vereiste kennis te beschikken om dingen voor elkaar te kunnen krijgen.

Javascript en de browser De verbazing is denk ik een blijvende, De presentatie/ interactie laag van een web applicatie kost door incompatibiliteiten in de implementatie in browsers mijns inziens veel te veel tijd. Ook het testen van aanpassing moet in zoveel verschillende browsers gebeuren. Ik vond het een interresante ervaring en ik heb wardering gekregen voor het kennis niveau wat Michael op dit gebied heeft. Ook ben ik er van onder de indruk hoe desondanks deze handicaps er toch nog hele mooi interactive interfaces gemaakt worden.

4.5.2 Michael

Over het algemeen ben ik best tevreden met de keuzes die we hebben gemaakt. Ik had andere aspecten van het ontwikkelen kunnen leren als we andere keuzes gemaakt hadden, bijvoorbeeld door te programmeren in Ruby en gebruik te maken van Ruby on Rails. Aan de andere kant had dit wel de voortgang belemmerd tijdens het programmeren, door een compleet gebrek aan ervaring met een dergelijke taal. Per belangrijke keuze zal ik toelichten hoe ik terugkijk op het maken van de keuze en het belang daarvan.

Zend Framework Zend Framework was over het algemeen goed te gebruiken. Zo nu en dan ondervonden we ook enkele nadelen aan onze keus voor Zend. We liepen snel tegen een aantal problemen aan waar Zend Framework niet voldeed aan onze eisen. Een voorbeeld hiervan is de Zend LDAP implementatie. Deze was niet bruikbaar om onze LDAP directory te gebruiken voor het inloggen en al helemaal niet om daarna nog informatie van gebruikers op te vragen. We hebben zelf de nodige aanpassingen gemaakt zodat dit wel kon en inmiddels is er aan Zend Framework een uitgebreide LDAP implementatie toegevoegd die ook alles kan wat we nodig hebben.
Zo zien we wel dat we soms problemen oplossen die door een aantal mensen als een algemeen probleem worden beschouwd en dus ook in het framework nog worden opgelost. Helaas kunnen we daar vaak niet op wachten, maar soms loont het wel de moeite om achteraf onze aanpassingen te verwerpen voor de nieuwe implementatie in het framework.
Als ik vergelijk hoe ik met Zend heb kunnen werken tegenover mijn ervaring met bijvoorbeeld CodeIgniter dan zie ik dat Zend Framework een stuk eenvoudiger en krachtiger is. Ik mis zo nu en dan wel dingen aan Zend, maar deze gebreken wegen niet op tegen de voordelen van het gebruiken van een goed framework.

jQuery Door onze keus om de applicatie toegankelijk te houden voor mensen zonder JavaScript was het nodig dat we een gewone web-interface zouden maken. We heben later gekozen om deze interactief te maken met behulp van jQuery. Erg laat in het project had ik wel spijt van deze keuze, omdat we met een andere JavaScript bibliotheek een betere interface hadden kunnen maken, ten koste van gebruikers die geen JavaScript hebben.
Aan de andere kant is jQuery erg goed bevallen. Het deed precies wat we er van verwachtten en het was makkelijk te gebruiken. JavaScript schrijven blijft altijd moeilijk, met name omdat het testen soms moeilijk kan zijn. jQuery heeft echter alle verschillen tussen verschillende browsers voor ons weggenomen. Over het algemeen schreven we de code, testte deze code in Firefox en later bleek alles ook te werken in Internet Explorer. Als iets direct werkt in Internet Explorer is dat vaak een verassing en op z'n minst een hele opluchting. Al met al heb ik dus niets te klagen over jQuery. Mocht ik nogmaals gevraagd worden om een web-interface te maken, dan zou ik wel voorstellen om dit dynamischer te maken dan we op dit moment hebben gedaan. De keuze zou hier wel mogelijk weer voor jQuery kunnen vallen, ook al zijn er andere bibliotheken die meer gericht zijn op het van de grond af aan opbouwen van een interactieve web-applicatie.

Hoofdstuk 5

Bronnen

PHP http://php.net/

Zend Framework http://framework.zend.com/

d Framework Manual http://framework.zend.com/manual/en/

jQuery http://jquery.com/

jQuery UI http://jqueryui.com/

Bijlage A

Opdrachtomschrijving

A.1 DENC

DENC staat voor Design-Engineering-Contracting. Opgericht in 2000 is DENC sterk in totaaloplossingen. De integrale design & build formule biedt klanten gemak en zekerheid in hun huisvestingsproces. Zij hebben één verantwoordelijk aanspreekpunt van de eerste analyse en locatiestudie tot aan oplevering en onderhoud. Het traditionele bouwproces kenmerkt zich door vele betrokken partijen met vaak tegengestelde belangen. Dat leidt tot onnodige complexiteit, suboptimalisaties, bouwfouten, faalkosten en afstemmingsverliezen. Het werkt kostenverhogend en verlengt de doorlooptijd van het project. Door de grijze gebieden tussen al die partijen liggen verdere vertragingen en meerwerken altijd op de loer. Steeds meer opdrachtgevers kiezen daarom voor een duidelijke totaalaanpak.
Deze efficiënte Design & Build methode levert de klant een helder en inzichtelijk proces, een snelle doorlooptijd en het comfort van één professioneel aanspreekpunt. De klant is bovendien verzekerd van een vaste, scherpe prijs, een vaste opleverdatum en één overall garantie.

A.2 DENC Docs

De opdracht richt zich op het ontwerpen en (gedeeltelijk) implementeren van een systeem om de communicatie tussen DENC en haar klanten te verbeteren.

De dienst die wij voor DENC willen ontwikkelen, hebben wij in een concept folder als volgt omschreven:

> Documenten zijn een belangrijk onderdeel van een bedrijfsproces. Het beheren van deze documenten, dus gestructureerd opslaan en 'up to date' houden, dient nauw gezet te gebeuren. Tevens heeft u veel verschillende programma's nodig om alle documenten bij te werken of simpel weg te bekijken. Neem nu bijvoorbeeld tekeningen en de vele ordners waarin deze zijn opgeslagen, het enige wat u wilt is de laatste tekening bekijken en mogelijk op A3 / A4-formaat kunnen printen.
>
> DENC Netherlands B.V. kan voor u de gewenste documenten archiveren, beheren en 'up to date' houden, u kunt dan via internet uw eigen documenten raadplegen en printen.

Als u de tekeningen op groot formaat wil dan zal DENC deze uitplotten en u doen toe komen, ook het reproduceren van gehele documenten (b.v. rapporten) behoort tot de mogelijkheden.

A.3 Opdracht

De opdracht is het ontwerpen van een toegankelijk document beheer systeem. Hiervoor is het noodzakelijk dat klanten vanaf hun eigen computer gebruik kunnen maken van het systeem, zonder dat daarbij allerlei extra software nodig is die de klant nog niet geïnstalleerd heeft staan. Daarnaast moet het voor de klant aanvoelen alsof hij zijn eigen archief beheert, zonder de nadelen van eigen beheer te ervaren.

Bijlage B

Definities en afkortingen

In deze sectie zullen een aantal afkortingen en definities uitgelegd worden. In de rest van de tekst zullen deze over het algemeen in de korte vorm voorkomen en niet opnieuw uitgelegd worden.

- AJAX *Asynchronous JavaScript And XML* is één van de manieren waarop een webpagina met behulp van javascript informatie kan versturen naar of opvraagt bij de server. Alhoewel de X voor XML staat, wordt deze term vaak ook gebruikt wanneer de informatie in een ander formaat wordt verstuurd, zoals bijvoorbeeld JSON.
- AutoCAD : Ontwerppakket voor het maken van technische tekeningen.
- DWG : Extensie voor AutoCAD bestanden, kort voor drawing.
- Hash : Cryptografische berekening naar een verkorte representatie. Bij gebruik van recente algoritmes levert de berekening een praktisch unieke waarde op. Theoretisch is het mogelijk om voor verschillende invoer een identieke uitvoer te krijgen. In de praktijk komt dit echter niet voor. Het berekenen van een dergelijke invoer wordt praktisch onmogelijk gemaakt door de snelheid van computers en de eveneens beperkte duur van een mensenleven.
- HashStore : Opslagmethode waarbij een bestand wordt opgeslagen op een locatie bepaald aan de hand van een hash van de inhoud van het bestand.
- JavaScript : Client-side scripting taal. JavaScript wordt al jaren ondersteund in alle grote browsers en is de meest betrouwbare manier om gebruikers een interactieve web-applicatie te bieden.
- JSON : JavaScript Object Notation. Eenvoudige manier om objecten te representeren in JavaScript. Ook PHP kan JSON objecten maken.
- LDAP *(Light-weight Directory Access Protocol)* : Een protocol dat beschrijft hoe gegevens uit directoryservices benaderd moeten worden over een netwerk. De gegevens zijn op een hiërarchische manier, gegroepeerd naar een bepaald attribuut opgeslagen en op te vragen.
- MVC *(Model-View-Controller)* : Patroon dat het ontwerp van complexe toepassingen in drie eenheden met verschillende verantwoordelijkheden deelt.

- MySQL : Gratis relationeel database management systeem aanspreekbaar met behulp van SQL.
- ORM *(Object-relational mapping)* : Methode voor het converteren van data tussen relationele databases en object-geörienteerde programmeertalen.
- PHP *(PHP Hypertext Preprocessor)* : Recursieve definitie van de naam van de programmeertaal PHP.
- PHP-APC *(Alternative PHP Cache)* : Deze uitbreiding voor PHP zorgt ervoor dat een aantal PHP objecten in het geheugen blijven zodat PHP ze niet telkens opnieuw hoeft te laden.
- SQL *(Structured Query Language)* : Taal om data op te vragen uit de database. Verschillende servers spreken deze taal vaak in hun eigen dialect.

Bijlage C

Orientation

C.1 Introductie

C.2 Beslissingen

Om het document beheer systeem voor iedereen toegankelijk te maken wil DENC het project ontwikkelen als web-applicatie. Hier vloeit een andere eis uit voort: het systeem moet cross browser compatible zijn.

C.2.1 Programmeertaal

Vanwege onze ervaring met het ontwikkelen van web-applicaties konden we al snel een keuze maken voor de programmeertaal. We hebben gekozen om het systeem te schrijven in PHP, onder andere vanwege de volgende punten:

Object Orientated Het is mogelijk om object orientated code te schrijven in PHP. Omdat we hier gebruik van willen maken is dit een noodzakelijke eigenschap.

Frameworks Er is een groot aantal PHP frameworks beschikbaar. Deze frameworks vereenvoudigen het maken van een applicatie en bieden een solide basis. Daarnaast bieden een aantal frameworks vrijwel alle voordelen die bijvoorbeeld gehaald kunnen worden uit het gebruik van een taal als Ruby.

Documentation De PHP website biedt duidelijke en volledige documentatie, vaak zelfs met extra documentatie en voorbeeldcode van gebruikers.

Performance PHP is één van de meest geschikte talen voor het bouwen van web-applicaties als we kijken naar de prestaties. Sommige grote websites maken al jaren gebruik van PHP, en kunnen zonder problemen de vele duizenden bezoekers per dag de juiste informatie leveren.

Redenen waarom we andere talen niet hebben gekozen:

- Mogelijke patent-problemen

- Mogelijke licentie-problemen
- Langzame executie
- Geen goede server implementatie
- Geen basis zoals de frameworks voor PHP

C.2.2 framework

Ik had al ervaring met het CodeIgniter framework, wat bekend staat als een licht en klein framework. Helaas liep ik met name tegen beperkingen aan. CodeIgniter is als framework niet erg uitbreidbaar, al claimen de developers dat het wel zo is. Daarnaast probeert CodeIgniter een aantal principes van PHP te verbergen. In plaats van dat CodeIgniter mensen leert werken met objecten in het algemeen, maakt CodeIgniter het alleen mogelijk om object te gebruiken die aan een strak patroon voldoen.
CodeIgniter viel dus al af, maar een logische ander alternatief hadden we nog niet gevonden. Op aanraden van een medestudent zijn we gaan kijken naar Zend Framework. De Zend website komt erg professioneel over, met een duidelijke introductie tot het framework. Naast de introductie is er echter nog veel meer te vinden, alles op een overzichtelijke manier gepresenteerd. Zend biedt op zijn website zeer uitgebreide *documentatie voor de meeste klassen, een complete* API reference en een goed gestructureerde *performance guide.*
Naast het feit dat het project zelf dus goed en compleet uitgevoerd is, is ook de code zeer compleet. Zend Framework biedt alle componenten om aan de slag te gaan met een Model-View-Controller structuur. Views en controllers zijn strak gedefinieerd, models kunnen worden gebruikt zoals je dat zelf wilt. Zend bevat ook een complete boom aan database klassen, met als hoogtepunt de Zend_Db_Table klasse. De database klassen in Zend zijn backend onafhankelijk, dus het is geen probleem om MySQL, PostgreSQL, SQLite of een ander (R)DBMS te gebruiken. Daarnaast zorgt de Zend_Db_Select klasse dat de ontwikkelaar bijna geen query meer met de hand hoeft te schrijven.
De keuze voor Zend bleek al snel een succes, na verloop van tijd werd zelfs duidelijk dat we nog beter gebruik van alle mogelijkheden van Zend konden maken dan we in eerste instantie dachten.

C.2.3 RDBMS

Als database systeem hebben we gekozen voor MySQL. Bijna iedereen die een keer een web-applicatie heeft geschreven is bekend met MySQL, zo ook wij. Er was al een MySQL database in gebruik voor het interne CRM (Customer Relationship Management) van DENC en er was sprake van een mogelijke integratie-eis om te zorgen dat CRM en het nieuwe document beheer netjes samen zouden werken. Ten slotte biedt MySQL bij gebruik van de InnoDB engine ondersteuning voor transactions. Omdat transactions belangrijk zijn om consistentie te kunnen garanderen in een grotere applicatie willen we deze optie op z'n minst open houden. Vanwege de abstractie tussen backend en frontend in Zend Framework is het echter mogelijk om nog steeds te wisselen naar een ander (R)DBMS.

C.2.4 Systeem architectuur

We maken gebruik van de Zend Framework componenten om onze applicatie te schrijven met behulp van een Model-View-Controller structuur. Naast models, views en controllers hebben we echter ook een eigen library, waarin we met name Zend Framework componenten extenden om functionaliteit die wij nuttig achten toe te voegen aan de basis (vaak abstracte) klassen. In sommige gevallen extenden we Zend componenten om direct bij het maken van het object een aantal eigenschappen in te stellen. Dit zorgt er onder meer voor dat controllers en views niet of zo min mogelijk gebruikt hoeven te worden voor het instellen van deze eigenschappen die altijd het zelfde zijn en bovendien zodat de code in controllers en views kort en duidelijk kan blijven.
Over het algemeen maken we voor een gedeelte van het systeem gebruik van de volgende componenten:

- 1 controller
- 1 of meer (data) modellen
- 1 of meer database klassen
- 1 of meer form klassen
- 1 of meer views

De modellen corresponderen over het algemeen met één database tabel en database klasse, maar in sommige gevallen biedt het model direct data afkomstig uit (een join van) meerdere tabellen. De form klassen geven aan wat geldige invoer is en worden gebruikt voor het valideren van invoer. Ten slotte zorgen de views voor een correcte weergave van de data, zonder dat daar op een andere plek al rekening mee gehouden hoeft te worden.

C.2.5 JavaScript bibliotheek

Zend Framework biedt standaard integratie met de Dojo JavaScript bibliotheek. Omdat we vinden dat bij het maken van een web-applicatie JavaScript noodzaak is en geen optie, hebben we gekozen om een JavaScript bibliotheek te gebruiken. Zelf had ik al ervaring met MooTools, een lichte en eenvoudige JavaScript bibliotheek die makkelijk in gebruik is. Omdat Zend integratie met Dojo aanbiedt, wilden we Dojo gaan gebruiken. Echter, snel bleek dat Dojo niet erg eenvoudig was om mee te werken, als we uberhaupt werkend konden krijgen wat we wilden maken. We zijn dus gaan evalueren wat een goede keus was. We kwamen tot de volgende drie logische opties:

Dojo Schijnbaar onbruikbaar, maar wel integratie met Zend.

MooTools Licht, eenvoudig, maar bekend.

jQuery Onbekend voor ons, grote userbase, in gebruik door grote websites.

Na kort naar jQuery te hebben gekeken hebben we besloten om dat te gaan gebruiken. Naast het gemak wat jQuery biedt als JavaScript bibliotheek is er ook een bijbehorend interface framework: jQuery-UI. jQuery-UI maakt het nog makkelijker om een gewone website om te toveren in een full-featured web-2.0 applicatie, dus hier hebben we dankbaar gebruik van gemaakt.

C.3 Ervaringen Martijn

C.3.1 Inleiding

Voor ik aan dit project begon was ik al werkzaam bij DENC. Mijn dagelijkse werkzaamheden komen echter niet overeen met hetgeen wij in dit project proberen te bewerkstelligen. Ik heb tot nu toe al veel geleerd, zowel op technisch als op organisatorisch vlak. En dat terwijl er nog een hoop werk voor de boeg is!

C.3.2 Technische Refletie

Omdat het kennisniveau van Michael en mij op het gebied van PHP, CSS en HTML nogal verschilt, was het voor mij in het begin best wennen. Mijn laatse project van enige omvang heb ik met PHP gedaan in 2003, en toen heb ik mij zo ver mogelijk gehouden van alles wat met de gebruiker te maken heeft. PHP is sinds ik het voor het laatst gebruikt heb uitgegroeid tot een taal met een breed scala aan mogelijkheden en eigenaardigheden. Dit samen met het feit dat om een moderne webapplicatie te maken, PHP gecombineerd moet worden met HTML, CSS en Javascript was - ondanks dat ik de systemen en hun plek in het totaal wel begrijp - een hele uitdaging. Om het allemaal nog ingewikkelder te maken, is de uiteindelijke presentatie in de browser bij gebruik van geavanceerde features, erg inconsistent . Hierdoor gaat veel tijd verloren en is ervaring een vereiste. Wat dat betreft heb ik erg veel van Michael geleerd.

C.3.3 Organizatorische Relectie

Ik heb ondervonden dat het ontwikkelen van een product in een commerciele omgeving (ondanks dat de schaal en belangen in dit geval niet eens zo heel groot zijn) totaal anders is dan dit doen binnen de muren van de universiteit. De levensvatbaarheid van een idee wordt anders beoordeeld. Daarnaast is er, anders dan bijvoorbeeld het geval is bij het tweedejaars project ST4, weinig ruimte om halfverwege het proces van richting te veranderen. Naar mijn idee leveren dergelijke beperkingen een beter proces op, misschien wel vergelijkbaar met de vormrestricties bij een gedicht, die niet ten koste gaan van de kwaliteit maar juist tot interresante, nieuwe uitkomsten leiden. Het best moeilijk om consequent code te documenteren en testcases te maken, zonder daarbij voor de korte termijn te gaan, zelfs nu wij allebei weten dat de tijd die hierin geinvesteerd wordt zich later terug betaalt. Dat wij een ongelijke technische domein-specifieke kennis hebben, heeft nogal eens voor discussie gezorgd bij het maken van ontwerpkeuzes. Dit temeer wij allebei eigenwijs zijn en het liefst meteen het ontwerp dat in ons hoofd zit zouden uitvoeren. Gelukkig helpt het om zaken op papier te zetten en je ontwerpkeuzes met argumenten te moeten onderbouwen. Op dergelijke momenten help het om een koppige teamgenoot te hebben die lastige vragen blijft stellen.

C.3.4 Conclusie

Tot nu toe ervaar ik het maken van een systeem in een “echte” omgeving als een zeer leerzaam en stimulerend proces. Het is heel leuk om theoretische kennis in de praktijk te brengen. Toch blijft het

mooiste dat wat wij maken echt gebruikt gaat worden en dat wij waarschijnlijk de kans krijgen om er ook na dit project aan te blijven werken.

C.4 Individuele ervaringen Michael

C.4.1 Werklocatie

Ik vond het erg fijn om op een kantoor te kunnen werken. Door op locatie aanwezig te zijn wordt je min of meer gedwongen om echt te werken aan het project, wat anders soms wat vergeten raakt. Daarnaast maakt het de communicatie makkelijker. Je ziet de opdrachtgever en je kan communiceren met de andere groepsleden wat het projectproces ten goede komt. Daarnaast krijg je op deze manier vaak al nuttige input tijdens het ontwikkelen van nieuwe functionaliteit.

C.4.2 Ontwikkelomgeving

Persoonlijk geef ik de voorkeur aan de *VIM* (Vi Improved) editor om mijn code (en documentatie) in te schrijven. Omdat we de beschikking hebben over computers met Ubuntu is VIM ook standaard aanwezig, dus ik kon direct aan de gang gaan. Later ben ik overgestapt op *GVIM*, de grafische variant van VIM. Dit maakt het mogelijk om meer kleuren te gebruiken voor syntax highlighting en kan gebruik maken van het klembord dat ook gebruikt wordt door andere programma's zodat ik makkelijker kan knippen en plakken tussen bijvoorbeeld de wiki en de editor. Daarnaast heb ik op mijn eigen computer een webserver geïnstalleerd om zo nieuwe features te kunnen testen voordat ze in de Subversion repository gaan.

C.4.3 Documentatie

Bij het implementeren van nieuwe functionaliteit schrijf ik eerst de functiespecificatie in phpDocumentor syntax. Op deze manier zie ik na het schrijven of dat de code die ik heb gemaakt nog klopt met wat ik in eerste instantie van plan was en dus nog past in het geheel.
Documentatie op de wiki schrijf ik vaak direct na of vlak voor het committen naar de Subversion repository. Vaak spreekt functionaliteit en code voor zich (we houden er een hele strakke structuur op na), maar soms is extra documentatie nodig. Zo hebben we onder andere een pagina waar wijzigingen in de database per revisie worden bijgehouden. Dit was de beste oplossing die we konden bedenken, zonder dat de database zelf in de repository zou gaan.

C.4.4 Issue tracking

Problemen die we tegen komen na het schrijven van code zijn vaak klein en dus is ook de oplossing vaak klein. In dat geval maak ik geen issue aan, maar los ik het probleem direct op en vermeld in de commit message dat ik een probleem heb opgelost. We hebben echter ook een aantal issues voor grotere problemen. Doordat we Trac gebruiken met post-commit hooks kunnen we in de commit message al aangeven dat we een issue willen sluiten, of verwijzen naar een wiki pagina. Hier maak ik dan ook dankbaar gebruik van, omdat dit later handig kan zijn als er een probleem opduikt.
Door gebruik te maken van milestones en daar issues aan te hangen krijgen we een overzicht van de vooruitgang. Ik ben van mening dat we eigenlijk te weinig milestones aanmaken, want we ronden wel degelijk regelmatig een milestone af op het moment dat we een nieuwe release doen.

C.4.5 Zend Framework

Ik had nog geen ervaring met Zend Framework toen ik begon aan dit project, dus in het begin moest ik vaak in de documentatie opzoeken hoe bepaalde functionaliteit te gebruiken was. Na verloop van tijd hoefde ik steeds minder te zoeken en wist ik zelf nuttige uitbreidingen te maken op Zend Framework die nu bijna overal in de code weer terug te vinden zijn.
In het begin bleek het ook moeilijk om efficient gebruik te maken van de object orientated aanpak, met name voor objecten in de database en de structuur van de modellen. Later ging dat al makkelijker en hebben we ook aanpassingen gemaakt aan eerdere modellen.
Ook het documenteren van functies en variabelen en met name klassen en packages bleef in het begin soms een beetje achter. Naar mate we vorderingen maakten ging ik hier wel beter mee om en was dat het eerste wat ik deed bij het schrijven van code.

C.4.6 jQuery

Werken met jQuery was erg makkelijk. De documentatie is helder, al is het soms nog duidelijker om even in de code te kijken welke stappen precies gemaakt worden. MooTools lijkt erg op jQuery, alleen verschillen veel functies van naam. Het grootste nadeel aan jQuery vind ik nog dat veel asynchrone functies niets aan foutafhandeling doen, waardoor ze niet bruikbaar zijn als foutcontrole van belang is. Ik ben blij dat we jQuery zijn gaan gebruiken, ik vond het een leuke persoonlijke ervaring om te leren hoe jQuery werkt.

C.4.7 HTTP response codes

Op een moment kwam ik een probleem tegen met het terugsturen van informatie op basis van een ingevuld formulier. Omdat er verschillende fouten op kunnen treden is het belangrijk om de fouten ook op verschillende manieren af te handelen. Zo kan er namelijk een fout optreden op de server waar niet in is voorzien, maar het kan ook zo zijn dat de invoer simpelweg ongeldig is. Omdat ik al werkte met verschillende HTTP response codes kwam ik op het idee om te zoeken of er niet een code was die ik kon gebruiken of misbruiken om een extra stukje informatie terug te geven. Al snel kwam ik code *409 - conflict* tegen. Deze code kan gebruikt worden als de data in het request ongeldig is. Bij voorkeur moet dan wel informatie worden teruggegeven waaruit de browser of de gebruiker kan opmaken wat wel geldige data is. Door hier het formulier met foutmeldingen terug te geven maken we efficiënt gebruik van de foutcode en kunnen we detecteren dat de invoer ongeldig was en de gebruiker het opnieuw moet proberen.

C.4.8 Terugblik

Ik ben tevreden over de voortgang van de eerste periode. Ik heb veel geleerd over technieken waar ik nog niet erg bekend mee was, met name wat betreft Zend Framework en jQuery. De omgang met de opdrachtgever was erg makkelijk, onze contactpersoon is duidelijk in zijn wensen en opmerkingen.

C.4.9 Vooruitblik

Ik hoop dat we naast de basis functionaliteit ook toe komen aan het implementeren van een stukje extra functionaliteit. Als we genoeg tijd hebben, kunnen we de documentatie nog uitgebreider maken dan het op dit moment al is. Ik hoop ook dat we mogelijk een aantal van onze uitbreidingen op het framework terug kunnen geven aan de community, zodat andere mensen er ook wat aan hebben.

Bijlage D

Plan van Aanpak

D.1 Inleiding

D.1.1 Het plan van aanpak

In dit document staat een uiteenzetting van de door ons voorgestelde probleem aanpak, een lijst met deliverables en een tijdsplanning.

D.1.2 Schets van het bedrijf

DENC Netherlands is een in 2000 opgericht bedrijf dat zich toelegt op het aan zijn klanten bieden van optimale oplossingen voor huisvestingsvraagstukken. De bedrijfsfilosofie is dat door een multidiciplinaire aanpak de geboden oplossing efficenter en effectiver wordt. Binnen het bedrijf werken architecten, werktuigbouwkundigen en civiel ingenieurs samen om tot totaaloplossingen te komen. Omdat het succes van complexe oplossingen in toenemende mate beÃ¯nvloed wordt door goede implementatie van informatietechnologie, is het bedrijf bezig ook de hieraan gerelateerde kennis te verkijgen. Dit project past dan ook in dit kader en moet vorm geven aan dit streven.

D.1.3 Achtergrond en aanleiding van de opdracht

DENC krijgt vaak opdrachten waarbij aanpassingen aan een bestaand gebouw nodig zijn. In veel gevallen blijkt al snel dat de opdrachtgever niet over de juiste bouwtekeningen beschikt en dat daardoor het gebouw opgemeten en nagelopen moet worden voordat begonnen kan worden aan de echte opdracht.
Dit probleem komt doordat de bedrijven die oorspronkelijk de opdracht hebben gegeven tot de bouw van het gebouw in kwestie, na de oplevering de tekeningen niet juist hebben gearchiveerd. Ze weten vaak simpelweg niet wat ze met de tekeningen moeten doen. Vaak komen de documenten terecht in een intern archief. De beheerder daarvan verlaat na verloop van tijd het bedrijf zonder de zaken netjes over te dragen, de nieuwe beheerder weet van niets en voor je het weet zijn de tekeningen voorgoed kwijt. Het zelfde gebeurt nogal eens na een (ingrijpende) reorganisatie, fusie, enz..
Met extern document beheer door DENC hoeven bedrijven zich geen zorgen (meer) te maken over de beschikbaarheid van hun documenten; ze zijn eenvoudig op te vragen zonder dat het bedrijf er zelf iets

voor hoeft te doen. Mocht iemand in het bedrijf niet weten dat DENC over de nodige documenten beschikt, maar wel dat DENC de documenten ooit heeft geproduceerd, dan is één telefoontje naar DENC voldoende om een account aan te maken waarna de klant weer bij zijn documenten kan.

D.2 Opdrachtomschrijving

D.2.1 Inleiding

D.2.2 De opdrachtgever

De opdrachtgever van dit project is DENC Netherlands B.V.. Zoals eerder gezegd, neemt de behoefte toe aan een geintegreerde aanpak van problemen. In een interdisciplinaire aanpak mag het informatie-technologisch aspect niet ontbreken. De heer Sjef Ruhe heeft ons in het bijzonder aangespoord om met dit project aan de slag te gaan.

D.2.3 Contactpersonen

Ir. Sjef Ruhe (06-53.60.12.56) zal ons gedurende dit project begeleiden en is de contactpersoon voor de universiteit.

D.2.4 Probleemstelling

Bedrijven zijn niet in staat om tekeningen te beheren die ze aangeleverd krijgen bij nieuwbouw of verbouw van met name bedrijsgebouwen. Op het moment dat een aanpassing nodig is, kan een bedrijf vaak geen tekening(en) van de huidige situatie aanleveren. Dit zorgt voor onnodige kosten en tijdsverlies.

D.2.5 Doelstelling

Ontwerp een document beheer systeem wat eenvoudig in gebruik is zonder dat de klant technische kennis van zijn gebouwen of versiebeheer hoeft te hebben.

D.2.6 Opdrachtformulering

Het ontwerpen van een document beheer systeem dat eenvoudig in gebruik is zonder dat de klant technische kennis van zijn gebouwen of versiebeheer hoeft te hebben.

D.2.7 Op te leveren producten

Bovengenoemde systeem, waarbij naast eerdergenoemde aspecten, de volgende kwalitatieve eisen van belang zijn: makkelijk in onderhoud, robuust, veilig, ook te begrijpen voor niet-technici, enz.

D.2.8 Randvoorwaarden

Het systeem moet passen binnen de huidige infrastructuur van het bedrijf .

D.2.9 Risicofactoren

Door onze release early, release often strategie sluipen er zo nu en dan fouten in de productie omgeving. Omdat deze fouten zo snel mogelijk opgelost moeten worden kan dit leiden tot het uitlopen van de planning en mogelijke irritaties bij gebruikers.

D.3 Aanpak

D.3.1 Inleiding

In dit hoofdstuk zullen wij onze aanpak van het project toelichten.

D.3.2 Methodiek

We hebben gekozen voor een *agile development*methodiek, namelijk de *scrum*aanpak. Hierbij definiëren we het te behalen doel voor een korte periode (over het algemeen proberen we aan het eind van de week een nieuwe feature release te houden). Na het doel te hebben vastgesteld houden we een korte discussie over de implementatie, mogelijke problemen en zetten kort op papier wat we nodig denken te hebben aan klassen en functionaliteit.
Over het algemeen implementeren we de functionaliteit in de volgende volgorde:

1. Data model (over het algemeen op basis van data uit een RDBMS tabel)
2. Overview/List functionaliteit in de controller
3. Toevoegen van data
4. Data bewerken
5. Data verwijderen
6. Betere weergave, web-2.0 functionaliteit

Door voor de scrum methodiek te kiezen kunnen we regelmatig nieuwe features demonstreren en feedback verwerken. DENC wil graag continu de voortgang volgen en indien nodig bijsturen, waardoor het voor ons belangrijk is om een dergelijke methode te gebruiken.

D.3.3 Technieken

We maken met name gebruik van *release early, release often.*Dit houdt in dat als we functionaliteit geïmplementeerd hebben, we deze functionaliteit in de productie omgeving beschikbaar maken. In sommige gevallen heeft dit als gevolg dat er fouten gevonden worden tijdens gebruik van de productie omgeving, die op dat moment zo snel mogelijk opgelost moeten worden.

D.3.4 Planning

Vanwege onze scrum en release early, release often aanpak is het mogelijk om per week te beslissen welke functionaliteit toegevoegd moet worden aan het systeem. Vaak krijgen we ook aan het eind of begin van de week te horen wat er van ons verwacht wordt, dus als we dan zelf al in gedachten hadden wat er in de volgende release moet komen moeten we dat soms bijstellen. We plannen dus maximaal een week of twee weken (in geval van incremental features of dependant features) vooruit, waarbij we op verzoek de planning bijstellen.

D.4 Projectinrichting

D.4.1 Inleiding

In dit hoofdstuk zetten wij uiteen hoe het project is ingericht

D.4.2 Informatie

Voor het uitwisselen van informatie met de opdrachtgever gebruiken we de wekelijkse bijeenkomst.

D.4.3 Faciliteiten

Voor de uitvoering van onze opdracht kunnen we onder andere gebruik maken van de volgende faciliteiten van DENC:

Werkplekken De DENC vestiging op het Forepark in Den Haag biedt werkplekken en afgesloten ruimtes voor vergaderen of brainstormen.

Computers Op de werkplekken zijn moderne computers beschikbaar.

Software Op de computers hebben we de beschikking over Ubuntu Linux. Extra software is direct te installeren. Windows (XP) is ook beschikbaar om mee te testen, maar vanuit de Linux omgeving is het ook mogelijk om virtueel een Windows installatie te draaien.

Servers Er is toegang tot een co-located server met Ubuntu Linux. Dit is de server waarop het systeem uiteindelijk geplaatst wordt.

D.5 Kwaliteitsborging

D.5.1 De kwaliteit

De kwaliteit van het document beheer systeem wordt gemeten aan de hand van de beschikbare functionaliteit. Het is voor DENC uitermate belangrijk dat de integriteit van de documenten bewaard blijft, maar dit wordt gezien als vanzelfsprekend. In eerste instantie bleek ook dat we de integriteit te restrictief bewaakten, waardoor we om onze eigen bescherming heen moesten werken om bepaalde handelingen uit te voeren, zoals het maken van previews. Omdat we nooit een bestand verwijderen (verwijderen in het systeem is verplaatsen naar een virtuele prullenbak) blijven de documenten altijd bewaard. Vanwege de regelmatige evaluatie van DENC weten we altijd of een probleem goed opgelost is en kunnen we er voor zorgen dat de ervaren kwaliteit hoog blijft.

D.5.2 Documentatie

Voor het maken van de handgeschreven documentatie gebruiken we TEX. Documentatie van geschreven code wordt gedaan in de code zelf. Hiervoor maken we gebruik van de *phpDocumenter* syntax, die ook door het Zend Framework wordt gebruikt.

D.5.3 Versiebeheer

Zowel bij het documenteren als ontwikkelen maken we gebruik van versiebeheer op basis van *Subversion*.We gebruiken hiervoor twee verschillende repositories, zodat code en documentatie netjes gescheiden blijven.

Over het algemeen plaatsen we alleen een nieuwe versie in de repository als deze geen regressies bevat.

D.5.4 Evaluatie

Omdat we de beschikking hebben over werkplekken in een vestiging van DENC vind er wekelijks een korte evaluatie plaats. Daarnaast is er maandelijks een telefonisch IT-overleg waarbij de voortgang ook besproken wordt. Er is ook al een kleine presentatie geweest voor de directie van DENC, waarbij het idee en de uitvoering zijn toegelicht.

D.5.5 De pilots

Als de basis functionaliteit beschikbaar is zal het systeem intern gebruikt gaan worden voor het beheer van de volgende documenten:

- Standaard projectdocumenten
- Documenten van P&O (personeel en organisatie)
- Declaraties (reiskosten, lunchvergoeding)

Van deze documenten is gebleken dat het moeilijk is om één definitieve versie vast te leggen, het document beheer systeem zal hier dus mee getest gaan worden.

Bijlage E

Requirements Analysis Document

E.1 Introductie

Het Requirements Analysis Document behandelt de problemen die we verwachten tegen te komen en hoe we deze problemen op gaan lossen. Het behandelt wie er gebruik maakt van het systeem, hoe er gebruik gemaakt gaat worden van het systeem en eventuele eisen die niet direct gerelateerd zijn aan de functionaliteit van het systeem.
We beginnen met een introductie tot het huidige systeem waarna we het voorgestelde systeem behandelen.

E.2 Current System

Er zijn twee huidige systemen die we kunnen vervangen door het document beheer systeem. Van beide systemen zullen we de problemen opsommen die het document beheer systeem kan oplossen.

E.2.1 Hardcopy systeem

Als DENC een project afrond voor een klant, krijgt deze klant een kopie van alle tekeningen. Dit zijn zowel papieren kopiën als een digitale kopie op CD. Bij de klant aangekomen, verdwijnen deze documenten over het algemeen in het archief. Iemand binnen het bedrijf van de klant beheerd het alsmaar groeiende archief, maar op het moment dat bijvoorbeeld bouwtekeningen nodig zijn kunnen ze niet meer worden gevonden.

Nadelen van het hardcopy systeem

- Klant moet actie ondernemen om archief te beheren
- Er is geen centrale locatie waar iedere klant zijn documenten kan vinden
- Als het document niet gevonden kan worden moet aan DENC verzocht worden om de documenten opnieuw te versturen
- Beheer van het archief is niet gericht op de specifieke documenten die DENC aanlevert.

E.2.2 Pseudo-managed systeem

Een bedrijf beheert zijn eigen archief van standaard documenten. Bij DENC is er bijvoorbeeld een digitaal archief van standaard contracten, standaard fasedocumenten voor projecten en nog veel meer. Het beheer wordt gedaan door één persoon of een kleine groep personen binnen het bedrijf. Beheer gebeurd op basis van een mappenstructuur waarin bestanden worden geplaatst. Al snel worden mappen genummerd, zodat ze makkelijker te vinden zijn.
Op een gegeven moment wordt het complete archief gekopiëerd naar een andere locatie zodat ook een andere vestiging kan profiteren van alle data in het archief. Op de andere vestiging worden aanpassingen gemaakt, maar deze worden niet teruggekoppeld naar de eerste vestiging. Na verloop van tijd ontstaan er zo verschillen tussen de archieven. Er komen nieuwe en oude versies, allemaal in eigen mapjes.
Op dit moment zoekt iemand een document. Omdat het archief onoverzichtelijk is geworden, weet hij niet waar hij moet zoeken. Dit kost tijd omdat de persoon zelf op onderzoek uitgaat, of iemand anders moet vragen waar de documenten zijn en kan er voor zorgen dat een verkeerd document wordt gebruikt.

Nadelen van het Pseudo-managed systeem

- Beheer op basis van een generiek systeem wat niet gericht is op documentbeheer
- Inefficiënt beheer door gebrek aan vaardigheden bij beheerders
- Geen centraal beheer van documenten
- Beheer van het archief is niet gericht op specifieke documenten

E.3 Proposed System

E.3.1 Overview

E.3.2 Functional Requirements

Actoren

In het beginsel onderscheiden we vier actoren:

- Beheerder
- Projectmanager
- Medewerker
- Klant

Beheerder

De beheerder is de gebruiker met de meeste rechten in het systeem. Een beheerder kan:

- Gebruikers toevoegen/verwijderen
- Projecten toevoegen/verwijderen
- Rechten toekennen aan andere gebruikers

Projectmanager

De projectmanager is iedereen die eindverantwoordelijk is voor een project. Een projectmanager kan alles wat een medewerker kan en bovendien ook:

- Medewerkers aan een project koppelen
- Medewerkers rechten geven binnen een project
- Documenten van status veranderen

Medewerker

De medewerker is iedereen die werkzaam is bij DENC. Een medewerker kan:

- Documenten wijzigen

Klant

Een klant is iedereen die niet werkzaam is bij DENC, maar wel een relatie heeft met het project. Een klant kan:

- Documenten inzien
- Feedback geven op documenten

E.3.3 Nonfunctional Requirements

User interface en mensenlijke factoren

De interface moet voor iedereen met een recente browser te gebruiken zijn. De interface moet zo simpel en intuïtief mogelijk zijn.

Documentatie

Er zal een korte handleiding aan de klant aangeboden worden met daarin stapsgewijs de belangrijkste taken uitgelegd.

Software overwegingen

Gebruikers dienen te beschikken over een browser die ten minste een subset van XHTML en CSS level 2 ondersteund. Afhankelijk van de gewenste actie met een document kan een PDF-reader nodig zijn.

Prestatie eigenschappen

Het systeem moet voor de gebruikers snel en responsief aanvoelen, dit betekent dat het beantwoorden van een request nooit langer dan vijf seconden mag duren in specifieke gevallen, en nooit meer dan een seconde in het algemeen.

Foutafhandeling in randgevallen

Indien mogelijk wordt de klant aangeboden om de actie te herhalen. Mocht een klant verbinding met de server verliezen, dan is er geen andere mogelijkheid om het systeem te gebruiken.

Systeeminterface

Het systeem werkt samen met de bestaande lijst van medewerkers (uit LDAP). Voor het converteren van documentformaten dient het systeem een RESTful interface te bieden.

Kwaliteitseisen

De integriteit van documenten en de integriteit van de rechten op documenten moet bewaard en bewaakt blijven.

Beveiligingseisen

Het systeem moet documenten beveiligen tegen ongeauthoriseerde toegang. De verbinding met het systeem moet hierom ook beveiligd zijn met een ondertekend certificaat.

Resources en management issues

Het systeem moet gebruikers zo snel mogelijk op de hoogste stellen van wijzigingen in documenten. Daarnaast moet het systeem bewaken dat wijzigingen niet onbehandeld blijven.

Autonomiteit

Het systeem moet rechten toekennen aan nieuwe documenten zonder dat de gebruiker hier expliciet opdracht voor geeft. Previews en mogelijke andere conversies van bestanden moet gemaakt worden zonder dat hier extra interactie voor nodig is.

E.3.4 Constraints

Project constraints

De opgelegde beperkingen komen voort uit de applicaties die al in gebruik zijn op het bedrijfsnetwerk van DENC. Hier wordt gebruik gemaakt van PHP en MySQL, dus zullen wij ook gebruik moeten maken van PHP en MySQL. Dit is nodig zodat de applicatie binnen de huidige infrastructuur gehost kan worden en zodat er mogelijke integratie plaats kan vinden.

E.3.5 System Models

Scenarios

Actor: Beheerder Gebruiker toevoegen
DENC Netherlands B.V. heeft de opdracht aangenomen om een nieuw magazijn te bouw voor Piet's Pizza's B.V.. De beheerder maakt een klant aan zodat de directeur van Piet's Pizza's B.V. kan inloggen om de tekeningen te bekijken.

Project toevoegen
DENC Netherlands B.V. heeft vernomen dat Piet's Pizza's B.V. mogelijk een nieuw magazijn wil laten bouwen. Daarom wordt een project aangemaakt zodat er tekeningen in geplaatst kunnen worden.

Rechten toekennen
Voor Piet's Pizza's nieuwe magazijn wordt samengewerkt met Cor de constructeur, die reeds aan een ander project heeft gewerkt voor DENC. Cor heeft al inloggevens, maar heeft nog geen rechten op het project voor het nieuwe magazijn. De beheerder geeft Cor rechten zodat hij alle tekeningen kan zien.

Actor: Projectmanager Medewerkers aan een project koppelen
Tom de tekenaar gaat het magazijn voor Piet's Pizza's B.V. tekenen. Hierom wordt Tom gekoppeld aan het project, zodat hij nieuwe tekeningen toe kan voegen aan het project.

Medewerkers rechten geven binnen een project
Tom de tekenaar maakt een zooitje van de tekeningen die hij bij projecten plaatst. De projectmanager

beslist dat alle tekeningen van Tom nu via hem geplaatst zullen worden. De rechten om bestanden te plaatsen in de tekeningenmap worden uitgezet voor Tom.

Verander status van een document

Tom de tekenaar heeft een nieuwe versie van de tekening van de voorgevel gemaakt. De tekening staat in het project. Mark de projectmanager verandert de status van de tekening zodat de klant deze ook kan bekijken.

Actor: Medewerker Wijzig een document

Tom de tekenaar moet een nieuwe wc tekenen voor Piet's Pizza's B.V.. Hij download de bestaande tekening, maakt wijzigingen waar nodig en plaatst de nieuwe versie in het project met de opmerking dat de wc is getekend.

Actor: Klant Bekijk een document

Tom heeft een nieuwe wc getekend, Piet is daar van op de hoogte gesteld. Piet bekijkt online de nieuwe tekening om te kijken of hij de nieuwe wc mooi vind.

Geef feedback op een document

Piet vind de wc nog niet zo mooi. Hij plaatst feedback dat de wasbak nog wat dichter naar de deur moet en dat de toiletrol-houder niet aan het plafond moet.

Use Cases

Inloggen in het systeem
Precondition: The user has an account.
Postconditie: THe user is logged in succesfully.
Scenario: The users starts using the system
Actor: User

User actions	System response
1. The user enteres the web address in his browser.	2. The browser shows the login page.
3. The users enteres his username / password and presses "login".	4. The system directs the user to the main menu / welcome page.

Tabel E.1: Use Case – Login

Bijlage F

Architectural Design Document

F.1 Purpose

In dit document beschrijven wij een aantal design issues, waar wij per probleem de relevante alternativen en onze keuze zullen presenteren. Het gaat hier voornamelijk om keuzes waar in het Requirements Document niet een evidente winnaar naar voren komt. Bij het afwegen van de beste keuze zullen zowel technische als organizatorische zaken megewogen worden. Het hoofdstuk Generieke prioriteiten zal verden op de prioriteiten ingaan.

Nu volgt een opsomming van de design issues.

- Access Control's
- Storage Backend
- User/Password Backends
- File convertion architecture

F.2 General priorities - design goals

De doelen die wij gesteld hebben voor het ontwerp zijn te verdelen in de volgende catagorieen.

- Betrouwbaarheid
- Veiligheid
- Bruikbaarheid
- Onderhoudbaarheid
- Volledigheid

F.2.1 Betrouwbaarheid

De betrouwbaarheid van het systeem is een belangrijk aspect aangezien de informatie die er in opgeslagen wordt een niet verwaarloosbare economische waarde heeft. Ook omdat het systeem een relatief lange tijd mee moet kunnen gaan is de betrouwbaarheid van belang.

F.2.2 Veiligheid

De gegevens die in het systeem opgeslagen zijn moeten veilig bewaard worden en alleen toegangkelijk zijn voor voor die gebruikers die expliciet of impliciet toegang gekregen hebben.

F.2.3 Bruikbaarheid

Het systeem is bovenal een hulpmiddel om een taak te verichten en in die hoedanigheid komt functie en het gemak waarmee het zijn functie diend hoog op de lijst van prioriteiten.

F.2.4 Onderhoudbaarheid

Omdat informatie systemen vaak een groot deel van de totale cost of owner ship in hun onderhouds periode verbruiken diend dit extra aantacht te krijgen.

F.2.5 Volledigheid

Het systeem ontleend zijn nut aan het vermogen om de complete set functies te herbergen en als zodanig is het van het uiterste belang dat het systeem de volledige set minimal eisen implementeerd.

F.3 Outline of the design

Het doel van het systeem is om klanten een eenvoudig en overzichtelijk bestandsbeheer te bieden, met de nadruk op het beheer van tekeningen. Hierbij is er de mogelijkheid om willekeurig wat voor bestanden dan ook te beheren en de mogelijkheid om speciale acties te verrichten met tekeningen.

F.3.1 Beheer van bestanden

Het systeem maakt het mogelijk om eenvoudig bestanden te beheren. Voor de eindgebruiker moet het systeem aanvoelen als een lokale file browser. Omdat het beheer door DENC gedaan wordt, kan de klant zich zeker voelen over de integriteit van zijn data.

Personeel van DENC kan nieuwe bestanden plaatsen in mappen die door de klant ingezien kunnen worden. Een klant kan daar opmerkingen bij plaatsen, bijvoorbeeld verzoeken voor wijzigingen.

Bestanden kunnen op twee manieren gevonden worden: door te bladeren, of door te zoeken. Het bladeren naar bestanden biedt een vergelijkbare functionaliteit met een file browser op een lokaal systeem. Zoeken maakt bladeren vaak overbodig, omdat zo veel sneller het juiste bestand te vinden is. Bij een bestand kan metadata geplaatst worden zodat het bestand makkelijk terug te vinden is vanuit de zoekfunctie.

F.3.2 Beheer van tekeningen

Het beheer van tekeningen is grotendeels gelijk aan het beheer van bestanden. Een tekening is gewoon een bestand, maar tegelijk betekent een tekening veel meer. Omdat een tekening in een gesloten formaat is, is er speciale software nodig om deze te kunnen openen.

Door tekeningen in het systeem om te zetten naar een open formaat, of andere versies van het gesloten formaat die vaker geopend kunnen worden, maken we het de klant makkelijker om zelf aan de slag te gaan met zijn tekeningen. Daarnaast is het ook belangrijk dat mensen een preview van een tekening kunnen zien zonder dat ze het bestand hoeven te downloaden, ook daar is omzetten dus van belang.

Van een tekening moeten ook afdrukken besteld kunnen worden, met een tekening kan dus fysiek meer dan met een willekeurig ander bestand.

F.4 Major design issues

F.4.1 Opslaan van bestanden

Opslaan van bestanden staat centraal in het systeem, het is dus belangrijk om dit efficient te doen zodat er niet later nog een grote conversie-stap gemaakt hoeft te worden.

Direct op schijf

In eerste instantie lijkt het het makkelijkst om de bestanden direct op de schijf op te slaan en een soort web-based file-browser te maken voor de bestanden op de schijf. Met behulp van Access Control Lists (ACL) op niveau van het bestandssysteem kan toegang echter niet afgeregeld worden, dus dit moet alsnog op het applicatie-niveau geregeld worden. Een verwijzing naar een bestand kan alleen op basis van het volledige pad, het bijhouden van verschillende versies van het bestand is echter praktisch onmogelijk. Omdat er veel extra data bij bestanden komt moet er vaak naar een bestand verwezen worden. Deze verwijzingen moeten bijgehouden worden als het bestand wijzigt. Ook op het bestandssysteem kan al enige extra data worden bijgehouden, helaas is dit echter niet efficient doorzoekbaar.

Alles in de database

Een andere optie is om alles in de database op te slaan. Bestanden worden dan in de volledigheid in een kolom in de database geplaatst. Vrijwel elk recent RDBMS heeft hier ondersteuning voor. Een groot nadeel is echter dat een RDBMS hier niet voor gemaakt is. Tabellen worden onnodig groot, waardoor de prestaties achteruit gaan. Omdat de performance van de database niet is afgesteld op zulke grote tabellen kan het RDBMS niet efficient cachen en wordt de performance nog slechter. Daarnaast maakt het opslaan in de database het moeilijker om conversies op bestanden uit te voeren. Hiervoor moet het bestand dan eerst tijdelijk worden opgeslagen, en pas daarna kan er iets mee gedaan worden.

HashStore

De optie die uiteindelijk deel uitmaakt van het systeem is een combinatie van de bovengenoemde oplossingen. Zo veel mogelijk informatie wordt opgeslagen in de database, maar het bestand zelf wordt opgeslagen op het bestandssysteem. Zo maken we efficïent gebruik van de database en van het bestandssysteem. Om een unieke naam te hebben voor bestanden op de schijf maken we gebruik van een checksum van de inhoud van het bestand. Hiervoor wordt het SHA-256 algoritme, omdat de kans op gelijke checksums hierbij heel klein is. Door de structuur van veel bestandsformaten wordt de kans op gelijke checksums nog kleiner. Het systeem gaat er dus wel van uit dat als twee bestanden dezelfde checksum hebben, dat ze dan ook dezelfde inhoud hebben, maar we verwachten niet dat dit problemen op zal leveren.

Doordat we gebruik maken van de checksum om de locatie van het bestand te bepalen kunnen we ook een splitsing maken voor het opslaan van bestanden. Dit kan in de toekomst nog een uitkomst bieden als we meer data zouden hebben dan we op één bestandssysteem weg zouden kunnen schrijven.

F.4.2 Beheer van gebruikers

Zonder gebruikers is het systeem waardeloos, dus een lijst van gebruikers is erg belangrijk. We hebben alleen te maken met verschillende soorten gebruikers, waarvan er een aantal al in onze lokale directory tree staan. Het is belangrijk dat we hier goed mee om gaan, dus hier moest over nagedacht worden.

Alle gebruikers in de directory tree

Het interne netwerk van DENC maakt gebruik van LDAP om haar gebruikers bij te houden. Dit biedt een groot voordeel omdat alle andere software direct of indirect gebruikers uit LDAP kan uitlezen. Als een gebruiker in LDAP bestaat, kan die gebruiker dus inloggen op de computers, mail en andere bedrijfsapplicaties.

Een optie zou dus zijn om alle gebruikers in de directory tree te stoppen. Het nadeel zou echter zijn dat dan in de directory tree en betere scheiding gemaakt zou moeten worden tussen DENC medewerkers en klanten die buiten het document beheer systeem geen enkele relatie met DENC hebben. Omdat we de directory tree eenvoudig willen houden viel deze oplossing dus sowieso al af.

Alles in de database

Een andere oplossing is om alle gebruikers in de database te stoppen. Het nadeel is hier dat er sprake is van duplicatie tussen de DENC medewerkers in LDAP en DENC medewerkers die inloggen op het systeem. Hoe nadelig dit ook is, hebben we toch besloten om voor deze aanpak te gaan. Er is echter een klein verschil tussen de manier waarop DENC medewerkers door het systeem ingelogd worden ten opzichte van de andere gebruikers. Voor DENC medewerkers wordt de wachtwoord verificatie gedaan tegen de LDAP server en niet tegen de database. Als een DENC medewerker nog niet eerder is ingelogd, maar wel bekend is in LDAP dan wordt zijn/haar account aangemaakt in de database. Het enige probleem wat hier dan overblijft is dat aanpassingen in LDAP niet automatisch doorgevoerd

worden naar de database, dus als een naam wijzigt blijft het systeem de oude naam gebruiken tot de wijziging handmatig ook in de database wordt doorgevoerd.

F.4.3 Converteren van documenten

Het converteren van documenten is vooral belangrijk bij tekeningen, maar ook voor andere bestandsformaten is het belangrijk dat er op z'n minst een voorbeeld beschikbaar is. We hebben een aantal methoden overwogen om de conversie uit te voeren.

Direct na uploaden van bestand

Als het bestand wordt geupload wordt direct een preview gemaakt. Als er meer conversies nodig zijn worden deze ook direct gemaakt. Het nadeel is dat de server onnodig lang kan bezig zijn met het maken van een preview en daardoor minder requests van andere gebruikers kan afhandelen. Daarnaast kan het ook echt lang duren voor een conversie van een tekening klaar is, dus hier kan een gebruiker toch niet op wachten.

Converteren op tijdsbasis

Een andere optie is om zo nu en dan te kijken of er nog conversies gedaan moeten worden en ze dan uit te voeren. Zo kan de conversie op de achtergrond uitgevoerd worden zonder dat de gebruiker hier last van heeft. Met wat extra werk is dit zelfs uit te breiden tot een gedistribueerd systeem, de server hoeft dan dus niet meer het zware werk te verrichten en kan gewoon requests blijven afhandelen.

Combinatie van beiden

Uiteindelijk hebben we besloten om een combinatie van beide oplossingen te gebruiken. Voor bestandstypes waarvan we weten dat een preview snel gemaakt kan worden, wordt dit meteen gedaan. Voor vrijwel alle andere bestandstypes draait er een daemon die zo nu en dan controleert of er nog conversies gedaan moeten worden. Moeten deze gedaan worden, dan wordt het bestand gedownload, inclusief de conversie-instructies, waarna de daemon dit uitvoert en de geconverteerde versies terug plaatst.

F.5 Other details of the design

Er zijn nog een aantal andere details die meerwaarde aan het systeem geven, terwijl deze niet noodzakelijk zijn voor de correcte werking van het systeem. De volgende opsomming geeft een aantal van deze details, zonder in te gaan op specifieke implementaties.

- Gebaseerd op open standaarden
- Command-line beheer en verificatie
- Integratie met third-party software (converters, virusscanners)

Bijlage G

MoSCoW

G.1 Overview

Dit document probeert een ordening aan te geven in de relative prioriteit die verschillende functionaliteiten ten opzichte van elkaar hebben. Het maakt hiertoe vier catagorieen.

- Must have Items in deze catagorie moeten geimplemnteerd worden om aan de meest basale functionele omschrijving te voldoen.
- Should have Items in deze catagorie moeten eigenlijk geimplemnteerd worden om het project als een succes te kunnen zien
- Could have Items uit deze catagorie hebben ofwel een grootte toegevoegde waarde ten opzichte van de kosten of een groote synergie met items van een hogere catagorie.
- Would have Items uit deze catagorie worden in commerciele projecten alleen geimplementeerd als een een overschot aan resources is, onder dit kopje vallen vaak de meest wilde/moeilijke ideeen.

G.2 Must have

- Beheer van mappenstructuur
  Mappen : maken, verwijderen, hernoemen, verplaatsen.
- Beheer van individuele documenten
  Bestanden : maken (uploaden), verwijderen, hernoemen, verplaatsen.
- Access Control List (ACL)
  Object gebonden hierarchische lijst van rechten.
  Deze lijst moet bekeken en gewijzigd kunnen worden en de rechten dienen afgedwongen te worden.
- Gebruikersbeheer
  Gebruikers : bekijken, maken, verwijderen, bewerken.

- Hash store backend (voor opslag bestanden)
  Objecten (bestanden) opslaan aan de hand van een hash om de inhoud te kunnen valideren (verifiëren).
  Maakt verspreide opslag mogelijk voor de toekomst.

G.3 Should have

- Conversie tussen bestandstypes/versies
  Tekeningen omzetten naar oudere AutoCAD versies voor externe partijen met oudere AutoCAD versies.
  Tekeningen omzetten naar PDF voor klanten.

- Archive (zip, tar) download
  Een set van bestanden en of mappen in een keer downloaden door deze in het systeem 'on-the-fly' in een archief te stoppen.

- Actieve foutmeldingen
  Het systeem moet in het geval van een fout deze opslaan en afhankelijk van de ernst van de fout de beheerder(s) informeren via SMS, XMPP of E-mail.

- Meertalige interface
  Het systeem meot vertaalbaar zijn en afhankelijk van de bezoeker de gewenste taal gebruiken.

- Opvragen/verwerken van metadata
  Het systeem moet, voor zover de bestandstypen dat toestaan, de metadata uit de bestanden extraheren en deze voor de gebruiker doorzoekbaar aanbieden. Een voorbeeld hiervan is dat foto's steeds vaker geografische informatie bevatten die afzonderlijk opgeslagen kan worden.

G.4 Could have

- Uploaden bestanden uit archief
  Bij het aanbieden van grote hoeveelheden bestanden aan het systeem is het voor de gebruiker wenselijk dit niet per bestand te hoeven doen. Massupload / zip-upload functionaliteit biedt hiervoor een uitkomst.

- Handmatige metadata invoer
  Waar metadata niet automatisch ingevuld kan worden moet de gebruiker de mogelijkheid hebben deze alsnog in te vullen.

- Interactive interface (WEB 2.0/AJAX)
  Een interactieve interface kan het verschil maken tussen een goede applicatie of een matige website. Door gebruik te maken van moderne technieken biedt een website bijna alle voordelen van een lokale applicatie.

- Bestellen van afdrukken van tekeningen
  In het specifieke gaval van digitale tekeingen is het op een makkelijke manier verkrijgen van een papiere kopie een meerwaarde die het systeem zou kunnen bieden.

G.5 Would have

- Tekeningen splitsen in lagen
  Tekeningen (met name AutoCad bestanden) bevatten vaak lagen. De informatie van de ene laag is dan relevant voor actor 1 terwijl de informatie van een andere laag relevant is voor een andere actor. Het zou mooi zijn als het systeem tekeneingen ook de informatie van een eventuele subset van lagen kan verwijderen tijdens het aanbieden. Dit kan tot een efficiënte en overzichtelijke representatie van de benodigde informatie leiden.

- Online aanpassen tekeningen
  Klanten weten zelf het beste wat ze willen. Als ze online de gewenste wijzigingen aan een tekening kunnen maken is voor DENC direct duidelijk wat de klant wil. Dit vergemakkelijkt de communicatie en geeft de klant het gevoel dat hij zelf bijdraagt aan de projectvoortgang.

- Online viewer tekeningen (ala Google Maps)
  Mogelijkheid om online tekeningen te bekijken met de optie om individuele lagen aan of uit te zetten, in te zoomen of uit te zoomen en verwijzingen naar andere tekeningen te openen.

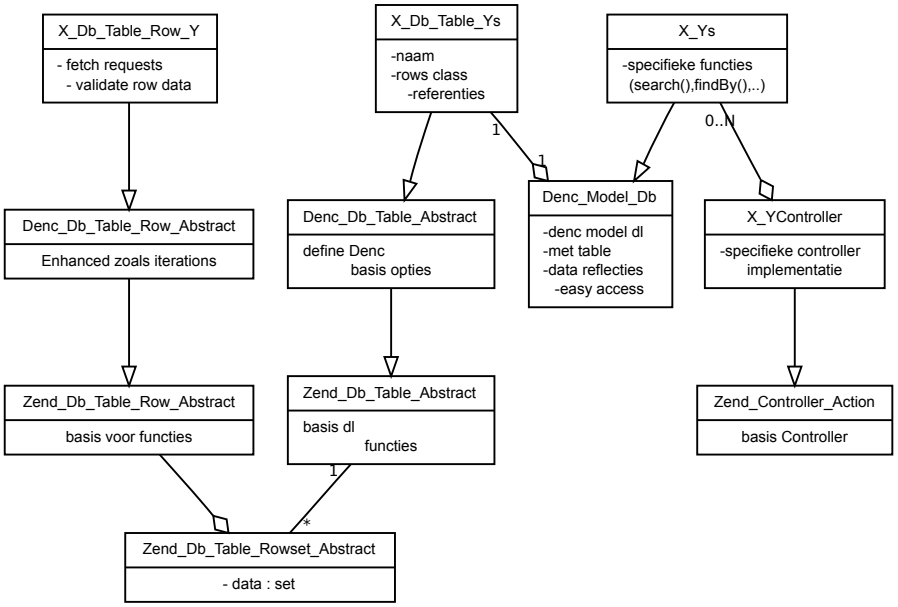
- Ondersteuning voor andere protocollen (FTP, WebDAV)
  Om gebruikers de mogelijkheid te geven om via een andere interface gebruik te kunnen maken van het document beheer systeem is het van belang om andere toegangsmethoden te implementeren.

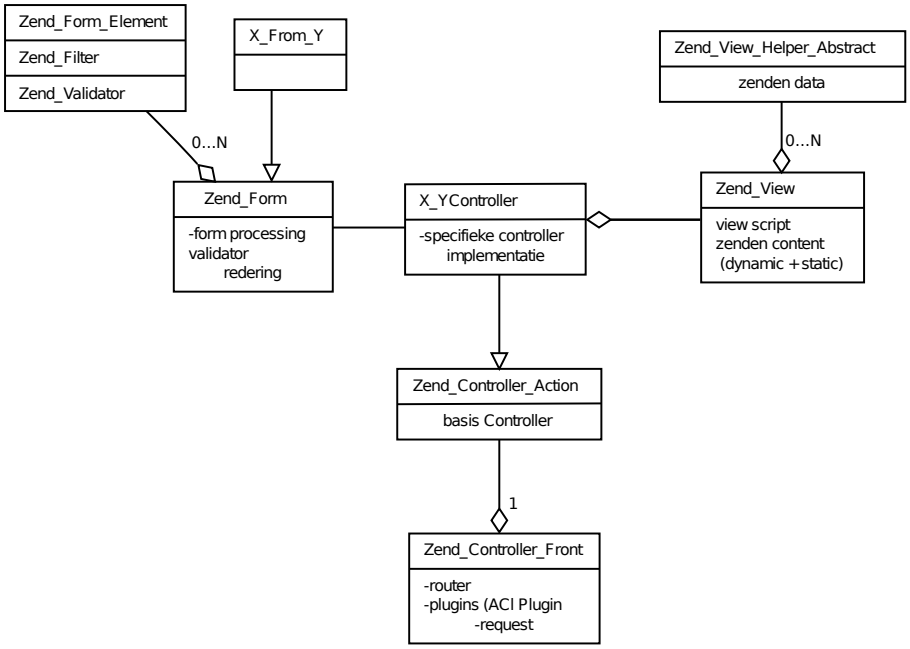
- Integratie met AutoCAD
  De mogelijkheid om in AutoCAD bestanden uit het systeem te openen en weer op te slaan met opmerkingen.

- Verifieren integriteit bestanden
  De mogelijkheid om te controleren of de inhoud van het bestand nog gelijk is aan wat de inhoud was bij het toevoegen van het bestand.

Bijlage H

Diagrams




Zend_Form_Element
Zend_Filter
Zend_Validator
X_From_Y
Zend_View_Helper_Abstract
zenden data
0...N
0...N
Zend_Form
-form processing
validator
redering
X_YController
-specifieke controller
implementatie
Zend_View
view script
zenden content
(dynamic + static)
Zend_Controller_Action
basis Controller
1
Zend_Controller_Front
-router
-plugins (ACl Plugin
-request