Technische Universiteit Delft
Technische Informatica
Software Technologie
Faculteit der Elektrotechniek, Wiskunde en Informatica

Eindverslag IN3405

Bachelor Project Software Technologie
TAG eForms

Commissie:
ir. B.R. Sodoyer
R. ten Oever MBA

maandag 20 april 2009

Vincent Simon van Beek 1217348
Mark Zijdemans 1217771

Inhoudsopgave

1. Voorwoord..... 6

2. Samenvatting.....8

3. Inleiding.....10

3.1 Reader.....10

4. Bedrijf.....11

5. Probleem/opdracht.....12

5.1 TAG eForms®.....12

5.2 Opdracht.....13

6. Aanpak..... 14

6.1 hoofdpunten plan van aanpak.....14

6.2 project team.....14

6.3 rollen.....14

6.4 planning methoden.....15

6.5 project fasen.....15

6.6 documenten.....17

7. Design..... 19

7.1 inleiding.....19

7.2 Requirements.....19

7.3 PHP framework.....20

7.4 JavaScript (JS).....21

7.5 Database ontwerp.....22

7.6 Formbuilder.....23

7.6.1 gebruiksvriendelijkheid en drag en drop.....23

7.6.2 formulieren inzichtelijk maken na wijziging van het formulier.....24

7.6.3 gevaar dataverlies client side.....24

7.7 Workflowbuilder.....25

7.8 Message Box.....26

8. Software ontwikkeling.....29

8.1 Ontwikkel omgeving.....29

8.2 Pair programming.....29

8.3 Unit testen.....30

9. Discussie.....31

10. Conclusie.....33

11. Recommendation.....34

12. Reflectie.....35

13. Merknamen.....37

14. Appendix.....38

A ontwerp.....38

A.1 Plan van aanpak.....38

A.2 Requirement definition.....52

A.3 Systeem eisen.....55

A.4 Class diagram.....56

A.5 Extended Entity-Relationship diagram.....57

B PHP skeleton.....59

C XP.....62

D Voorbeeld Unit Test resultaat.....63

E.Engelbart's Unfinished Revolution [5].....64

F. Artikel over Comet.....66

G. Artikel Onderzoek naar de toepasbaarheid van RUP en MoSCoW voor kleine software development projecten.....72

H. Artikel Adeptive user interface in web application.....81

15. Bronnen.....87

1. Voorwoord

Dit document omvat een stage verslag van Vincent van Beek en Mark Zijdemans bij het bedrijf TAG software. Zowel Vincent als Mark studeren technische informatica aan de faculteit EWI van de Technische Universiteit Delft. In dit document zullen wij uiteenzetten welke beslissingen we hebben moeten maken en wat de uitkomsten van deze beslissingen waren. TAG is een software bedrijf dat zich gespecialiseerd heeft in kwaliteitscontrole software.

Documenten binnen een bedrijf zijn altijd een oplossing voor kleine problemen. Maar deze documenten veroorzaken zelf ook veel problemen welke opgelost kunnen worden. Een van deze problemen die TAG eForms tekkelt zijn:

- milieu/afval probleem
  denk aan oud papier, maar ook het maken van papier is milieu belastend. Zien we niet steeds vaker onder aan een mailtje staan: "denk aan het milieu voordat u deze print".
- archivering
  Als 100 medewerkers elke week hun uren moeten doorgeven. Zijn dit minimaal 4000 documenten per jaar. Dit kan zomaar een stapel papier zijn van 40cm. Digitale informatie neemt veel minder ruimte in.
- uniformiteit
  Het vergt discipline om alle documenten binnen een bedrijf uniform te houden, een standaard template uit TAG eForms helpt hierbij
- workflow
  Naar wie moet een document? TAG eForms koppelt een document onlosmakelijk aan een workflow.

Wij bedanken de heer Sodoyer voor de academische begeleiding rondom dit project. De heer Sodoyer zorgde ervoor dat wij bleven nadenken over de situaties en issues vanuit een academische invalshoek.
Verder willen wij de heer Ten Oever bedanken voor de begeleiding op bedrijfskundig niveau. De heer Ten Oever zorgde ervoor dat de ideeën en onderdelen altijd een doel hadden dat betrekking had tot de wensen van toekomstige gebruikers.
Deze twee mensen zorgden er zo voor dat het product nuttig werd en daarnaast goed gemaakt werd.
Tot slot willen wij de medewerkers van de IT afdeling van TAG software bedanken. Welke belangrijke rollen gespeeld hebben tijdens het ontwikkel proces van het prototype. Om te beginnen Danny Hoogendoorn voor het ontwerpen van de meeste grafische onderdelen. Michael van Rijswijk voor het testen als zijnde een gebruiker en zijn kennis op het gebied van gebruikersproblemen. Emiel van Goor voor het aanreiken van mogelijkheden ten tijden van keuzes voor oplossingen en producten van derden.

Vincent van Beek
Mark Zijdemans
Dordrecht 2008

2. Samenvatting

Tijdens het stage project TAG eForms is een prototype ontwikkeld, dit prototype dient na de stage verder ontwikkeld te worden tot een product. Dit verslag probeert ook enkele recommendations aan te leveren hiervoor.
Het bedrijf TAG software & consultancy bestaat ondertussen een jaar of 20. In deze tijd heeft het bedrijf een enorme schat aan ervaring opgedaan op het gebied van kwaliteitsmanagement, interne audits en performance scans. Tegenwoordig levert TAG software & consultancy ook standaard software hiervoor en hoogwaardige consultancy diensten op het gebied van kwaliteits-, proces-, risk- en klachtenmanagement, strategische beleidsvorming en organisatie vraagstukken.TAG software & consultancy heeft 22 medewerkers, waaronder enkele internationaal aangestelden.
Formulieren worden binnen een organisatie zo vaak gebruikt dat het soms gewoon niet meer opvalt, dat hetgeen waarmee gewerkt wordt in essentie als een formulier gezien kan worden. Zolang voor deze formulieren geen software oplossing gekocht/gemaakt is, dan zijn dit veelal Microsoft Word of Excel bestanden.Deze zijn vaak niet uniform, leveren een berg papier op of moeten geprint of gekopieerd worden alvorens gebruikt te worden. Dit zijn enkele nadelen die met eForms opgevangen kunnen worden.
De opdracht was dan ook het ontwikkelen van een prototype van eForms. Een product waarmee formulieren gemaakt, beheer en gebruikt kunnen worden. Daarnaast moet het product nog aan een aantal specifieke eigenschappen voldoen welke terug te vinden zin in de requirements definition.
Het product dient uiteindelijk te functioneren i.c.m. een windows of linux server en een MySQL of SQL server database. TAG eForms zal gebruik maken van MySQL of msSQL, verder wordt er geprogrammeerd in PHP en JavaScript. Deze code dient te worden gedocumenteerd door middel van een tweetal tools, JSdoc en PHPdoc. Deze genereren documenten aan de hand van de objecten methoden, attributen en het commentaar hierbij.
Voor de planning en project structuur is sterk gekeken naar eXtreme Programming en pair programming. Het betreffende project team bestond buiten de twee eerder genoemde studenten uit: Rene ten Oever, Emiel van Goor, Danny Hoogendoorn en Michael van Rijswijk. Het project is 4 globale fasen ingedeeld, formbuilder, workflowbuilder, mailboxen en back-end. Deze onderdelen zijn vrijwel niet in de code gekoppeld en kunnen dus prima los van elkaar ontwikkeld worden. Elke fase heeft 3 subfasen namelijk ontwikkelen, testen en debuggen.
Aan het begin van het project is, zoals eerder genoemd een aantal requirements opgesteld. Deze requirements zijn vervolgens onderverdeelt in een MoSCoW document, waarin de prioriteit van de functionaliteit in het stage project verduidelijkt wordt.
Verder is er in het ontwerp rekening gehouden met de verschillende database servers waarmee de software moet kunnen werken. Verder is het belangrijk dat eForms kan werken met het ASP model van TAG software & cosultancy.
eForms is gemaakt met het MVC principe als framework
Tot slot is er in eForms gebruik gemaakt van extJS, een javascript library die het ontwikkelen van gebruiksvriendelijke clientside vereenvoudigd. Het is

enorm belangrijk dat het maken en beheren van formulieren zo natuurlijk mogelijk is voor de gebruiker, wil een dergelijk product aanslaan bij het grote publiek.
Er werd gebruik gemaakt van pair programming, een methode waarbij 2 programmeurs een workstation delen, zodat zij gezamenlijk tot een resultaat komen. Dit levert een kwalitatief beter product op en een prettigere werksfeer, naar mening van de programmeurs. Voornamelijk omdat de discussie tijdens of voor het schrijven van de code al gevoerd kan worden en zo de beste keuzes gemaakt worden.
Het project heeft een prototype van eForms opgeleverd waarmee verder gewerkt gaat worden om te zorgen dat het een product wordt. De meeste van de vele gemaakte keuzes zijn goed bevallen. Vooral de keuze voor extJS is hier een goed voorbeeld van. Ook de ervaring die opgedaan is binnen TAG bij dit project, wordt gezien als een interessante aanwinst.

3. Inleiding

Het project dat wij uitgevoerd hebben ging om het ontwikkelen van een prototype van het product eForms. eForms staat voor elektronisch formulier. eForms omvat het ontwerpen, beheren en gebruiken van formulieren en het beheren van de workflow van deze formulieren. Naast het ontwikkelen van eForms heeft TAG ons gevraag een project opzet te documenteren zodat deze in het vervolg bij het opzetten van nieuwe project gebruikt kan worden. Wij hebben ons dan ook in de eerste fase van het project verdiept in verschillende project modellen die gebruikt zouden kunnen worden bij het opzetten van een software ontwikkel project.

De vragen die na het doorlopen van dit stage project beantwoorden konden worden zijn:

- Hoe werkt pair programming in de praktijk?
- Kost pair programming meer tijd en hoe verhoudt zich dit tot de kwaliteit van de code.
- Is het gebruik van Unit testing bij PHP echt voordelig?
- Is eXtreme Programming geschikt voor kleine projecten?
- Sluit ExtJS goed aan bij het te ontwikkelen product?
- Wat zijn de volgende stappen richting een Alpha release?

3.1 Reader

Dit document is als volgt opgebouwd.
We beginnen met een beschrijving van TAG gevolgd door de opdracht omschrijving. Als derde zullen wij de aanpak toelichten, hier komen een aantal punten uit het plan van aanpak terug en zullen we bij een aantal punt beschrijven hoe het in de praktijk heeft uitgewerkt. Na de aanpak komen we toe aan het design van het product. We beschrijven de ontwerpen, waarbij we voornamelijk zullen ingaan op de keuzes die wij hebben moeten maken.
Als vijfde beschrijven we het ontwikkel proces, we hebben gebruik gemaakt van eXtreme Programming en we willen hier een aantal bevindingen van beschrijven. Het volgende hoofdstuk beschrijft de te nemen stappen richting een volledige release.
Het laatste hoofdstuk gaat in op een aantal hoogtepunten van het project, we beschrijven zowel technische zaken als organisatorische zaken.
Tot slot zullen we in de conclusie / discussie beschrijven hoe wij de stage hebben ervaren en we zullen verder proberen antwoord te geven op een aantal vragen die wij aan het begin van het project hadden.

4. Bedrijf

TAG software & consultancy is bijna 20 jaar geleden begonnen als een multidisciplinair adviesbureau met de naam Ten Oever Advies Groep BV. Inmiddels is TAG uitgegroeid tot een software & consultancy bedrijf met ruim 20 vaste medewerkers. In 2006 is de naam aangepast aan de uitgebreide bedrijfsactiviteiten en sindsdien is de naam TAG software & consultancy bv.

TAG software & consultancy levert standaard software op het gebied van kwaliteitsmanagement en hoogwaardige consultancydiensten op het gebied van kwaliteits-, proces-, risk- en klachtenmanagement, strategische beleidsvorming en organisatie vraagstukken.

Met de advieservaring die jarenlang is opgedaan hebben wij software ontwikkeld op het gebied van kwaliteitsmanagement. Klanten van TAG zijn bedrijven binnen allerlei mogelijke branches met de nadruk op de bouw, handel, industrie, automotive, dienstverlening, zorg en transportsector. TAG focust zich hierbij op middelgrote en grote ondernemingen, clusters van ondernemingen (brancheorganisaties e.d.) en formule-matige opererende bedrijven.

TAG software heeft 22 medewerkers (incl. TAG software international) bestaande uit 10 universitair opgeleid, 6 HBO en 6 MBO. Hun gemiddelde ervaring bedraagt ca. 10 jaar.
TAG software heeft ongeveer 900 klanten, waarvan ca 40% actief en zo'n 5% internationaal. Klanten waren vroeger veelal klein tot middelgroot, tegenwoordig alleen maar middelgroot tot groot. Klanten buiten Benelux alleen maar groot, dit voornamelijk, omdat ze groot genoeg moeten zijn om er heen te vliegen. Partners zijn internationaal opererend advies- en certificeringsinstellingen.
De huidige omzet van TAG software is voldoende voor de huidige operations, maar er wordt verwacht dat met de internationale betrekkingen deze zal groeien tot een gewenst niveau.

5. Probleem/opdracht

5.1 TAG eForms®

De meeste mensen moeten op hun werk regelmatig formulieren invullen. Dat gaat van onkostenstaat naar verlofbriefjes en van verzekeringsformulieren tot rapporten van de verkoopgesprekken. Binnen je bedrijf maakt iedereen zijn eigen formulier voor bepaalde zaken en op den duur is het overzicht verdwenen. En nu zijn er al programma's die dit zeer specifiek oplossen, denk aan ERP of CRM programma's of de gebruikte accounting software. Echter zijn er veel formulieren binnen een organisatie die in deze niet thuis horen. Hierdoor zijn ze vaak organisatie specifiek opgesteld en niet eenvoudig de digitaliseren in een uniform systeem, dus?

Waarom niet één systeem waarmee je formulieren kunt ontwerpen, verspreiden en onderhouden? Dat is exact wat TAG eForms is!

Je kunt gebruiksvriendelijke, interactieve formulieren ontwerpen en ze in je bedrijf publiceren. Zelfs klanten, partners en leveranciers kunnen gebruik maken van de standaard documenten wanneer je ze toegankelijk maakt via internet. Dit betekent natuurlijk een enorme stap voorwaarts in je communicatie.

Het gebruik van de formulieren zorgt voor weinig overhead. Aanpassingen zijn zo gemaakt, gebruikers hebben altijd toegang tot de nieuwste versie van een formulier. Doordat je een formuliersjabloon kunt aanmaken met meerdere weergaven heb je minder formulieren nodig. De onkostenstaat wordt door de medewerker ingevuld, de tweede weergave wordt door de manager goedgekeurd en de derde weergave is voor de werknemer die zorgt voor uitbetaling.

5.2 Opdracht

TAG eForms moet een off-the-shelff product worden dat TAG aan haar huidige klanten en toekomstige klanten kan verkopen. TAG eForms moet dus zowel zelfstandig product kunnen werken als geïntegreerd kunnen worden in een pakket van TAG producten. TAG eForms zal ontwikkeld worden met behulp van PHP, JavaScript, MySQL en MSsql.

TAG biedt een stage project om TAG eForms te ontwikkelen tot een prototype. Deze opdracht omvat het verzamelen van de requirements, het ontwerpen van de software en opstellen van de benodigde UML documenten, het implementeren van software en het testen van de geïmplementeerde modules.

De studenten zullen begeleid worden door 2 experts. Als eerste zullen ze op technisch gebied begeleid worden door E. van Goor, deze zal het programmeer werk controleren, met name de de ontwerpen en de keuzes in het design. Ten tweede zal al het andere werk begeleidt worden door R. ten Oever, het gaat hier onder meer om de meer projectmatige werkzaamheden.

De opdracht voor de studenten is het ontwikkelen van een prototype van het product eForms. Het prototype is een testversie van eForms en zal slechts door een beperkt, vooraf samengesteld, aantal klanten van TAG bekeken worden. Dit in combinatie met een kleine training en informatie hoe het beste feedback gegeven kan worden aan TAG. Het ontwikkelde prototype zal dus door een beperkte groep lead users getest worden.

De lead users zijn inmiddels uitgezocht en benaderd, het gaat om grotere organisaties, met onder hen bedrijven die wereldwijd actief zijn. De contactpersonen hebben enthousiast gereageerd en hebben toegezegd graag te willen bijdragen aan eForms.

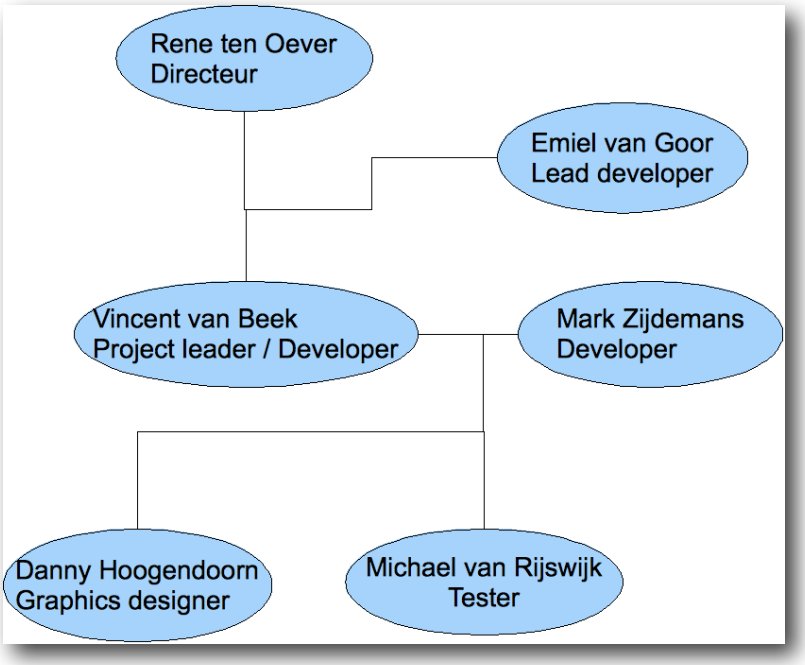
Tijdens de analysefase van het project zal een MoSCoW document opgesteld worden, dit document i.c.m. het verloop van het project bepaald de uiteindelijke vorm van het prototype.

6. Aanpak

6.1 hoofdpunten plan van aanpak

Hier volgt een korte opsomming van belangrijke punten uit het plan van aanpak.
Het eind resultaat van het project zal een prototype zijn. Ontwikkeld op basis van de kennis opgedaan tijdens een brainstorm sessie gevolgd door review stappen zoals beschreven in het plan van aanpak. Het ontwikkelen van de software zal gebeuren met behulp van Eclipse en een SVN repository. De volgende programmeertalen en databases zullen gebruikt worden: PHP, JS, MSSQL en MySQL. De documentatie van de PHP zal gebreuren met behulp van PHPdoc en de documentatie van de JS code zal gebeuren met behulp van JSdoc. Voor de planning en project structuur zullen we eXtreme Programming (XP) gebruiken. In de volgende paragraven zullen een aantal hoofdpunten nog verder besproken worden.

6.2 project team



figuur 6.1. projectteam eForms

Hierboven ziet u de situatie van het projectteam rond eForms.

6.3 rollen

Buiten de stagiaires Vincent en Mark bestond het team ook nog uit een aantal medewerkers van TAG. Rene de directeur van TAG is de opdrachtgever en het project team moet aan hem wekelijks verantwoording afleggen over de vorderingen en de planning. Emiel is de lead developer van TAG, hij is beschikbaar als er zich problemen voordoen op het gebied van techniek. Het project team bestaat dan uit vier personen met Vincent en Mark in een

leidende rol. Danny de graphics designer van TAG staat tot de beschikking van het team als het gaat om vormgeving. Michael is de support medewerker en tester van TAG en is beschikbaar als er user tests moeten worden uitgevoerd.

6.4 planning methoden

Aan de start van het project hebben we onderzocht of het mogelijk is RUP toe te passen op een klein project. Tijdens dit onderzoek [1] kwamen we er achter dat RUP te zwaar is voor een klein project en kwamen we na een zoek tocht terecht bij een model dat bekend staat als eXtreme Programming (XP). Toen de keuze eenmaal op XP was gevallen hebben we er ook voor gekozen om pair programming toe te passen dit wordt namelijk door het model aanbevolen en wij hadden het idee dat dit zou moeten werken maar we waren er nog niet helemaal van overtuigd en wilde het dus doormiddel van een praktijktest testen. Deze keuze had direct invloed op de wekelijkse planning en het verdelen van de taken. Iedere maandag ochtend evalueerde Vincent en Mark de vorderingen en maakten een planning voor de komende week. Na de analyse fase is er een algemene planning gemaakt met deadlines voor de te ontwikkelen modules. De voortgang van deze planning wordt wekelijks met Rene besproken.

6.5 project fasen

Het project is in een aantal fasen op te delen. Het project startte met een periode van onderzoek en het schrijven van een aantal artikelen. Tijdens deze fase hebben we een aantal technische onderdelen van het project onderzocht en we hebben een onderzoek gedaan naar de beste planningsmethoden voor kleine projecten.

Tijdens de opstart fase hebben we met de kennis opgedaan tijdens de eerste fase een plan van aanpak geschreven. In dit plan hebben we beschreven aan welke voorwaarde het project moet voldoen. en welke methoden we zullen toepassen om aan de gestelde voorwaarden te voldoen.

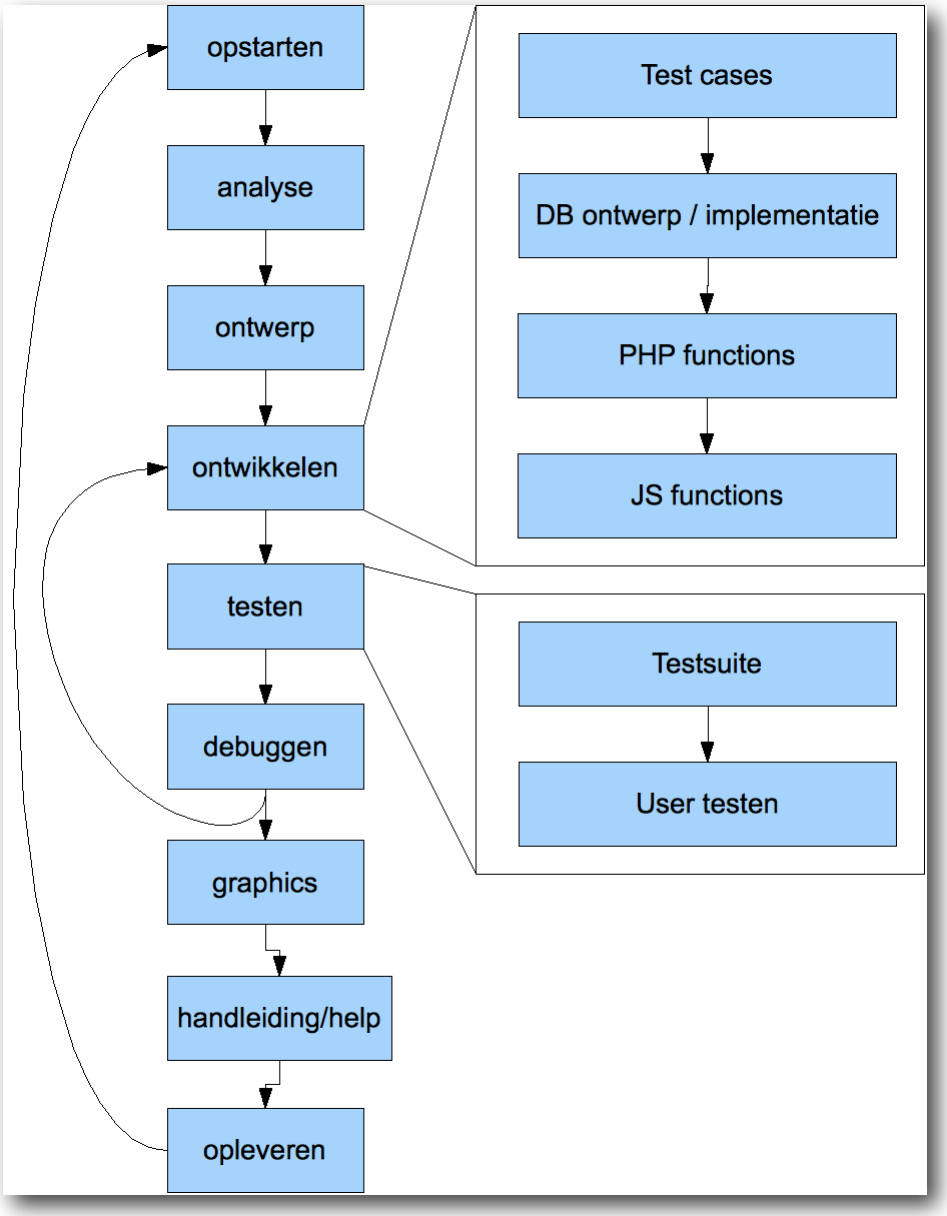
De analyse fase is op te delen in een aantal subfasen. We zijn begonnen met een brainstorm sessie. De kennis opgedaan tijdens deze sessie hebben we verwerkt tot requirements definitions en use-case models.

Na de analyse zijn we begonnen aan de ontwikkel fasen. 1 iteratie van de ontwikkel fase bestaat uit 3 subfasen namelijk ontwikkelen, testen en debuggen. Met behulp van de MoSCoW methode hebben we de requirements opgedeeld in modulen. Deze modulen hebben we op een iteratieve wijze ontwikkeld.

Na het ontwikkelen van alle functies hebben we een fase van het project geweid aan het implementeren van de graphics.

Als laatste voor de oplevering hebben we tijd besteed aan het maken van de handleiding en de helpfiles.

Project fasen na de onderzoek fase.



figuur 6.2. project iteraties

6.6 documenten

Er is een aantal ontwerp documenten opgesteld. Twee documenten tijdens de analyse fase, de andere tijdens de iteraties van de ontwikkel fase. Het requirements document en de use-case analyse zijn tijdens de analyse fase opgesteld. Het database ontwerp is tijdens de iteraties gemaakt. Ook hebben we aan het begin van iedere iteratie een soort tussen vorm van een classe diagram en een sequence diagram gemaakt. Dit laatste diagram diende er voor om de functionele vereisten uit het use-case diagram formeler te maken. De documenten zijn opgeslagen in het doc formaat en geëxporteerd naar PDF (t.b.v. distributie onder de lezers). Verder zijn alle documenten van een versie nummer en een status voorzien.

iteraties	fasen	oplevering
Formbuilder	ontwikkelen	01-12-2008 05-12-2008
	testen	05-12-2008 08-12-2008
	debuggen	08-12-2008 11-12-2008
Workflowbuilder	ontwikkelen	10-12-2008 18-12-2008
	testen	18-12-2009 19-12-2009
	debuggen	15-01-2009 17-01-2009 (za)
mailboxen / weergave	ontwikkelen	18-12-2009 19-12-2008
	testen	19-02-2009 26-02-2009
	debuggen	27-02-2009 30-02-2009
Back-end	ontwikkelen	02-02-2009 10-02-2009
	testen	11-02-2009 13-02-2009
	debuggen	16-02-2009 20-02-2009

tabel 6.1. oplever data

In de bovenstaande tabel worden de data van de initiële ontwerpen en ontwikkelingen niet vermeld, deze vonden in de maand november voornamelijk plaats.

7. Design

7.1 inleiding

In dit hoofdstuk wordt getracht de onderdelen van het ontwerp toe te lichten en de gemaakte keuzes duidelijk te maken. We beginnen met de requirements, waaronder de must en should haves. Hierna zullen de use cases toegelicht worden. Vervolgens zullen de 2 fundamenten van eForms worden toegelicht, namelijk het PHP framework en de JavaScript library. Het database ontwerp zal vervolgens aanbod komen, hier wordt de toch vrij complexe database die voor eForms gebruikt wordt beschreven. Tot slot zullen de bijzondere delen van eForms beschreven worden, het formulieren ontwerp systeem en de workflowbuilder. In dit deel worden ook enkele punten beschreven die bij een applicatie horen die deels clientside werkt

7.2 Requirements

De requirements van het systeem zijn tot stand gekomen volgens de in het plan van aanpak beschreven methode. Deze methode omhels de volgende stappen:

1. brainstorm sessie
2. opstellen verslag
3. review verslag
4. opstellen requirements
5. review requirements

De uitkomst van dit proces is verwerkt in een document waarin de requirements zijn opgesplitst in een aantal categorieën: functioneel, gebruiksvriendelijkheid, design, beveiliging en systeem eisen.
Ieder onderdeel van de requirements wordt door een aantal kern requirements gekenmerkt. De functionele requirements zijn ingedeeld naar prioriteit, we hebben dit gedaan met behulp van het MoSCoW [1] model. De andere requirements zijn niet ingedeeld naar prioriteit, deze requirements zijn eigenlijk meer te zien als eisen waar het porject minimaal aan moet voldoen. Een belangrijke design eis is dat het eForms product moet passen in de TAG stijl. Een aantal belangrijke systeem eisen zijn bijvoorbeeld de programmeertaal PHP5 en de databases MySQL 5 en SQL 2005. Eisen omtrent de gebruiksvriendelijkheid zijn eigenlijk meer uitsluitingen van een aantal zaken om zo het project af te bakenen. Denk hier bijvoorbeeld aan het ontbreken van functionaliteit voor slecht zienden.
De beveiligingseisen zijn te verdelen in 2 groepen, de functionele eisen en kern eisen. De functionele eisen zijn ondergebracht bij de functionele requirements en de kern eisen zijn beschreven als het beschikbaar zijn van een https mogelijkheid voor klanten die een ASP (ASP staat binnen TAG software voor Application Sevice Provided, hierbij worden de programma's op de servers van TAG software gehost.) licentie gebruiken.

De functionele requirements zijn in de volgende 9 hoofdpunten verdeeld en naar prioriteit gerangschikt volgens het MoSCoW model :

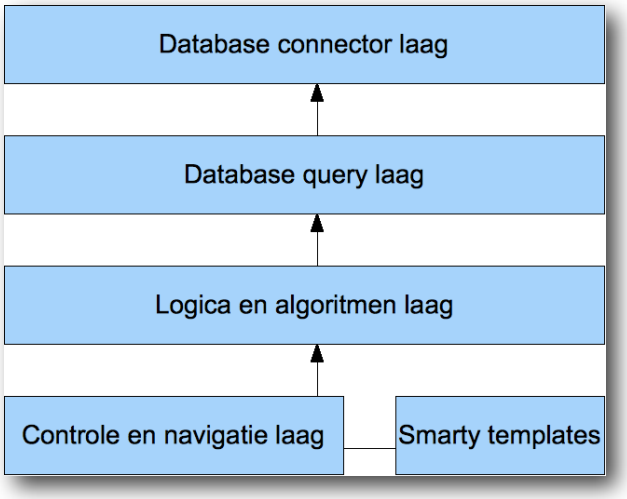
Must	Should	Could	Won't
4. Formbuilder	2. Berichtboxen	7. Backend	3. Printen
1. Formbehandeling		8. Gebruikersbeheer	6. Rapportage
5. Workflow beheer			
9. Security			

tabel 7.1. MoSCoW model TAG eForms stage project

Het te ontwikkelende prototype moet enige functionaliteit bevatten, hieruit is voortgekomen dat de formulieren gemaakt moeten kunnen worden met daarachter een workflow. Verder is de standaard inlog situatie van TAG software voldoende beveiliging voor het prototype. Verder was het mooi om het voor elkaar te krijgen om de berichtboxen enigszins te ontwikkelen, zodat het prototype te demonstreren is met een complete cyclus van een eForms

7.3 PHP framework

Voor de serverside software hebben wij als programmeer taal PHP gebruikt. Om de PHP code zo goed mogelijk te structureren hebben wij er voor gekozen een PHP framework te gebruiken. Dit framework splits de software in drie lagen volgens het Model View Control (MVC) model. Waar in het originele MVC model de Model laag wordt gerepresenteerd door Objecten hebben wij er voor gekozen deze laag tot een database laag om te vormen. Het model gaat er dan als volgt uit zien:



figuur 7.1. php model, afgeleid van appendix A4

Er zijn vele PHP frameworks beschikbaar en deze verschillen in complexiteit functionaliteit en ondersteuning. Wij hebben er voor gekozen om framework te gebruiken dat uitsluitend de structuur bepaald. Het framework bied verder geen functionaliteit, uitsluitend de ruimte om deze te ontwikkelen. Aangezien wij het framwork moeten in passen in bestaande software van TAG bied dit framework ons alle vrijheid en ruimte dit te doen. Vooral de mogelijkheid om

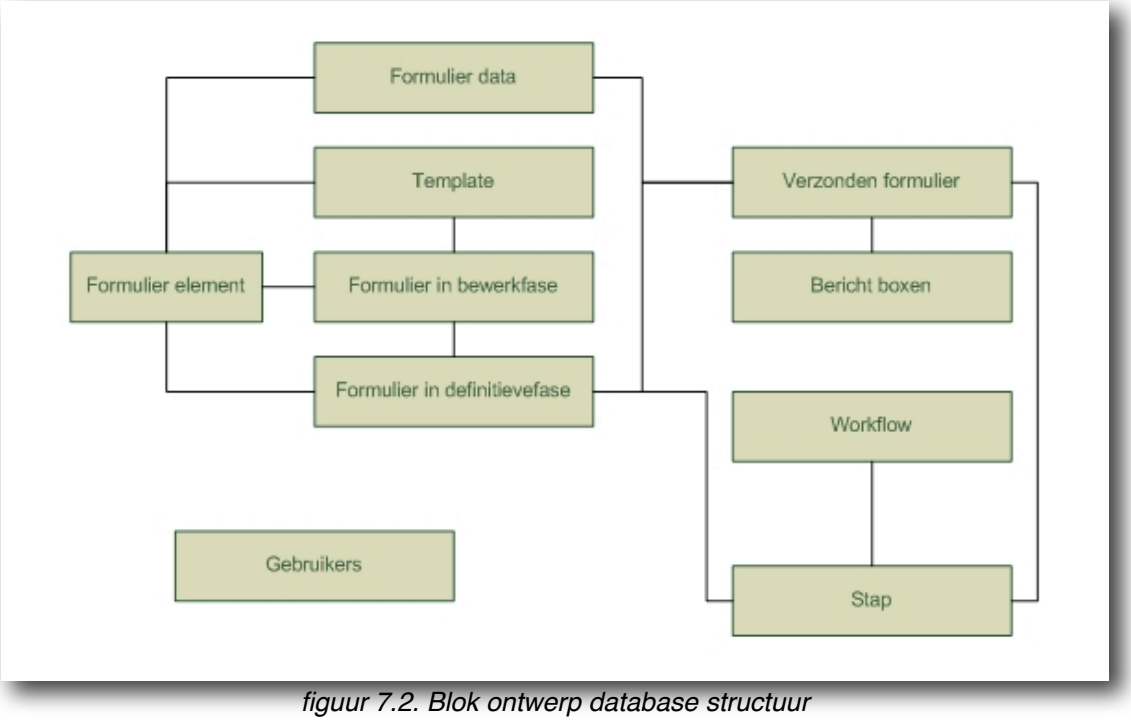
de ondersteuning voor verschillen de databases in te bouwen en de mogelijkheid om in een MVC structuur te werken heeft ons tot deze keuze doen komen.

7.4 JavaScript (JS)

Om een goede basis te hebben waarop de rest van de clientside software gebouwd kan worden hebben wij gekozen om een JS library te gebruiken. Er zijn veel JS library's beschikbaar en het kiezen van de meest geschikte kan op een aantal verschillende manieren. Bij TAG is er gekozen om de keuze van de library af te laten hangen van de ondersteuning en service die bij de library gegeven wordt. Veel library's zijn opensource en bij deze ben je voor de service en ondersteuning afhankelijk van de community die achter het project zit. De library die bij TAG uiteindelijk gekozen is is ExtJS [2], deze library is zeer uitgebreid en wordt professioneel ondersteund. Hiernaast hebben wij ook vast kunnen stellen dat Ext in hun ontwikkeling van extJS rekening houdt met de webstandaarden en recomdation van W3C. Dit zien wij als een zekerheid voor de langere termijn en voor de eventuele uitbreidbaarheid van het extJS door programmeurs van TAG.
Tijdens de eerste fase van het project hebben we ook nog onderzocht hoe deze library inzetbaar zou zijn voor eForms. Aangezien de library een aantal kern punten van eForms ondersteund sluit het dus ook goed aan bij dit project. Een aantal van deze kern punten zijn de standaard ondersteuning van AJAX, Drag & Drop, Folder Tree, Ordered lists, in-window forms en Tabbladen.
Een van de grote problemen bij het ontwikkelen van eForms was het debuggen van functies die gebruik maaken van AJAX. Als een JS functie bijvoorbeeld een JSON [3] string verwacht en er een PHP error terug gegeven wordt door de server dan is het vaak lastig om de bron van de fout te vinden. Een hulpmiddel wat dit debug proces duidelijk versnelt heeft is Firebug [4]. Firebug is een plugin voor Firefox. Met behulp van deze plugin is het mogelijk de status van JS variabelen te evalueren en om de het http verkeer tussen de browser en de server te evalueren. Verder is Firebug ook een debugger voor JS zelf, in de console van Firebug worden JS errors gemeld met het regel nummer, dit regelnummer is wel het regelnummer dat in de browser vast gesteld wordt, maar geeft toch een goede indicatie van de locatie van de bug.

7.5 Database ontwerp

Bij het ontwikkelen van de database structuur hebben we vanaf het begin rekening gehouden met het feit dat de database zowel op MySQL 5 en SQL server 2005 gebruikt gaat worden. Om te voorkomen dat er veel verschil moet worden gemaakt tussen de queries voor de verschillende databases is het van belang een generieke structuur te ontwikkelen. Bij het ontwikkelen van de database structuur hebben wij gebruik gemaakt van een blackbox principe. We hebben in de eerste ontwikkel fase van het project een algemene structuur opgezet, deze structuur hebben we later tijdens de iteraties van de ontwikkel fase verfijnd en uitgewerkt.

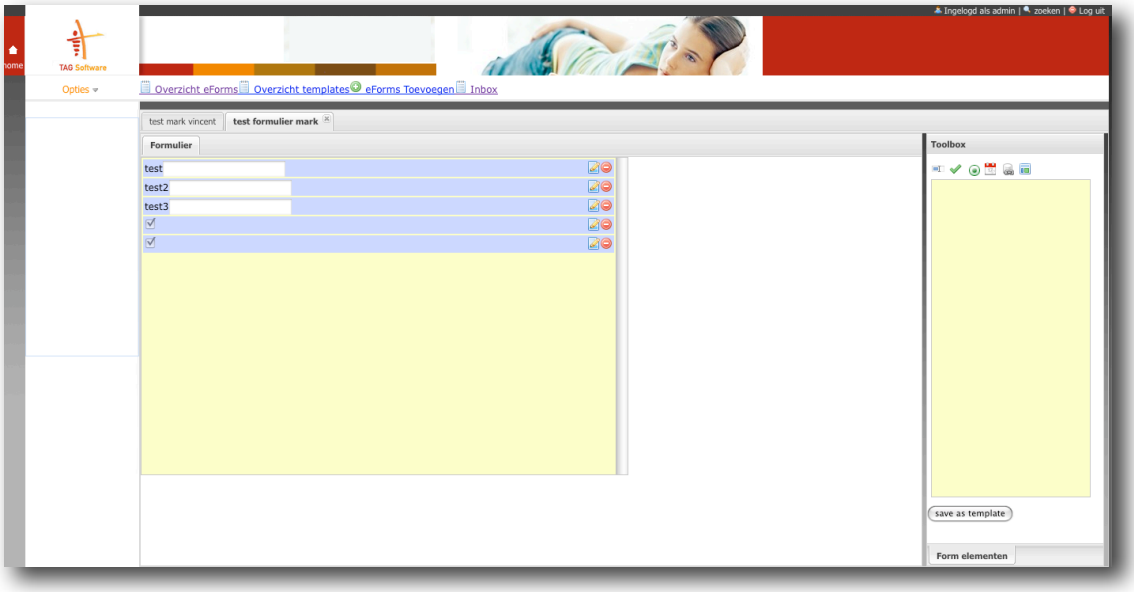


figuur 7.2. Blok ontwerp database structuur

Het voordeel van het werken met dit principe is dat het niet nodig is om de gehele structuur op voorhand te ontwerpen. Dit zorg ervoor dat er goed in iteraties gewerkt kan worden en dat er de mogelijkheid is kleine aanpassingen te doen zonder dat eerst het gehele ontwerp moet worden aangepast. Het wordt uit deze blok opzet wel duidelijk waar toekomstige verbindingen komen te liggen tussen de verschillende modules.
De volledige database structuur is te vinden in appendix A5.

7.6 Formbuilder

In eerste instantie is het eForms project ontstaan uit de vraag of het mogelijk is papieren formulier makkelijk te vervangen door elektronische formulieren. Het maken van de formulieren is dus een kern taak van eForms. We kwamen er echter snel achter dat alleen het maken van formulieren niet voldoende was, deze papieren formulieren die vervangen moesten worden werden namelijk van persoon a verplaatst naar persoon b en beide personen moesten een deel van het formulier invullen. Het eForms project bleek dus niet alleen over formulieren te gaan maar ook over een workflow die bij het formulier hoort. In deze paragraaf zullen wij verder ingaan op de ontwikkeling van de formbuilder, dit is het deel van de software waarmee het elektronische formulier gemaakt en bewerkt wordt.
Bij de ontwikkeling van de formbuilder stonden een aantal vragen centraal, hoe gebruiken we drag en drop om de formbuilder gebruiksvriendelijk te maken, hoe zorgen we er voor dat oude formulieren inzichtelijk blijven nadat het formulier gewijzigd is en hoe zorgen we er voor dat er zo min mogelijk data verloren gaat als het client side deel van de software crasht (JS en browser).



figuur 7.3. screenshot eForms formbluider

7.6.1 gebruiksvriendelijkheid en drag en drop

Sinds de invoering van de muis en de graphical user interface [5] zie appendix E, is ook het gebruik van drag en drop ook niet meer weg te denken uit de huidige computer systemen. Het principe van drag en drop is zo ingeburgerd dat het geen uitleg meer behoefd aan de gebruikerszijde. Dit is dan ook de rede dat wij er voor gekozen hebben om drag en drop te gebruiken om het maken van een formulier zo gebruiksvriendelijk en intuïtief mogelijk te maken.

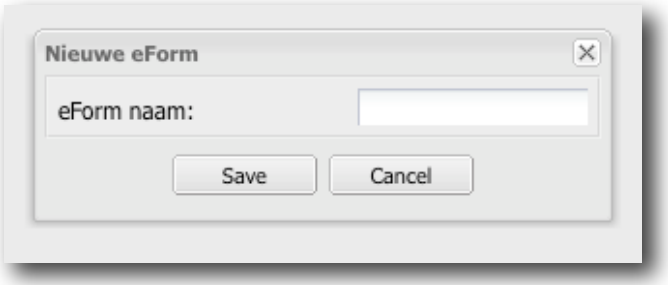
7.6.2 formulieren inzichtelijk maken na wijziging van het formulier

In appendix A5 is te zien dat een formulier wordt opgeslagen als elementen gekoppeld aan een formulier. Deze elementen kunnen later natuurlijk weer worden gewijzigd. Er ontstaat dan echter een probleem. Het is namelijk zo dat de data die ingevuld wordt in een formulier gekoppeld moet worden aan een veld, anders is de data later niet meer terug te vinden. Wordt een veld echter gewijzigd of zelfs verwijderd dan zou de data van een oud formulier dus niet meer terug te plaatsen zijn. Dit probleem is vergelijkbaar met situaties waarin een papieren formulier in een organisatie vervangen wordt door een nieuw formulier. De oude papieren formulieren zullen blijven bestaan met alle elementen er op. Dit moet in de software dus ook zo blijven.
Dit probleem hebben wij als volgt opgelost. We hebben twee tabellen in de database (form_edit en form_elements_edit), deze tabellen representeren een formulier dat kan worden bewerkt. Is een formulier af dan wordt het formulier “definitief” gemaakt, dit betekend dat het formulier en alle elementen wordt gekopieerd naar twee andere tabellen (form_use en form_elements_use). De formulieren die door deze twee tabellen worden gerepresenteerd kunnen niet meer worden gewijzigd en kunnen worden ingevuld en de data kan dus aan de velden worden gekoppeld en zal dus niet meer verloren gaan. Het te bewerken formulier zal blijven bestaan, wordt dit formulier gewijzigd (bijvoorbeeld een veld toevoegen) dan kan het formulier daarna weer “definitief” worden gemaakt. Dit laatste betekend dat er een nieuwe kopie van het formulier in de tabellen (form_use en form_elements_use) wordt gezet. Formulieren ingevuld voor de wijziging zullen dan dus worden weergegeven met behulp van de oude data uit (form_use en form_elements_use) en nieuwe formulieren gebruiken de nieuwe kopie in (form_use en form_elements_use).
De kopieer procedure gebeurd met behulp van querys, deze querys worden uitgevoerd in een transactie om er zeker van te zijn dat er nooit corrupte formulieren in de tabellen (form_use en form_elements_use) terecht komen.

7.6.3 gevaar dataverlies client side

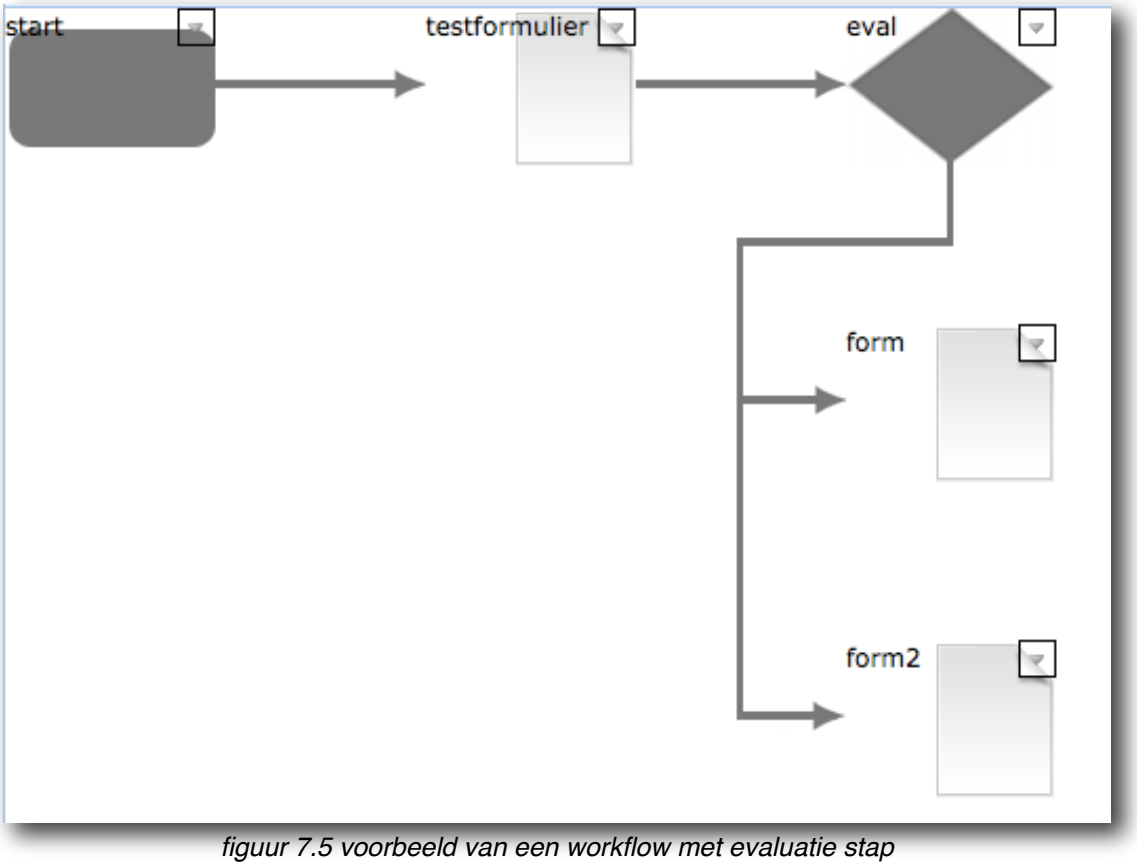
Om gebruik te kunnen maken van drag en drop hebben wij JavaScript moeten gebruiken. De formbuilder is grotendeels geprogrammeerd in JavaScript, JavaScript wordt uitgevoerd op de client PC en niet op de server. Dit heeft tot gevolg dat er data verloren kan gaan als de gebruiker vergeet op te slaan of als bijvoorbeeld de client crashed. Wij hebben er dus voor gekozen om iedere wijziging direct met behulp van AJAX naar de server te sturen en direct te verwerken in de database. Een andere keus had kunnen zijn om eerst het gehele formulier te maken en daarna het geheel naar de server te sturen. Dit zou iets minder dataverkeer opleveren. Het gaat hier echter om zulke kleine hoeveelheden data dat dit nooit zal opwegen tegen de kans van het verlies van data aan de client zijde.

7.7 Workflowbuilder



figuur 7.4. screenshot eForms nieuw eForm

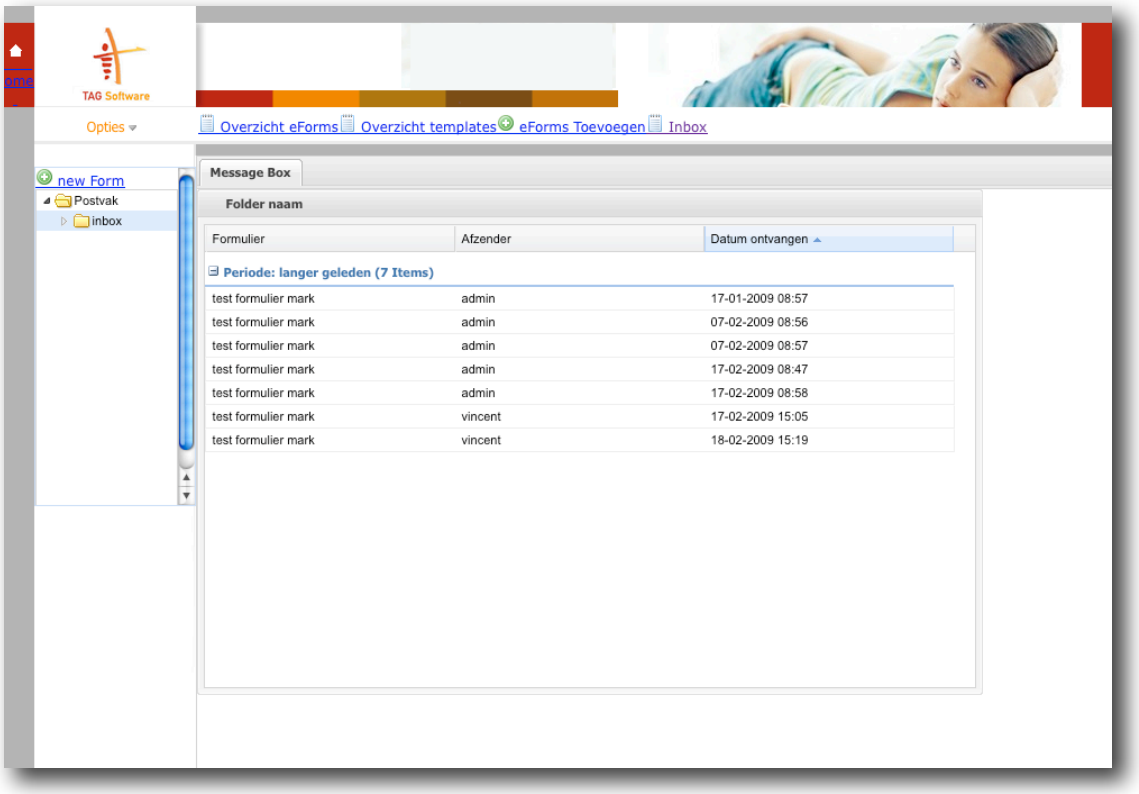
Zoals al beschreven bij de Formbuilder bestaat een formulier niet alleen uit een formulier maar ook uit een “workflow”. Onder een “workflow” verstaan wij de stappen die een formulier moet maken om van de initiator tot de eind bestemming te komen. Om deze “workflow” inzichtelijk te maken voor de maker van het formulier hebben wij een workflowbuilder gemaakt die de workflow met behulp van een diagram weergeeft. Een workflow diagram wordt gekenmerkt door een aantal elementen. Zoals in figuur 7.5 te zien is bestaat een workflow uit een start element, een formulier elementen, evaluatie elementen en pijlen. Het start element dient er voor om aan te kunnen geven welke personen het formulier mogen initiëren. Een formulier element vertegenwoordigd een formulier (in de papieren variant vertegenwoordigd dit een deel van een formulier dat door 1 persoon moet worden ingevuld). Een evaluatie element vertegenwoordigd een keuze. Op het voorgaande formulier moet dan een keuze element bestaan en aan de hand van de ingevulde waarde wordt dan het volgende formulier gekozen. Voor de gemiddelde gebruiker geeft deze manier van werken geen natuurlijk gevoel. Echter is dit enkel door de maker van het formulier te gebruiken en voor hem is dit zeker te spiegelen tegen de werkelijkheid.



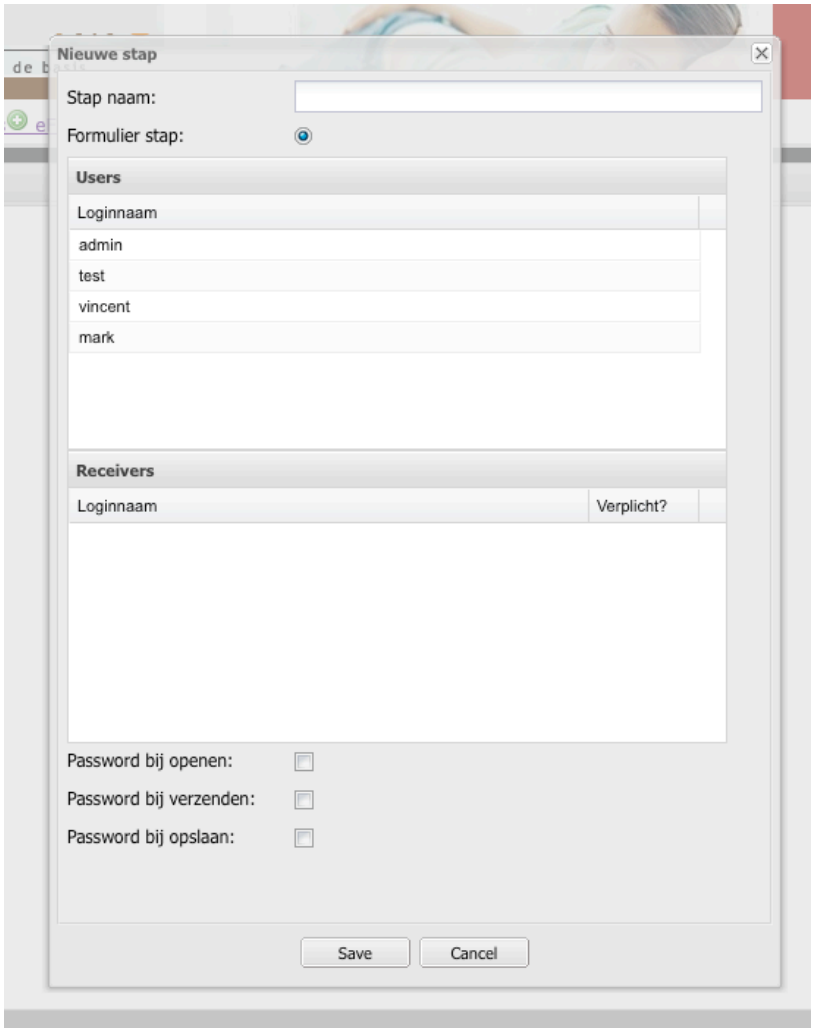
figuur 7.5 voorbeeld van een workflow met evaluatie stap

7.8 Message Box

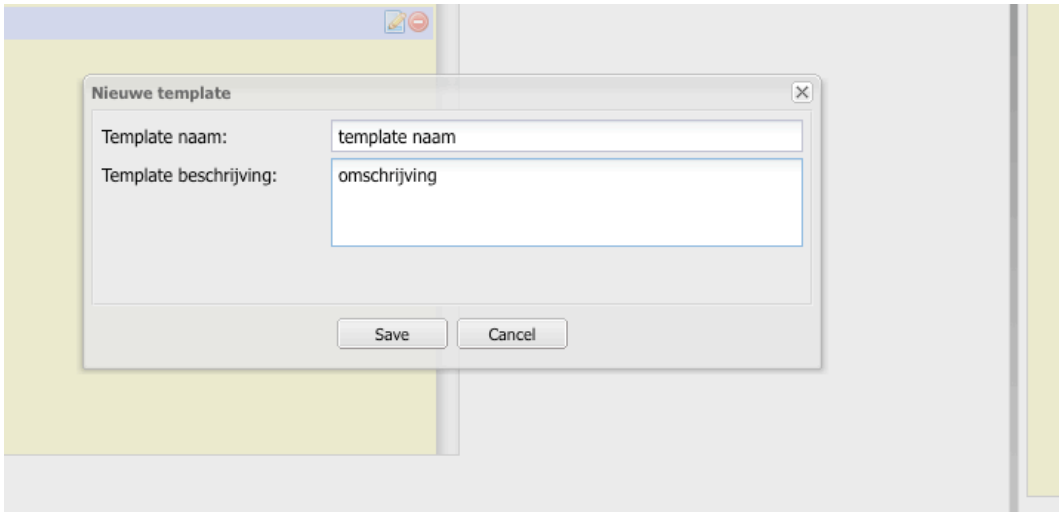
Formulieren moeten van persoon naar persoon worden gestuurd. In de echte wereld zou dit plaatsvinden door een deels ingevuld formulier bij een tweede persoon in het postvakje te leggen. Het elektronische formulier heeft dus ook een postvak nodig. Het mooie van een elektronisch postvak is dat het heel makkelijk is om meerdere vakjes te maken voor verschillende doeleinde, bijvoorbeeld archief mappen, 'postvak in' map en een 'postvak uit' map. Ook maakt een elektronische map het mogelijk om formulieren heel erg makkelijk te sorteren op bijvoorbeeld datum van ontvangst. Zeker omdat wij gebruik maken van ExtJS, deze library bevat namelijk een API waarin ondermeer een een sorteerbare lijst gefaciliteerd wordt. Deze lijst heeft vrijwel de zelfde look en feel als bijvoorbeeld de veel gebruikte mail client Outlook van Microsoft. Dit zorgt er voor dat het gebruikt van de mappen en de postvakken erg intuïtief en gebruiksvriendelijk is.



figuur 7.6. screenshot eForms een willekeurige inbox



figuur 7.7. screenshot eForms nieuwe stap in de workflowe



figuur 7.8. screenshot eForms nieuw template

8. Software ontwikkeling

8.1 Ontwikkel omgeving

Er is ervoor gezorgd dat de gebruikte ontwikkel omgeving bij TAG eForms, dezelfde is als de uiteindelijke systeem vereisten. Zo werd er gewerkt met PHP5 en MySQL5. Hieronder vindt u een korte lijst met de overige elementen.

1. Microsoft SQL Server 2005
2. Microsoft SQL Server 2008
3. Microsoft IIS 6
4. Microsoft IIS 7
5. Apache 2.x
6. MySQL 5.x
7. PHP 5.x
8. Zend optimizer 3.3.3 (i.c.m. PHP 5.1.4 en Apache 2.0.58)
9. Linux 2.4
10. Linux 2.6
11. Windows Server 2003
12. Windows Server 2008

Punt 8 in de bovenstaande lijst is de server configuratie die bij TAG software vast gesteld is als de meest stabiele combinatie van de 3 toepassingen.

Er zijn ook omgevingen waar bijvoorbeeld geen rekening mee gehouden is. Dit veelal gezien dat de versie dusdanig achterloopt van de betreffende toepassing, zo is MySQL 4 en eerder niet ondersteund. Ook PHP 4 is niet geschikt voor TAG eForms.

8.2 Pair programming

Pair programming [6] is voornamelijk, zoals de naam al zegt, het programmeren in een paar. Pair programming is een methode om de kwaliteit van de software beter te laten zijn, dan tijdens conventioneel programmeren in project verband.
Programmeren in paar verband kost op de lange termijn veel minder tijd, echter op korte termijn kost het enige overhead. Deze overhead is voor kleinere bedrijven vaak wel de reden om niet te kiezen voor pair programming. Tijdens het project eForms zagen wij dat het de kwaliteit zeker ten goede kwam. Ook zagen wij dat het maken van modules die sterk afhankelijk zijn van onderdelen waar juist de “ander” het meeste vanaf weet, juist sneller gemaakt worden als die persoon ook daadwerkelijk meedenkt.
Echter zien we ook dat bij druk van deadlines vaak automatisch gekozen wordt om paralel te werken, terwijl dit uiteindelijk niet sneller hoeft te zijn. Wat we met zekerheid kunnen zeggen met de ervaring van het eForms project is dat pair programming de kwaliteit ten goede komt. Een extra voordeel is het afnemen van frustratie momenten, mocht het zo zijn dat een programmeur er even niet uitkomt.

8.3 Unit testen

Gezien dat eForms zoveel mogelijk object georiënteerd is geprogrammeerd, is het evident dat unit testing uitstekend geschikt zou zijn om de objecten en hun methoden te testen. Unit testing houdt in dat je op basis van kleine stukken samen horende code losse tests uitvoert of deze aan de gestelde eisen voldoen. Met name de methoden van de objecten zijn mooie “units”.[7]
Elke methode van de aanwezige objecten is door middel van unit testing getest om ervoor te zorgen dat ze voldoen aan het ontwerp. Een klein voorbeeld van een willekeurig test resultaat van eForms kunt u vinden in appendix D.
Een ontwikkel methode waar naar gekeken is door het project team is “test driven development” [7]. Aangezien wij voor de rapid development methoe eXtreme Programming hebben gekozen bleek test driven development hier zeer goed bij aan te sluiten. Iedere ontwikkel iteratie startte namelijk met het programmeren van de test klassen. Omdat er ook gebruik werd gemaakt van pair programming werd er in deze fase vaak even een laptop bij gehaald zodat de tweede developer alvast de klasse en de methodes kon aanmaken die in de test klassen werden omschreven. Ook werd een deel van de PHPdoc al in deze fase geschreven. Na het opstellen van de test klassen werd de database verder uitgebreid naar de wensen van de te ontwikkelen functionaliteit. De volgende fase was het ontwikkelen van de PHP methoden. Deze werden direct na het programmeren getest met de eerder geschreven test klassen. Door deze manier van testen hebben we veel tijd kunnen besparen. Dit had een aantal redenen, de voornaamste rede was dat het aantal user test drastisch werden terug gebracht. We gingen pas over tot user testen nadat alle unit tests positief waren. Een andere belangrijke rede voor het besparen van tijd was het snel terug vinden van tik fouten en syntax errors. Ook het feit dat de
methode voorhanden “vers” in de gedachte van de programmeur zit levert veel tijd winst op.
Om unit testen van de PHP code mogelijk te maken hebben we gebruik gemaakt van Simpletest [10]. Simpletest is een testsuite die veel overeenkomsten heeft met JUnit voor Java. Ook maakt Simpletest het mogelijk om daadwerkelijk op functioneel niveau te testen, den hier bij aan het testen van inloggen en invoeren van data in een formulier. Dit laatste deel van Simpletest hebben wij door de tijdsdruk niet verder onderzocht en getest maar een aantal voorbeelden hebben ons er van overtuigd dat het zeker de moeite waard is dit deel van Simpletest verder te onderzoeken.
De ervaring van de team leden leerde dat user testing [7] alleen een tijdrovende bezigheid is en ook vervelend. Het terugvinden van de daadwerkelijke bug kost nog eens extra veel tijd. Dit is dan ook enkel geschikt voor validatie van het software product.

9. Discussie

drag and drop

Drag and drop lijkt vooralsnog een natuurlijk methode voor het maken van een formulier. Het geeft de gebruiker een overzichtelijke methode voor het plaatjes van elementen op een veld. Echter zitten er technische nadelen aan het principe. Een voorbeeld hiervan is dat je als programmeur/ontwerpen eenvoudig de methodes te specifiek maakt. Dergelijk "fouten" zorgen ervoor dat het lastiger is om later de drag and drop module uit te breiden met andere items of de drag and drop tool te gebruiken voor andere onderdelen van TAG eForms of andere producten van TAG.

workflowbuilder

De workflow die gemaakt moet worden voordat de formulieren aan de stappen toegekend kunnen worden, komt niet natuurlijk over. Uiteraard is het werkbaar en zullen gebruikers snel genoeg wennen aan de situatie, het is zelfs prima te verklaren waarom deze methode ook logisch is. Echter is het verstandig dit verder uit te zoeken, op dit moment werkt Danny Hoogendoorn (graphics designer) aan een voorstel om de flow naast het eForms weer te geven en zo een natuurlijker werkproces te creëren.

help

Tijdens de stage is er weinig tijd gebleken voor het ontwikkelen van een helper functie. Het is misschien niet noodzakelijk voor een prototype, maar gezien de wens vanuit TAG software om producten te ontwikkelen met een hoog gebruiksvriendelijkheidsgehalte leek het ons zeer nuttige om hier goed over na te denken en indien de tijd het toe liet een opzet te maken. Er is goed over nagedacht door het project team, hier uit bleek ondermeer dat het interessant is om de help file te laten openen met items die van toepassing zijn op hetgeen dat de gebruiker voor zich heeft in eForms. Verder was ook vastgesteld dat een speciale helper applicatie (denk aan het paperclipje in Microsoft Word, maar dan eenvoudiger en prettiger) niet voldoende zou kunnen bijdragen aan het product.

pair programming

Pair programming is in de smaak gevallen bij de projectleden, we merkten wel op dat een bedrijf als TAG software moet wennen aan het feit dat 2 mensen tegelijk achter 1 PC werken. Het lijkt op het eerste gezicht wel een productiviteitsverlies, echter stijgt de kwaliteit enorm. Ook is het werk veel minder frustrerend in het geval van problemen, je kunt ze gelijk samen aanpakken. Ook komen sommige problemen minder snel voor.

Het is wel de vraag of bij een uitgebreider ontwerp (als er dus geen extreme programming gebruikt wordt) pair programmingen net zo veel voordelen met zich mee brengt, gezien het feit dat dan de documentatie uitgebreider en strenger is. Er is dan dus minder ruimte voor eigen inbreng, waardoor de issues waarbij pair programming ideaal is, minder voorkomen, denk hierbij aan onduidelijkheid over het formaat van de geretourneerde data van een functie ed.

extJS

extJS is als library erg goed bevallen voor de ontwikkelaars. De leercurve vergde veel geduld en uitzoek werk, maar is het resultaat meer dan waard. We zien dan ook dat de andere lopende project bij TAG software deze tool snel overgenomen hebben na de eerste demonstraties van de kracht. De kosten van deze tool zijn op kleine schaal, en bij het voldoen aan bepaalde eisen zeer goed te overzien, zeker als we de voordelen bekijken is deze tool zeer goedkoop en toch professioneel van aard.

10. Conclusie

Tijdens het stage project rondom TAG eForms is een goed prototype ontwikkeld. Er is veel ervaring opgedaan in het ontwikkelen van een webapplicatie waarbij een grote hoeveelheid client-side dynamiek noodzakelijk is. Echter zal, zoals eerder vermeld er nog veel werk zijn tot TAG eForms een verkoopbaar product zal zijn. Voornamelijk moet nu gekeken worden hoe het creëren en onderhouden van de formulieren op een natuurlijkere wijze kan voor de gebruikers. Verderop in hoofdstuk 11. de Recommendation zullen wij verder opsommen welke stappen genomen zullen moeten worden volgens het projectteam.
Als projectteam hebben we de sterke punten van extJS dusdanig positief ervaren dat we deze library sterk kunnen aanraden aan ieder team dat een web-applicatie ontwikkeld waarbij veel client-side dynamiek een echte pre is. Niet alleen zijn de mogelijkheden enorm, maar ook de ondersteuning en de aanwezigheid van voorbeelden is meer dan toereikend. Echter kost commerciële toepassing van deze tools wel geld.
Tijdens het project is gebruik gemaakt van eXtreme Programming, deze methode viel goed bij het project team. We hebben gemerkt dat onderdelen van deze methode al gebruikt werden binnen TAG doordat deze automatisch past bij een van de lopende projecten. Vanuit management kant was het wennen om te zien dat de ontwikkelaars gebruik maakte van Pair Programming, deze methode was binnen TAG niet gebruikelijk en het samenwerken achter 1 systeem lijkt inefficiënt. Echter heeft het TAG eForms project bewezen dat de methode de tijd goed maakt in winst in minder overleg tijd en toename in kwaliteit. De overige delen van eXtreme Programming pakten goed uit binnen het project.
Vanuit de oorsprong is PHP een script-taal en geen object georienteerde taal. Echter is PHP uitgegroeid tot een professionele object georienteerde taal. En dit brengt veel voordelen met zich mee.
Unit testen van de PHP code heeft in het geval van eForms zeker veel voordelen. Aangezien wij voor een object georiënteerde aanpak hebben gekozen sluit Unit testen daar naadloos bij aan. Ook het feit dat wij voor rapid development (in de vorm van eXtreme Programming) hebben gekozen rechtvaardigt de keuze voor Unit testen. Uit de praktijk ervaring is gebleken dat Simpletest een goede tool is om Unit testen van PHP code mogelijk te maken. Het resultaat van Unit testen is een kortere een minder frustrerende debug periode en een waarborg voor de kwaliteit van de software.
Zoals bij de reflectie (hoofdstuk 12.) beschreven zal worden, hadden wij meer tijd van de overige aanwezige krachten, die dit project ondersteunde, willen vragen. Dit te kort schieten bracht het project niet in gevaar, maar had misschien een completer prototype op doen leveren. Echter zijn we, zoals eerder gemeld, toch tevreden over eind resultaat van dit stage project. Dit voornamelijk gezien het feit dat het prototype voldoet aan de gestelde eisen. Maar zoals ook eerder genoemd, was het een wens om toch een fractie meer te produceren.

11. Recommendation

1. Een programmeur minimaal 2 dagen per week op het vervolg project zetten, deze programmeur dient enige ervaring te hebben met PHP en object georiënteerd programmeren. Verder moet hij zich goed verdiepen in de mogelijkheden van extJS en JavaScript.
2. extJS framework versie kiezen voor company wide use, extJS versies verschillen vooralsnog te veel, zo is er geen backwardscompatibility tussen versie 1 en 2. Echter als we de uitstraling van extJS mogen geloven zal dit voortaan beter zijn. Maar is het niet onverstandig om net als in andere gevallen een formele keuze te maken voor een versie.
3. design/thema kiezen en naar huisstijl aanpassen, het design van het product dient nog sterk verbeterd te worden. Daarnaast is het noodzakelijk dat de standaard extJS look and feel aangepast wordt naar een stijl die binnen TAG software gebruikelijk is.
4. De gebruikte maps- en bestandsstructuur van eForms is een sterke verbetering t.o.v. oudere producten van TAG. Deze structuur hoeft niet letterlijk overgenomen worden, maar een afgeleide hiervan zou formeel gedefinieerd moeten worden als een richtlijn voor toekomstige producten/versies van producten van TAG software
5. SOAP interface, gebruik makende van nusoap[8]
6. Uitzoeken wat de meeste natuurlijke wijze van gebruik is, we hebben gemerkt dat de huidige wijze van het opstellen van een formulier, niet altijd even natuurlijk is voor nieuwe gebruikers.
7. TAG eForms moet uiteindelijk in 5 talen beschikbaar zijn, de huidige wijze (binnen TAG) om producten te vertalen is voor de vertalers ideaal, maar creëert een overhead op de betrokken systemen, die eigenlijk getackeld zou moeten worden.
8. extJS Internet Explorer problemen moeten opgelost worden voordat het product uitgerold kan worden
9. Verder onderzoeken van de mogelijkheden van Simpletest voor het functioneel testen van de software. Onderzoeken of Simpletest kan worden geïntegreerd met het ticket systeem dat bij TAG wordt gebruikt voor het afhandelen van bugs.

12. Reflectie

Hier willen wij kort de punten beschrijven die wij anders gaan doen als we de volgende keer een dergelijk project uitvoeren. Ook willen we kort belichten wat we positief ervaren hebben.

1. Zoals gebleken is, hebben wij het eForms project met twee studenten uitgevoerd en 4 mensen van TAG software. Echter merken we dat twee personen eigenlijk net te weinig is voor de manier waarop wij gewerkt hebben en deze korte periode. Wij hadden liever wat meer van het product opgeleverd. Echter hebben wij ons wel aan de minimale planning gehouden. Maar onze wens als software engineer in spe is toch ook om productief te zijn. Het project team vanuit TAG was wel aanwezig, wij hadden meer van hun tijd moeten op eisen voor ons project, hier was natuurlijk niet altijd de ruimte voor, maar bij een volgend project zullen ook hier duidelijkere afspraken over gemaakt gaan worden.
2. Eerder werd al gesproken over extJS, dit is een javascript library voor het ontwikkelen van webapplicaties. Deze library heeft een bepaalde leercurve waarbij het enige tijd kost om het principe onder de knie te krijgen. Maar eenmaal bekent met de mogelijkheden en enige ervaring met het vinden van de juiste informatie, is het een tool waarmee zeer eenvoudig veel bereikt kan worden. Wij raden deze library aan voor iedereen om te overwegen voordat begonnen wordt met een webapplicatie. Voor mobiele interfaces is deze tool minder geschikt. En zoals in de Recommendation al gezegd werd, zijn niet alle mogelijkheden die in extJS tot je beschikking staan geschikt voor Internet Explorer. Echter mogen we verwachten dat dit op termijn verbeterd zal worden, dan wel aan de kant van extJS, dan wel middels meer uniformiteit in de gebruikte JavaScript engines.
3. Het gebruikt van JSON voor asynchrone communicatie is heel eenvoudig, maar naar onze mening minder elegant dan bijvoorbeeld XML en afgeleiden van XML. Maar het is een enorm gebruiksvriendelijke notatie en zeer goed te combineren met de XML varianten die eleganter zijn. Wij zijn dan ook van mening dat JSON het ideaal is om object,arrays en primitieven over te brengen van de server naar de client. De wijze waarop kan echter eleganter dan in eForms gebruikt is.
4. Naar onze mening is het schrijven van artikelen aan het begin van het project een welkome toevoeging. De enige opmerking die we er wel bij zouden willen plaatsen is dat het naar onze mening beter zou zijn de artikelen in groepjes van minimaal 2 personen te schrijven. Je houdt elkaar scherp en het zal de diepgang van de artikelen ten goede komen.
5. Wij hadden in het plan van aanpak al gekozen om geen logboek bij te houden. Achteraf denken wij dat het beter was geweest om wel een logboek bij te houden. Het is dan makkelijker om terug te vinden wie wat wanneer gedaan heeft. Zeker als het gaat om wijzigingen in het ontwerp dan zou het handig zijn om deze terug te kunnen vinden.
6. Wij hadden ons in het plan van aanpak een nummering (verwijzing) voorgenomen die een verband zou moeten scheppen tussen de requirements, ontwerp en code. In de praktijk is dit niet gebeurd, in het vervolg zouden wij dit toch wel beter willen doen om het zo makkelijker te

maken voor de opvolgers. De rede dat dit niet gebeurd is, is dat de naamgeving in object en methoden dusdanig overeenkomt met de namen in de requirements dat het in eerste instantie overbodig leek. Echter zien we dat het toch veel zal toevoeging voor toekomstig onderhoud als er eenvoudig op de requirements nummering gezocht kan worden om alle code terug te vinden die met een bepaald onderdeel te maken heeft.

13. Merknamen

Microsoft - i.c.m. Word en Excel zijn dit gereistreerde merknamen van Microsoft Inc.
SQL Server - is een product van Microsoft Inc.
MySQL - is een eigendom van SUN microsystems Inc.
Zend - en de bijbehorende programma's zijn eigendom van Zend Technologies Ltd.
extJS - is een product van Ext company
JavaScript - is een handelsmerk van Sun Microsystems
Internet Explorer - is een product van Microsoft Inc.
Firefox - is een product van Mozilla

14. Appendix

A ontwerp
A.1 Plan van aanpak

Plan van aanpak
BSc Project Software Technologie
eForms

Vincent Simon van Beek	1217348
Mark Zijdemans	1217771

Inleiding
Dit document beschrijft de kaders waarbinnen het project wordt uitgevoerd. Na goedkeuring van dit document zal het gezien worden als werkproduct (artifact).

Het project
Het BSc stage project zal uitgevoerd worden bij TAG software & consultancy BV. TAG heeft een project gedefinieerd dat moet leiden tot een off-the-shelff product. Het project omvat het ontwikkelen van een web-applicatie waarmee formulieren op een interactieve manier te maken en te gebruiken zijn binnen een professionele organisatie. Het doel hier van is om papieren formulieren te vervangen door digitale equivalenten.

Externe klanten
Externe klanten zullen bij de ontwikkeling van de prototype release van dit project niet direct betrokken worden. TAG heeft een aantal lead users (klanten) gekozen (in overleg) waar de prototype release getest en beoordeeld zal worden. Bevindingen van deze klanten zullen meegenomen worden bij het ontwikkelen van de BETA release.
De BETA release zal buiten de scope van het BSc stage project liggen.

Interne experts
Bij de ontwikkeling zal van de prototype release zal worden samengewerkt met een aantal experts binnen TAG. Deze experts zullen vooral betrokken worden bij het opstellen van de requirements analyses. De volgende experts zullen betrokken zijn bij het project.

René ten Oever	Directeur	Project initiator
Auke van der Endt	Accountmanager	Customer knowledge
Dirk Jansen	Senior consultant	Customer knowledge
Emiel van Goor	Sr. Software Engineer	technische kennis
	IT coordinator	Ervaring ontwikkeling kwaliteitsmanagement-software

De informatie die nodig is om tot een goed requirements document te komen zal verzameld worden door middel van een open gesprek (brainstorm sessie) met de interne experts. Dit gesprek zal bestaan uit een presentatie van de algemene visie door VSvB en MZ, gevolgd door een brainstorm sessie. Een verslag van dit gesprek zal worden gemaakt door VSvB en MZ. Dit verslag zal aan alle deelnemers van het gesprek worden voorgelegd ter controle. Eventuele incorrectheden zullen dan nog worden verwerkt, dit zal leiden tot een definitief document. Uit dit document zal een puntenlijst worden gedestilleerd die zal leiden tot de requirements definitions.

Documenten
Alle documenten zullen opgesteld, bijgehouden en opgeslagen worden in doc formaat.
Daarnaast zullen alle bestanden ge-exporteert worden naar PDF voor controle, goedkeuring en externe archivering. Alle documenten zullen voorzien worden van een versie nummer en de status van het document. Documenten kunnen zich in de volgende statussen bevinden: opstellen, controleren, rectificeren en definitief.

De volgende documenten zullen worden opgeleverd.

- Plan van aanpak (PvA)
- Project omschrijving (door TAG)
- requirements analyse (incl. verslag eerder genoemde open gesprek)
- use case models
- UML models
- sequence diagrams
- technical feasibility
- omschrijving ontwikkel methode.

Repository
Als repository gebruiken we SVN, zowel de code als de documenten zullen in de repository worden opgeslagen.
Structuur SVN repsitory:

- source code (werkende releases)
- test code (commits tijdens het ontwikkelen)
- documenten
    - zie bovenstaande
    - handleiding
    - trainingen
    - installatie manuals
    - TAGwiki export over eForms

Code documentatie
Comments, en code zal worden opgemaakt volgens de TAG coding standards. Commentaar bij functies en objecten zal in PHP volgens de PHPdoc stijl gebeuren, in functies en objecten in JavaScript zullen volgens de JSdoc stijl gebeuren.
Voor verdere uitwerking hiervan zie bijlage 1 en bijlage 2.

- PHPdoc.org
- JSdoc.sourceforge.net

Nummering / koppeling ontwerp en code
Module.functionaliteit.[sub].[subsub].[...].requirementlink(alpha)
UI.1.2.3.a.b
Een uitgebreid voorbeeld is te vinden in bijlage 3.

Ontwikkel software
Als programmeer omgeving gebruiken we Eclipse en Textmate. Om Eclipse PHP en JS code goed te laten weergeven gebruiken we twee plugins, PDT en JSeclipse
Om verbinding te maken met SVN gebruiken we zowel de client software als de Eclipse plugin Subclipse (http://subclipse.tigris.org/).

Testen
Het testen van de software zal voornamelijk gebeuren door user testing. Er zal dus veel op functioneel niveau getest worden. Om er voor te zorgen dat er zo weinig mogelijk fouten optreden zullen we complexe functies en algoritmen met Unit test testen. Na het opstellen van de UML documenten zal er een testplan geschreven worden. Dit plan zal beschrijven welke functies met Unit testing getest zullen worden en welke met user testing. Het plan zal verder beschrijven hoe deze testen uitgevoerd dienen te worden, wat de input moet zijn en wat de resultaten van de tests moeten zijn.

Visual Design
De userinterface moet voldoen aan de TAG huisstijl. DH zal er op toezien dat dit gebeurd en zal ondersteuning bieden mocht dit nodig zijn. Deze ondersteuning moet gezien worden in het aanleveren van images, css code, html code en kleur/stijl schema's.

Logboek
Er zal geen logboek worden bijgehouden. Er zal wekelijks een rapportage gemaakt worden over de vorderingen. Deze rapportage zal een lijst bevatten met de voor die week geplande activiteiten en de daadwerkelijk uitgevoerde activiteiten.

Planning
Voor de planning van dit project zullen twee modellen worden gehanteerd, namelijk het MoSCoW model en het RUP model. Er zal een MoSCoW document worden opgesteld over het gehele project en per module die met RUP zal worden gepland. Wij hebben gekozen voor RUP omdat dit ons de mogelijkheid geeft een interne tester van TAG bij het project te betrekken voor user testing.

Glossary
Tabel met veel gebruikte afkortingen

VSvB
Vincent Simon van Beek
MZ
Mark Zijdemans
DH
Danny Hoogendoorn
MoSCoW
- M - MUST have this.
- S - SHOULD have this if at all possible.
- C - COULD have this if it does not affect anything else.
- W - WON'T have this time but WOULD like in the future.

RUP
Rational Unified Process
PDT
PHP Development Tools
PHP
PHP Hypertext Preprocessor
JS
JavaScript

Bijlagen

bijlage 1: PHPdoc

phpDocumentor tags

How to use tags in DocBlocks

by Gregory Beaver
Tag Documentation written by cellog@php.net
Copyright 2002, Gregory Beaver
Tags Manual
Welcome to the phpDocumentor Tags Manual

phpDocumentor tags are very similar to tags for the JavaDoc tool for Sun's Java Programming Language. Tags are only parsed if they are the first thing on a new line of a DocBlock. You may use the @ character freely throughout documents as long as it does not begin a new line. An example:

```
1./**
2. * tags demonstration
3. * @author this tag is parsed, but this @version tag is igno
     red
4. * @version 1.0 this version tag is parsed
5. */
```

Any tags that phpDocumentor does not recognize will not be parsed, and will be displayed in text flow as if they are part of the DocBlock long description. The example below displays text "tags demonstration @foobar this is silly," and also displays the author as "this tag is parsed, but this @version tag is ignored"

```
1./**
2. * tags demonstration
3. * @foobar this is silly
4. * @author this tag is parsed, but this @version tag is igno
      red
5. */
```

Inline tags display in the text flow where they appear, and are not separated from the description. As of version 1.2, there are several inline tags. The allowed inline tag list is different for tutorials and regular in-code documentation. See phpDocumentor Inline tags for more information.

The example below will display the text "this function works heavily with **foo()** to rule the world" where **foo()** represents a hyperlink that links to the function foo()'s documentation.

```
1./**
2. * inline tags demonstration
3. *
4.
      * this function works heavily with {@link foo()} to rule
      the world
5. */
6.function bar(){
7.}
8.function foo(){
9.}
```

Standard phpDocumentor Tags

Most tags can be in the DocBlock of any documentable PHP element (for a definition of a documentable PHP element, see Documentable PHP Elements). Here is a list of the standard tags:

@access

If @access is private, the element will not be documented unless specified by command line switch --parseprivate.
@author

The @author tag is used to document the author of any element that can be documented (global variable, include, constant, function, define, class, variable, method, page). phpDocumentor will take any text between angle brackets
(< and >)
and try to parse it as an email address. If successful, it will be displayed with a mailto link in the page

NEW v1.2 - @author is now inherited by child classes from a parent class, see inline {@inheritdoc}.
@copyright

The @copyright tag is used to document the copyright information of any element that can be documented (global variable, include, constant, function, define, class, variable, method, page). phpDocumentor will display the copyright string unaltered.
NEW v1.2 - @copyright is now inherited by child classes from a parent class, see inline {@inheritdoc}.
@deprecated

The @deprecated tag is used to document the deprecation version or other information of any element that can be documented except for page (global variable, include, constant, function, define, class, variable, method). If present, phpDocumentor will display the optional version/info string unaltered.
Use @deprecated to notify users of deprecated elements that should not be used any longer
@example

The example tag can be used to parse an example file for syntax highlighting and linking to documentation. This versatile tag attempts to read the file from the full path specified, and will accept any path that http://www.php.net/fopen will accept. phpDocumentor checks the path to ensure that the file retrieved has a valid .php extension as defined in phpDocumentor.ini, and then opens the file. It will parse the file, and output syntax-highlighted source with line numbers, links to documentation and will then add a link to the documentation to that file.

If given an absolute path, phpDocumentor will not search for the example file any further. If given a relative path (no leading c:\ or /) phpDocumentor searches for examples files first in the directory specified by the -ed, --examplesdir command-line, if present. As of phpDocumentor 1.2.1, it will next search for the file in an examples/ subdirectory of the current file's directory. Otherwise, it will search for a subdirectory named "examples" in the top-level parsing directory, and if not found, in the top-level directory.
The top-level parsing directory is simply the directory that every file parsed has in common.
The @filesource tag serves a similar purpose, but instead of parsing a separate file, it parses the current file's source.
To do an inline code example, use the html tag <code> or the new inline {@example} tag

@ignore

Use @ignore to prevent phpDocumentor from documenting an element, such as a duplicate element.
@internal

Use @internal to facilitate the creation of two sets of documentation, one for advanced developers or for internal company use, and the other for the general PHP public. Use this tag or its cousin, inline {@internal}}
@internal responds to the command-line -pp, --parseprivate, as both @internal and --parseprivate share the same purpose. In both cases, the intent is to allow the creation of two sets of documentation, one for public use, and the other for a subset of developers, either the project developers or those who wish to extend the project. In essence, it allows the creation of both user-level and programmer-level documentation from the same source, and is one of phpDocumentor's most useful features.
@link

You may use the @link tag to document any element (include, page, class, function, define, method, variable). The guidelines for giving the element name to @link are the same as outlined in the @see manual page.

Unless linking to an element, @link assumes the arguments passed are fully-formed URLs. Generally speaking, if you want to link to an element's documentation, use @see or inline {@link}... you can use @link, but the other options are better.
URL is any valid Uniform Resource Locator (http://www.example.com, telnet://example.com, ftp://ftp.example.com, mailto:email@example.com, etc.)
Note that as of version 1.2.0, it is possible to specify a short description for the link to use as the link text instead of the URL.
@see

The @see tag may be used to document any element (global variable, include, page, class, function, define, method, variable)
@see only displays links to element documentation. If you want to display a hyperlink, use @link or inline {@link}
New in version 1.2: You can link to any defined function in the current php version using the function's name. This linking is done using the standard php function http://www.php.net/get_defined_functions, and so relies on the version of php that is used to execute phpDocumentor. A benefit of this method is that the function highlighting will automatically upgrade with a php upgrade without any change to the underlying code. You may also link directly to any portion of the php website using the fake package override PHP_MANUAL (as in PHP_MANUAL#get_defined_functions, or PHP_MANUAL#support.php)
Along with inline {@link}, the @see tag is among the most useful features of phpDocumentor. With this tag, you can create a link to any element (except include/require) in the documentation with a very wide range of options. The @see parser can be told exactly where to look using some basic punctuation:

- :: -- This is the class scope override. Use it to tell the @see parser which class to look in for the element specified, like classname::methodname() or classname::$variablename.
- () -- When present at the end of elementname, like elementname(), this tells the @see parser to search for a function or method.
- $ -- When present at the beginning of elementname, like $elementname, this tells the @see parser to search for a variable in the current class.

However, @see is also intelligent enough to recognize shorthand. If @see receives an elementname with no punctuation, it will search for an element in this order:

1. is elementname the name of a class?
2. is elementname the name of a procedural page? (file.ext)
3. is elementname the name of a define?
4. if the DocBlock containing the @see is in a class:
    1. is elementname a method?
    2. is elementname a variable?
5. is elementname a function?

@see parsing is slightly slower when passed an elementname with no punctuation, especially if the elementname is a function, so use it sparingly in large projects (500+ elements with @sees in their DocBlocks). The best use for punctuation-less elementname is in a project whose classnames are in flux.
@since

The @since tag may be used to document the release version of any element that can be documented (global variable, include, constant, function, define, class, variable, method, page). phpDocumentor will display the version/info string unaltered.
Use @since to document revisions, as in "This function has been a part of this package since version 2.0"
@tutorial

The @tutorial tag may be used to document any element (global variable, include, page, class, function, define, method, variable)

@tutorial only displays links to tutorials/extended documentation. If you want to display a hyperlink or link to a documented element, use @see or @link or inline {@link}
Along with inline {@tutorial}, the @tutorial tag is among the most useful features of phpDocumentor. With this tag, you can create a link to any tutorial or extended documentation. The @tutorial parser can be told exactly where to look using a format similar to a URI:

- package/ -- Preface to specify a tutorial in a specific package
- subpackage/ -- Preface to specify a tutorial in a specific package/subpackage
- #section -- Link to a specific section of the documentation package/subpackage/tutorial.ext#section
- .subsection -- Link to a specific sub-section of the documentation package/subpackage/tutorial.ext#section.subsection

However, @tutorial, like @see, is also intelligent enough to recognize shorthand. The search order for a tutorial is identical to what is specified in @see
@version

Documents the version of any element, including a page-level block
NEW v1.2 - @version is now inherited by child classes from a parent class, see inline {@inheritdoc}.
inline {@internal}}

Unlike other inline tags, {@internal}} may contain other inline tags due to its purpose. To terminate an {@internal}} inline tag, you must use two ending brackets "}}"
Use {@internal}} to facilitate the creation of two sets of documentation, one for advanced developers or for internal company use, and the other for the general PHP public. Use this inline tag or its cousin, @internal
inline {@inheritdoc}

The {@inheritdoc} inline tag is used in the DocBlocks of classes, methods and class variables of child classes. phpDocumentor will automatically inherit the @author tag, @version tag, and @copyright tag from a parent class. In addition, if there is no documentation comment present, it will inherit the parent's documentation.

Note that if the {@inheritdoc} inline tag is not present, and a child class is undocumented, the child class will still attempt to inherit documentation from the parent class as is. {@inheritdoc} allows flexibility of where to put documentation from the parent class in a child class's documentation.
inline {@link}

The inline {@link} tag is used to document any element (page, global variable, include, class, function, define, method, variable).
New in version 1.2: You can link to any defined function in the current php version using the function's name. This linking is done using the standard php function http://www.php.net/get_defined_functions, and so relies on the version of php that is used to execute phpDocumentor. A benefit of this method is that the function highlighting will automatically upgrade with a php upgrade without any change to the underlying code. You may also link directly to any portion of the php website using the fake package override PHP_MANUAL (as in PHP_MANUAL#get_defined_functions, or PHP_MANUAL#support.php).
inline {@link} displays links directly in the natural text flow of a DocBlock. If you want to display links prominently after descriptive text, use @see or @link.
For parameter information, see @see or @link. The guidelines for giving the element name to inline @link are the same as outlined in the @see manual page.

Usage

The general idea of JSDoc is to pick up the form and documentation of code. A major part of JSDoc's functionality is based around documentation strings embedded in JavaScript code. The definitive reference to writing documentation strings for Java (with a large amount of carry-over to JSDoc) can be found at the javadoc reference page.

JSDoc currently supports a subset of javadoc's '@'-prefixed attributes, but the parser is customizable (as explained later). An example JavaScript file (test.js) is included with the JSDoc distribution. Most functionality of JSDoc can be seen within this script file, however for the impatient a small (and incomplete) example of a common usage of JSDoc is shown below

```
/**
  * Shape is an abstract base class. It is defined simply
  * to have something to inherit from for geometric
  * subclasses
  * @constructor
  */
function Shape(color){
     this.color = color;
}

// Bind the Shape_getColor method to the Shape class
Shape.prototype.getColor = Shape_getColor;

/**
  * Get the name of the color for this shape
  * @returns A color string for this shape
  */
function Shape_getColor(){
     return this.color;
}

/**
  * Circle is a subclass of Shape
  */
function Circle(radius){
     this.radius = radius;
}

/**
  * A very rough value for pi
  */
```

```
Circle.PI = 3.14;

/**
  * Get the radius of this circle
  * @returns The radius of this circle
  */
function Circle_getRadius(){
     return this.radius;
}

// Circle is a subclass of Shape
Circle.prototype = new Shape(null);
```

One important difference between javadoc and JSDoc is that JSDoc deals with a much more dynamic language that javadoc. For example, in JavaScript there are many different ways to make one class a subclass of another. The JSDoc parser is able to catch some of these methods, but for particularly dynamic scripts the parser will not be able to do it all on its own. For that reason, JSDoc is customizable with a file called .jsdoc_config that resides in the JSDoc install directory. In this configuration file, JSDoc's behaviour can be customized for any kind of '@'-prefixed attribute. Although this configuration file actually contains Perl code, simple customizations can be done by someone with minimal experience with Perl.

JSDoc also uses a several non-javadoc '@'-attributes: `@constructor` and `@private` are two of the most important ones. As noted above, JavaScript has the capability to be a much more dynamic language than Java. The JSDoc parser can usually find class constructors on it's own, but there are some situations when it needs some additional assistance. For this reason, it is a good practice to always include the `@constructor` tag in the documentation for a class constructor, as shown below:

```
/**
  * Nothing is a class that represents nothing
  * @constructor
  */
function Nothing(){
     // ... initialization ...
}
```

The `@private` attribute simply displays the marked method as being private.

Tag Reference

`@param`
Provide information about a function parameter. A datatype indicator can be added between curly braces with this tag, as follows:

```
        /**
         * @param {String} paramName This is a string parameter
         */
```

`@argument`
Synonym for `@param`
`@return`
Provide information about the return value of a function.
`@returns`
Synonym for `@return`
`@author`
Provide information about the author of a JavaScript file or a function.
`@deprecated`
Signify that a function or class is deprecated, and should not be used if possible.
`@see`
Link to another class or function that is of importance to the current class or function. This tag can take several forms.
To link to another method within the same class:

```
@see #methodName
```

To link to another class:

```
@see ClassName
```

To link to a specific method of another class:

```
@see ClassName#methodName
```

`@version`
Show the version number of the current file or class
`@requires`
Define a dependency upon another class. The syntax for this tag is as follows:

```
@requires OtherClassName This is text to be shown
```

`@throws`
Show that a method can throw a certain type of exception. The syntax for this tag is:

```
@throws ExceptionType This is the label text
```

`@exception`
Synonym for `@throws`
`@link`
This is a powerful tag that can be used to link to a large number of other pages. It is also the only tag that can be used in the description text of a documentation string before the '@'-tag section. The usage is very similar to that of the `@see` tag, but the entire tag is wrapped in curly braces. For example:

```
/**
 * This utility method is also a member of the {@link
 * String} class,
 * in the form of the {@link String#utility} method.
 */
```

`@fileoverview`
This is a special-use tag. If the first block of documentation in a file starts with a `@fileoverview` tag, the rest of the documentation block will be used to provide a file overview in the documentation.
`@class`
This tag is used in a constructor's documentation block to provide information about the actual class. The included documentation will then not be included in the constructor's documentation.
`@constructor`
Signify that a function is the constructor for a class.
`@type`
Show the return type of a function. For example:

```
/**
 * This function returns a String.
```

```
         * @return The name of the current user
         * @type String
         */
```

`@extends`

Used to show that a class is a subclass of another class. JSDoc is often quite good at picking this up on its own, but in some situations this tag is required.

`@private`

Signify that a function or class is private. Private classes and functions will not be available in the documentation unless JSDoc is run with the `--private` commandline option.

`@final`

Flag a value as being a final (constant) value.

`@member`

Show that a function is a member of a given class:

```
        /**
         * @member MyClass
         */
      function SomeFunc(){
      }
```

`@ignore`

Tell JSDoc to totally ignore this method.

`@base`

Force JSDoc to view the current class constructor as a subclass of the class given as the value to this tag:

```
       /**
        * This is a subclass of Shape
        * @constructor
        * @base Shape
        */
       function Circle(){
           // ...
       }
```

`@addon`

Mark a function as being an "addon" to a core JavaScript function that isn't defined within your own sources, as shown below:

```
      /**
       * This is an addon function to SomeCoreClass which is
       * not defined within our own sources.
       * @addon
       */
      SomeCoreClass.someFunction = function(){
            // ...
      }
```

`@exec`

Experimental!

Force JSDoc to "execute" this method as part of its preprocessing step, in the same way that class contructors are executed. This can allow attributes to be added to a class from within a function.

bijlage 3: nummering van requirement -> code

Voorbeeld van de in dit project te gebruiken nummering:

Requirement definition

1. User management.
    1.1. Toevoegen van gebruikers
    1.2. Bewerken van gebruikers
    1.3. Verwijderen van gebruikers

Use-Case
1.1. Toevoegen van gebruikers

user
systeem
klik op toevoegen
return formulier
vul formulier in

klik op opslaan
valideren

opslaan user

return status

Datamodel (UML)

USER (core.1.1.)

-naam string
-password string
-add_user(naam, password)

PHP code

```
/**
  * Dit object implementeert core.1.1.
  * @param string $naam
  * @param string $password
  * @return bool
 /*
class User{
      private $naam = "";
      private $password = "";
      function add_user($naam, $password){
            $this->naam = $naam;
            $this->password = $password;
            return true;
      }
}
```

A.2 Requirement definition

Requirements definitions

BSc Project Software Technologie
eForms

Inleiding

Van de brainstorm sessie bij TAG is een verslag gemaakt. Dit verslag is voor rectificatie naar de deelnemers van het gesprek gestuurd. Na het rectificeren is dit verslag gebruikt als basis voor dit document.

Functional Requirements definitions

1. Formbehandeling
   - 1.1. Formulieren weergeven
     - 1.1.1. Workflow weergeven
   - 1.2. Formulier invullen
     - 1.2.1. Ingevuld formulier opslaan in de map “Concepten”
     - 1.2.2. Ingevuld formulier versturen naar een in de workflow gedefinieerd persoon
   - 1.3. Formulier opstarten
     - 1.3.1. Selecteer een toegankelijk formulier
     - 1.3.2. Formulier broadcasten
       - 1.3.2.1. Verzend datum/tijd
       - 1.3.2.2. Selecteer ontvangers (wordt beperkt door de work-flow)
   - 1.4. Formulier verwijderen
2. Berichtboxen
   - 2.1. Lijst van mappen weergeven
     - 2.1.1. Er zijn 4 standaard mappen die niet verplaats of hernoemd kunnen worden, namelijk: postvak in, concepten, archief en prullenbak (de namen wijzigen wel mee in verschillende talen)
     - 2.1.2. Mappen sorteren
   - 2.2. Map maken
   - 2.3. Map bewerken
     - 2.3.1. Map naar een andere map verplaatsen
     - 2.3.2. Map verwijderen
     - 2.3.3. Map hernoemen
   - 2.4. Bericht naar een andere map verplaatsen
   - 2.5. Inhoud van een map weergeven
3. Printen
   - 3.1. Formulier printen
4. Formbuilder
   - 4.1. Overzicht van formulieren
     - 4.1.1. Per formulier een overzicht van voorgaande versies
   - 4.2. Formulieren toevoegen, bewerken en opslaan
     - 4.2.1. Tijdens het bewerken het formulier blokkeren voor anderen
     - 4.2.2. Bij het opslaan van bewerkingen een nieuwe instantie van het formulier maken

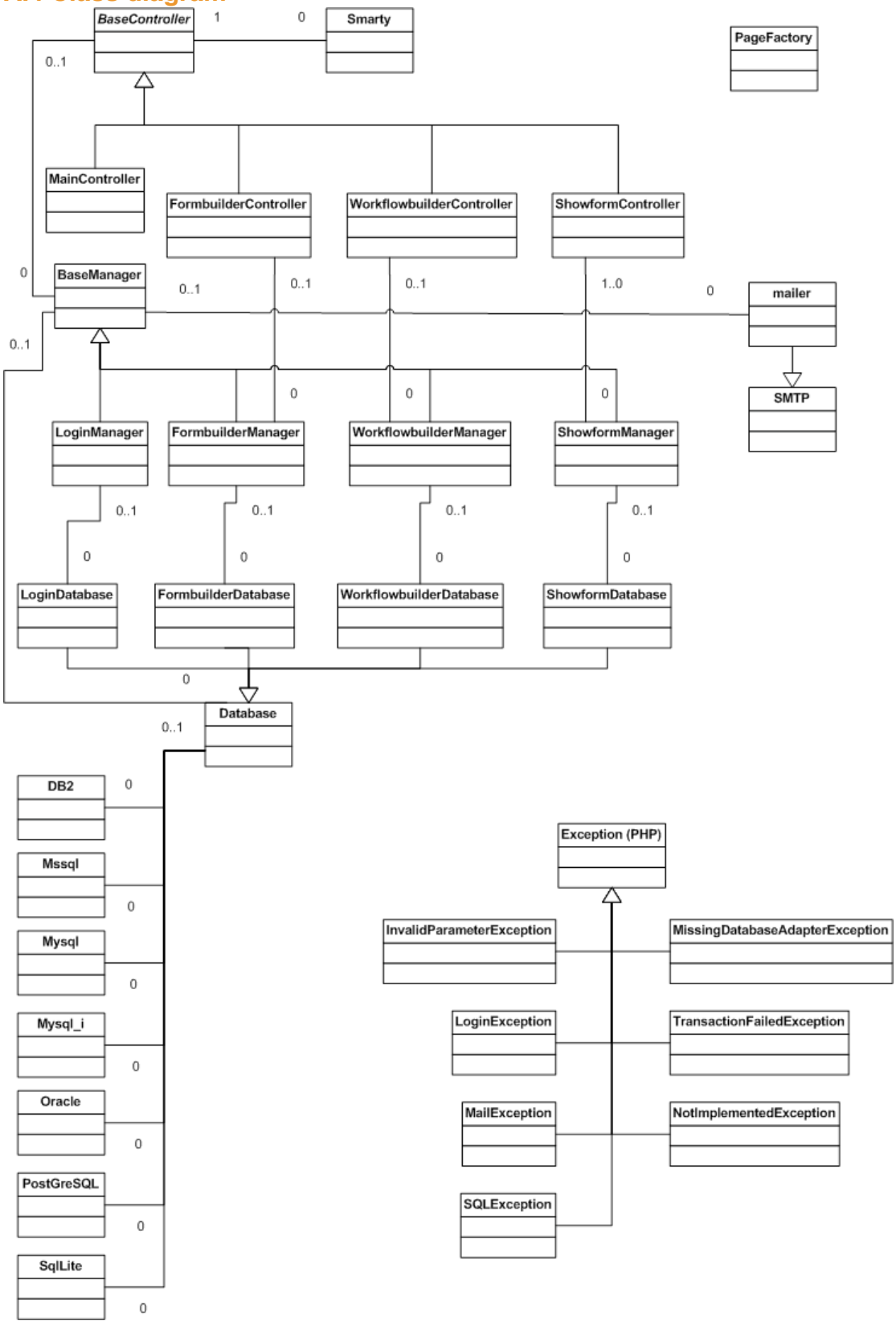
4.3. Velden toevoegen aan en verwijderen van een formulier of layout element
4.3.1. text-field toevoegen
4.3.1.1. eigenschappen instellen, label, size, minimum aantal karakters, maximum aantal karakters en de standaard waarde
4.3.2. radio-button toevoegen
4.3.2.1. eigenschappen instellen, label, value, standaard waarde (checked), groep
4.3.3. checkbox toevoegen
4.3.3.1. eigenschappen instellen, label, value, standaard waarde (checked)
4.3.4. drop-down menu toevoegen
4.3.4.1. eigenschappen instellen, label, size (aantal zichtbare items minimaal 1), datasource (ja/nee)
4.3.4.2. items toevoegen als er geen datasource is
4.3.4.2.1. eigenschappen item instellen, weergavenaam, value, selected
4.3.4.3. item verwijderen
4.3.4.4. item bewerken
4.3.4.5. Datasource kiezen als (datasource ja/nee ja is)
4.3.5. submit button toevoegen
4.3.5.1. eigenschappen instellen, label, value
4.3.6. reset button toevoegen
4.3.6.1. eigenschappen instellen, label, value
4.3.7. informatie (sommatie) veld toevoegen
4.3.7.1. eigenschappen instellen, label, standaardwaarde,
4.3.7.2. formule bewerken
4.3.8. standaard velden toevoegen
4.3.8.1. datum veld “reageer voor *datum*”
4.4. Velden bewerken
4.5. Layout element toevoegen aan een formulier
4.5.1. eingenschappen instellen, kolommen, rijen, vermenigvuldigbaar, sommatie rij (ja/nee)
4.6. Layout element verwijderen van een formulier
4.7. Validaties aan velden koppelen
4.8. Huisstijl layout kiezen
5. Workflow beheer
5.1. Overzicht van toegankelijke workflows
5.2. workflow selecteren
5.2.1. Workflow bekijken
5.2.1.1. overzicht van formulieren (4.1)
5.2.2. Workflow bewerken
5.2.2.1. initiator bewerken
5.2.2.2. Stap toevoegen/bewerken
5.2.2.2.1. ontvangers selecteren (groepen en individuen)
5.2.2.2.2. return (afkeur) stap bepalen
5.2.2.2.3. Formulier toevoegen/verwijderen
5.2.2.2.4. Data eerdere stappen weergeven
5.2.2.2.4.1. selecteer formulieren van eerdere stappen

5.2.2.3. Stap bewerken
5.2.2.4. Stap verwijderen
5.2.3. Workflow verwijderen
5.3. nieuwe workflow maken
5.3.1. Bepaal de initiator
5.3.2. Stap toevoegen
5.3.2.1. ontvangers selecteren (groepen en individuen)
5.3.2.2. return (afkeur) stap bepalen
5.3.2.3. Formulier toevoegen
5.3.2.4. Data eerdere stappen weergeven
5.3.2.4.1. selecteer formulieren van eerdere stappen
6. Rapportage
7. Backend
7.1. Configuratie scherm
7.1.1. taal instellen
7.2. Datasource beheer
7.2.1. overzicht van bestaande datasources
7.2.2. datasource maken
7.2.2.1. data importeren
7.2.2.2. SELECT query opstellen
7.2.2.3. value kolom kiezen
7.2.2.4. weergavenaam kolom kiezen
7.2.2.5. weer te geven velden defineren en kolommen er aan koppelen
7.2.3. bewerken van datasources
7.3. Validatie beheer
7.3.1. overzicht van bestaande validaties
7.3.2. maken van validaties
7.3.3. bewerken van validaties
8. Gebruikersbeheer
8.1. Overzicht van gebruikers
8.1.1. Filter mogelijkheden
8.1.2. Zoek mogelijkheden
8.2. Toevoegen van gebruikers
8.3. Bewerken van gebruikers
8.4. Importeren van gebruikers
8.5. Groepen overzicht
8.5.1. Filter mogelijkheden
8.5.2. Zoeken mogelijkheden
8.6. Groepen maken
8.7. Groepen bewerken
8.8. Gebruikers aan groepen koppelen
9. Security
9.1. Inloggen
9.2. Password wijzigen
9.3. Password recovery

A.3 Systeem eisen

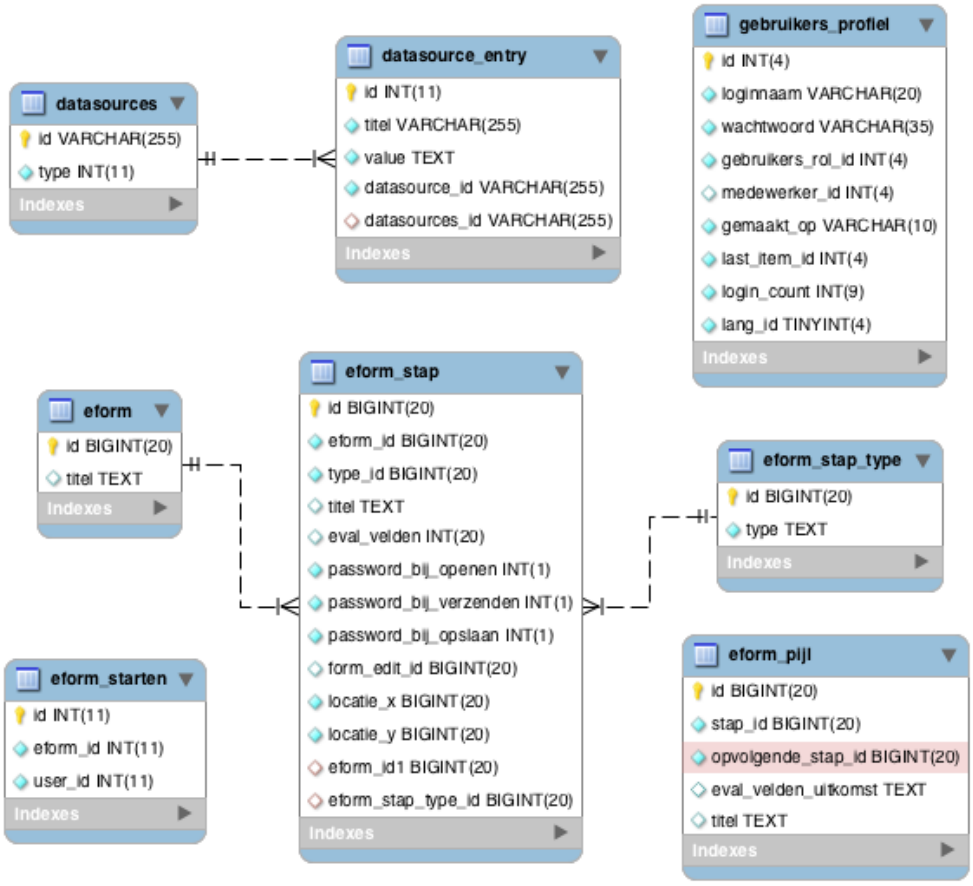
- PHP5
- MySQL 5
- MSSQL 2005
- Utf-8
- Pdf
- Tranactie ondersteuning SQL
- extJS
- XHTML transitional
- GD2
- IE 7/8
- FireFox
- Safari
- Opera
- Chrome
- Lettertypen:
- Arial
- Tahoma
- Verdana
- Plaintext invoer
- Afbeeldingen
- Png
- Jpg
- Gif
- Bmp
- Instellingen maken voor de ge-accepteerde up te loaden typen, bedrijf kan dus eisen dat enkel pdf's geupload worden

A.4 Class diagram

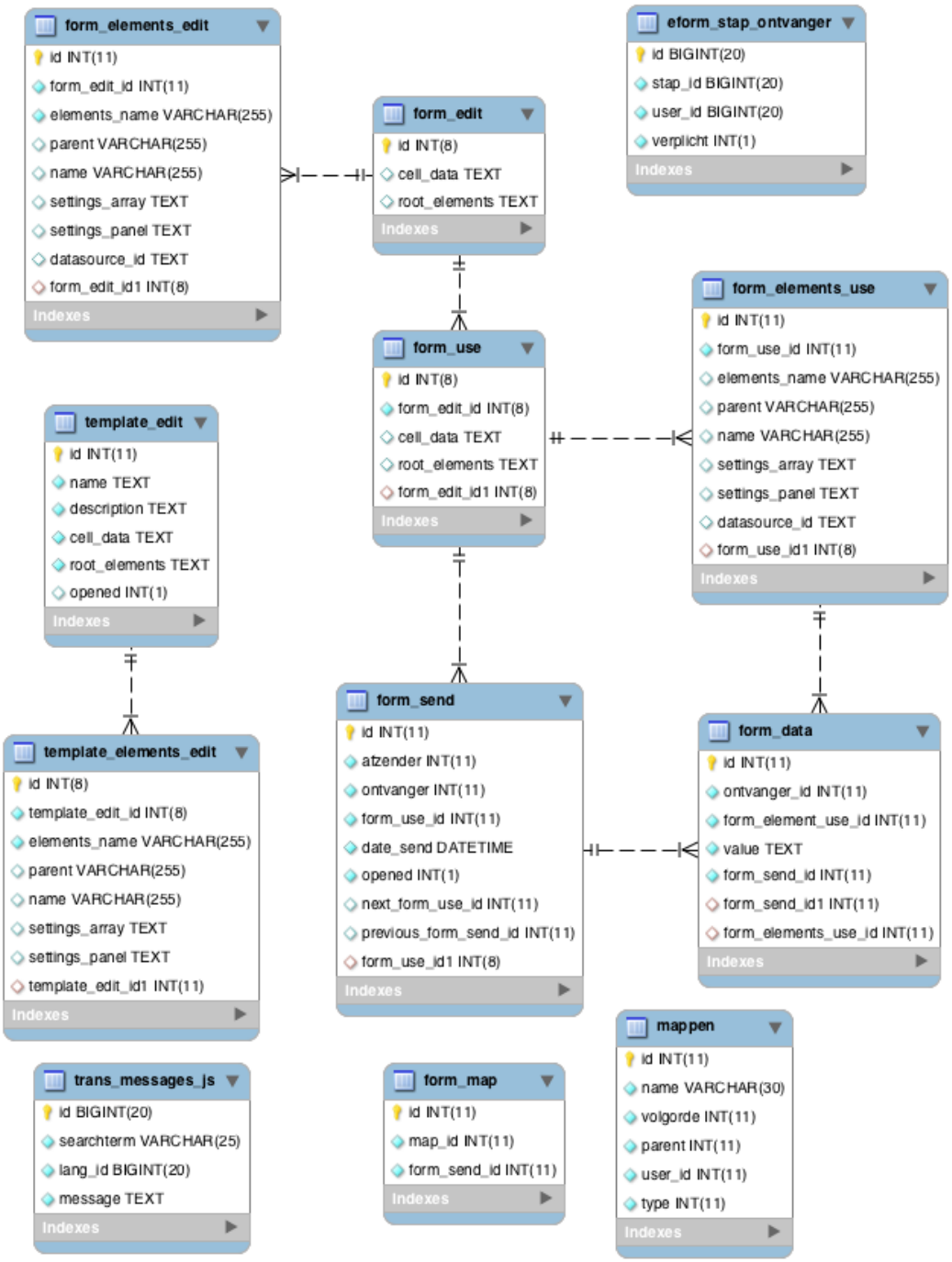


Figuur A.1. classdiagramm

A.5 Extended Entity-Relationship diagram



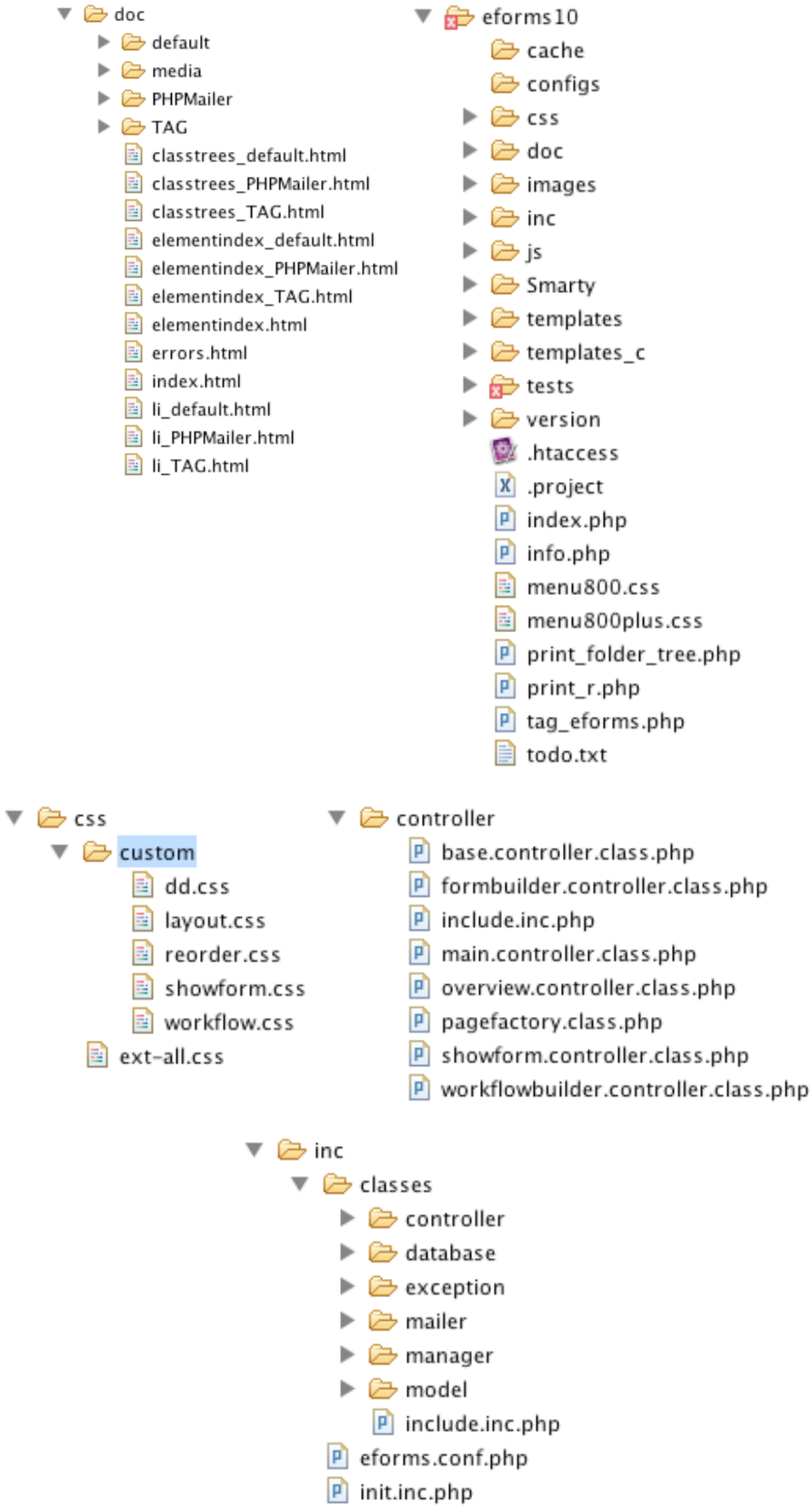
Figuur A.2. extended entity reliantionship diagram (deel 1)

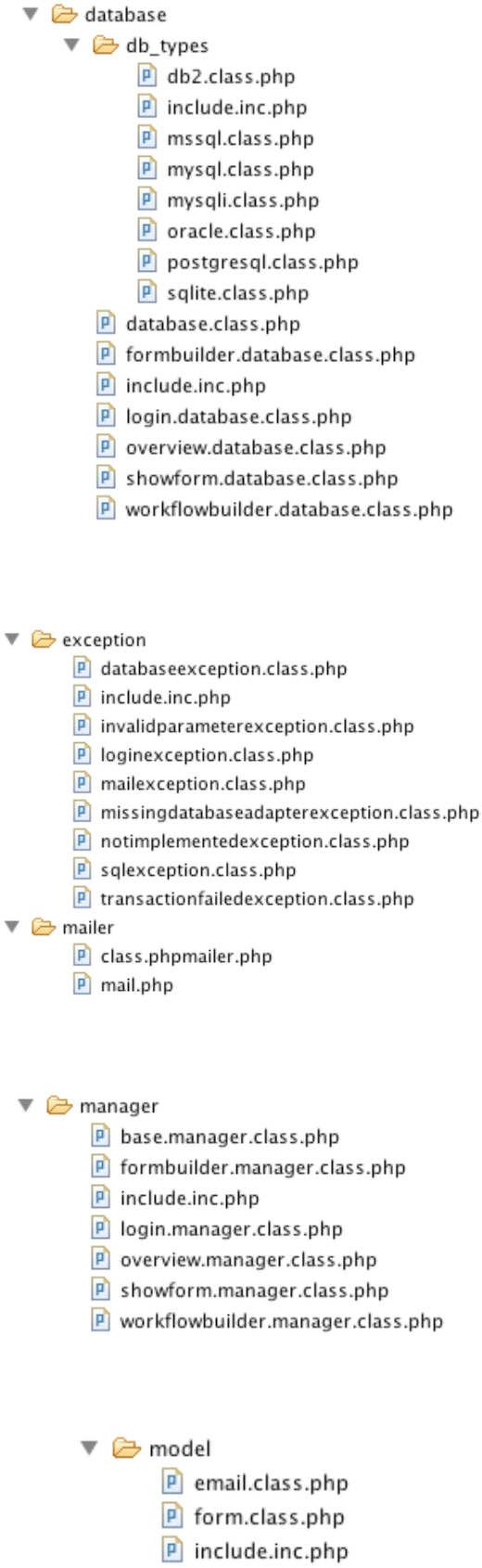


Figuur A.2. extended entity reliantionship diagram (deel 2)

B PHP skeleton

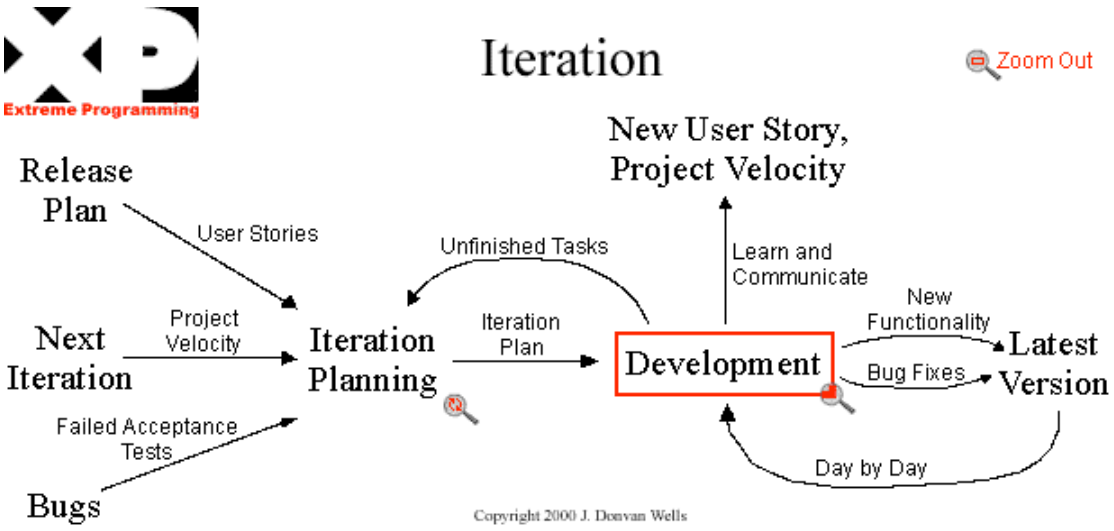
Hieronder vindt u de mapstructuur van de eForms applicatie.




database
db_types
db2.class.php
include.inc.php
mssql.class.php
mysql.class.php
mysqli.class.php
oracle.class.php
postgresql.class.php
sqlite.class.php
database.class.php
formbuilder.database.class.php
include.inc.php
login.database.class.php
overview.database.class.php
showform.database.class.php
workflowbuilder.database.class.php
exception
databaseexception.class.php
include.inc.php
invalidparameterexception.class.php
loginexception.class.php
mailexception.class.php
missingdatabaseadapterexception.class.php
notimplementedexception.class.php
sqlexception.class.php
transactionfailedexception.class.php
mailer
class.phpmailer.php
mail.php
manager
base.manager.class.php
formbuilder.manager.class.php
include.inc.php
login.manager.class.php
overview.manager.class.php
showform.manager.class.php
workflowbuilder.manager.class.php
model
email.class.php
form.class.php
include.inc.php

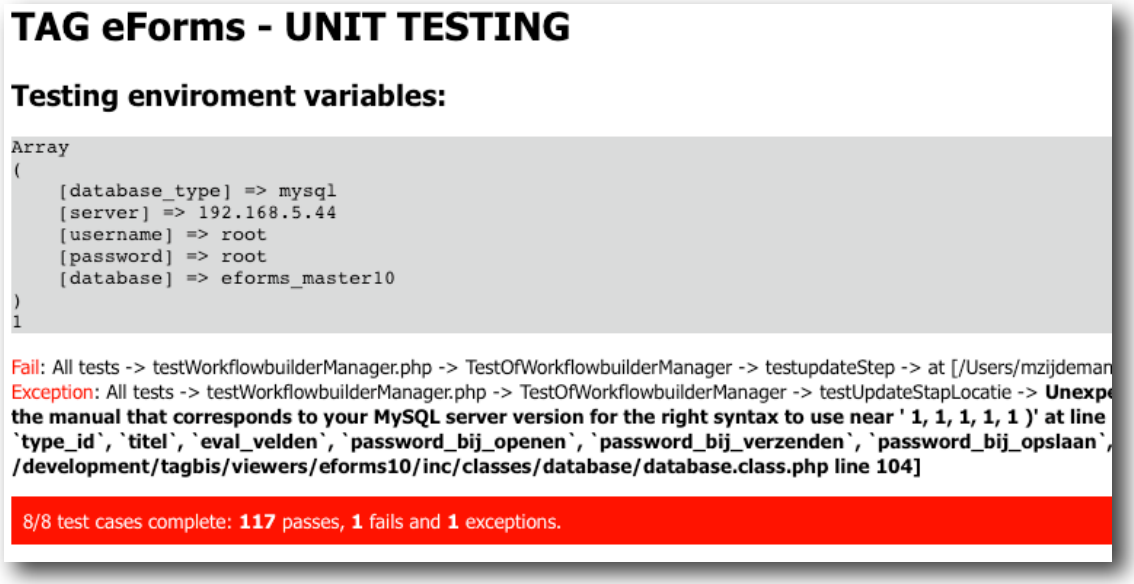
Figuur B.1. mappen en bestanden TAG eForms

C XP



Figuur C.1. iteratie schema extreme programming [9]

D Voorbeeld Unit Test resultaat



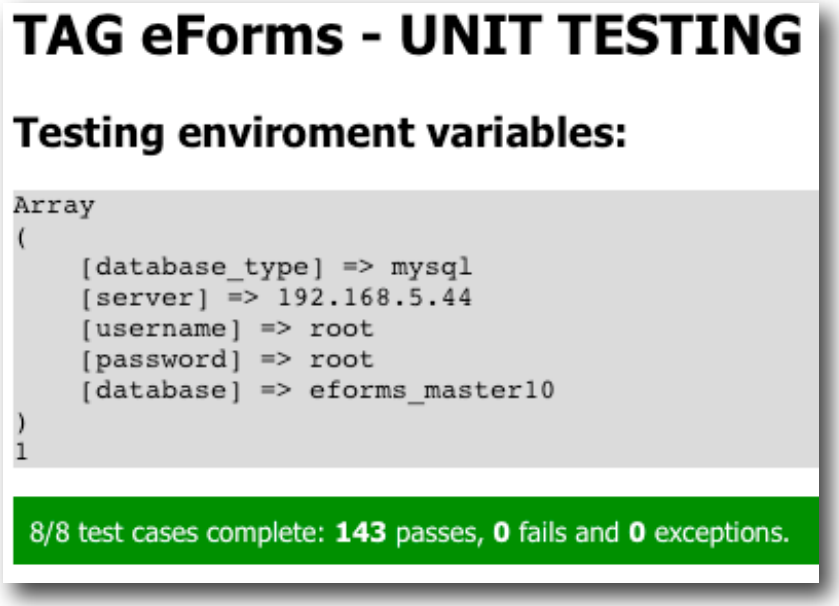
TAG eForms - UNIT TESTING

Testing enviroment variables:

```
Array
(
    [database_type] => mysql
    [server] => 192.168.5.44
    [username] => root
    [password] => root
    [database] => eforms_master10
)
1
```

Fail: All tests -> testWorkflowbuilderManager.php -> TestOfWorkflowbuilderManager -> testupdateStep -> at [/Users/mzijdeman
Exception: All tests -> testWorkflowbuilderManager.php -> TestOfWorkflowbuilderManager -> testUpdateStapLocatie -> **Unexpe**
the manual that corresponds to your MySQL server version for the right syntax to use near ' 1, 1, 1, 1, 1)' at line
`type_id`, `titel`, `eval_velden`, `password_bij_openen`, `password_bij_verzenden`, `password_bij_opslaan`,
/development/tagbis/viewers/eforms10/inc/classes/database/database.class.php line 104]

8/8 test cases complete: **117** passes, **1** fails and **1** exceptions.



TAG eForms - UNIT TESTING

Testing enviroment variables:

```
Array
(
    [database_type] => mysql
    [server] => 192.168.5.44
    [username] => root
    [password] => root
    [database] => eforms_master10
)
1
```

8/8 test cases complete: **143** passes, **0** fails and **0** exceptions.

E.Engelbart’s Unfinished Revolution [5]

Doug Engelbart, December 1968.

Engelbart's Unfinished Revolution

A Symposium at Stanford University

December 9, 1998

Try to imagine "personal" computing without the following::

- The mouse and pointer cursor
- Display editing
- Outline processing
- Multiple remote online users of a networked processor
- "Linking" and in-file object addressing
- Multiple windows
- Hypermedia
- Context-sensitive help

These features, which we take for granted in 1998, were unheard of before Doug Engelbart's inquiries into "Augmented Human Intellect" led to a revolutionary vision of the computer, a vision which was revealed to the computer world on December 9, 1968 ...

On that day Doug Engelbart and a small team of researchers from the Stanford Research Institute stunned the computing world with an extraordinary demonstration at a San Francisco computer conference. They debuted: the computer mouse, graphical user interface, display editing and integrated text and graphics, hyper-documents, and two-way video-conferencing with shared workspaces. These concepts and technologies were to become the cornerstones of modern interactive computing

On December 9th, 1998 Stanford University Libraries and the Institute for the Future presented a day-long, public symposium that brought together Engelbart and members of his historic team, along with other computer visionaries, to consider the impact of Engelbart's work on the last three decades of the computer revolution, to explore the challenges facing us today, and to speculate about the next three decades.

That landmark 1968 demonstration took place at the American Federation of Information Processing Societies' Fall Joint Computer Conference. At a time when computers were little more than huge number-crunchers, Engelbart and his team's introduction of their two-way interactive system, called NLS (for oN Line System) was a shock.

Today the mouse, graphical user interface, hyper-documents, display editing and integrated text and graphics are taken for granted. Other features of NLS, such two-way video-conferencing with shared workspaces, remain more vision than reality even today. And some of NLS' most important elements, such as the concept of bootstrapping, remain so novel that they are all but unknown to the current generation of systems designers.

NLS' 1968 demo was a watershed that fundamentally changed the trajectory of the computing revolution, contributing not only ideas, but also many of the people who would later build the systems we would use today. Alumni of the NLS project (and its successor, the Augment project) include many of the most influential figures in Silicon Valley (and elsewhere). Others credit Doug's work in general, and the 1968 demo in particular with influencing their design philosophies. As personal computing pioneer and visionary Alan Kay once observed, "I don't know what Silicon Valley will do when it runs out of Doug's ideas."

The first mouse

Comet in een mail-like web-applicatie

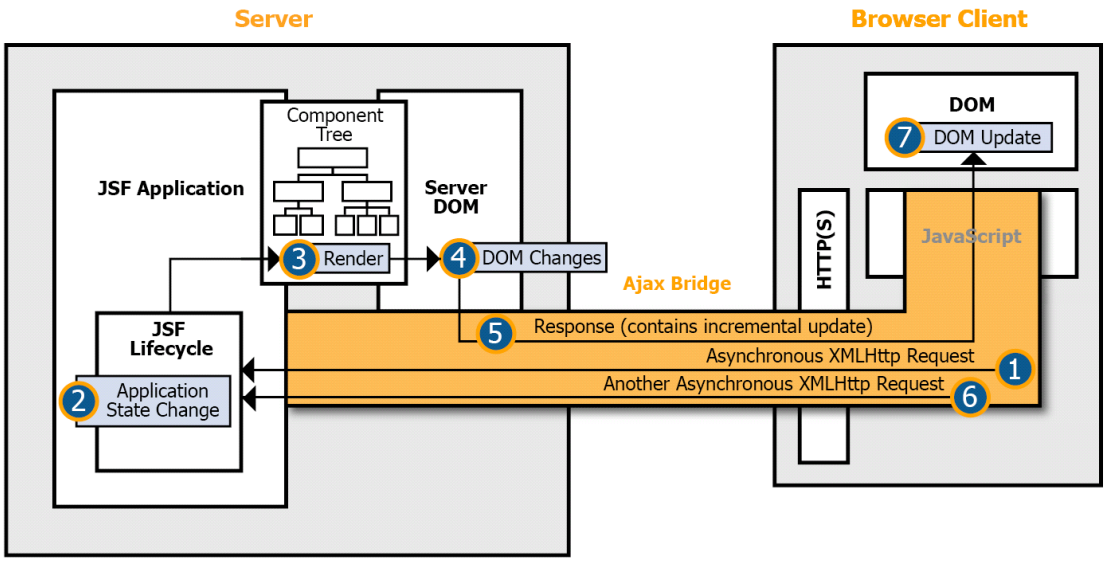
Mark Zijdemans

abstract
TAG is een bedrijf dat graag de modernste technieken beheerst op het gebied van web-applicaties. Comet is een nieuwe techniek, die interessant is om te begrijpen en te beheersen, deze techniek berust echter wel op langer bestaande technieken, maar is onlangs pas bekender geworden.
Verder in dit artikel zullen we Comet als naam gebruiken voor de verzameling van deze push-technieken, welke soms andere namen genieten, omdat verschillende mensen verschillende ideeën hierover hebben of omdat het gebruik soms iets afwijkt van dat van het “originele” Comet. Echter zijn de principes erachter gelijk.

inleiding
Comet, ook bekend als server-push of Ajax-push[1], brengt een web-applicatie nog dichterbij een real-time-applicatie als dat Ajax dit bijvoorbeeld al deed. We willen aantonen dat het wel, of niet, interessant is om deze techniek te gebruiken in TAG eForms.
De eerste vraag die naar voren komt is, of Comet überhaupt te gebruiken is in een applicatie als TAG eForms. Als Comet te gebruiken is, wat zijn dan de bijdragen op de korte en lange termijn?

techniek
Comet is techniek om een push effect toe te voegen aan een web-pagina. Dit voegt het “real-time”-principe toe aan een web-site.
Van oudsher berusten web-servers op een protocol dat verplicht stelt dat er een request noodzakelijk is voordat er iets gedaan wordt. Bij real-time informatie is dit een hindernis die lastig te nemen is. Informatie die bij de cliënt aanwezig is, loopt dan altijd achter, want deze wordt niet direct bij beschikbaarheid verzonden naar de gebruiker.
Comet lost dit op via 1 van de 2 volgende methoden: “streaming” of “Ajax with long-polling”. De eerste vereist een openstaande verbinding en over deze verbinding wordt de data verzonden, kortom, een request die dus nooit afgehandeld wordt. De tweede is een Ajax aanroep naar de server, welke pas beantwoord wordt zodra het kan, hierna volgt een volgende aanroep. In tegenstelling tot de eerste, sluit deze verbinding zo af en toe, de streaming methode behoudt de verbinding.
Hieronder vindt u in figuur 1 een schematische weergaven van Comet, beide typen worden in grote lijnen door dit schema beschreven. U ziet een soort brug tussen de server en de client waarover de server de data stuurt op het moment dat deze beschikbaar is. De informatie rondom deze pseudo-

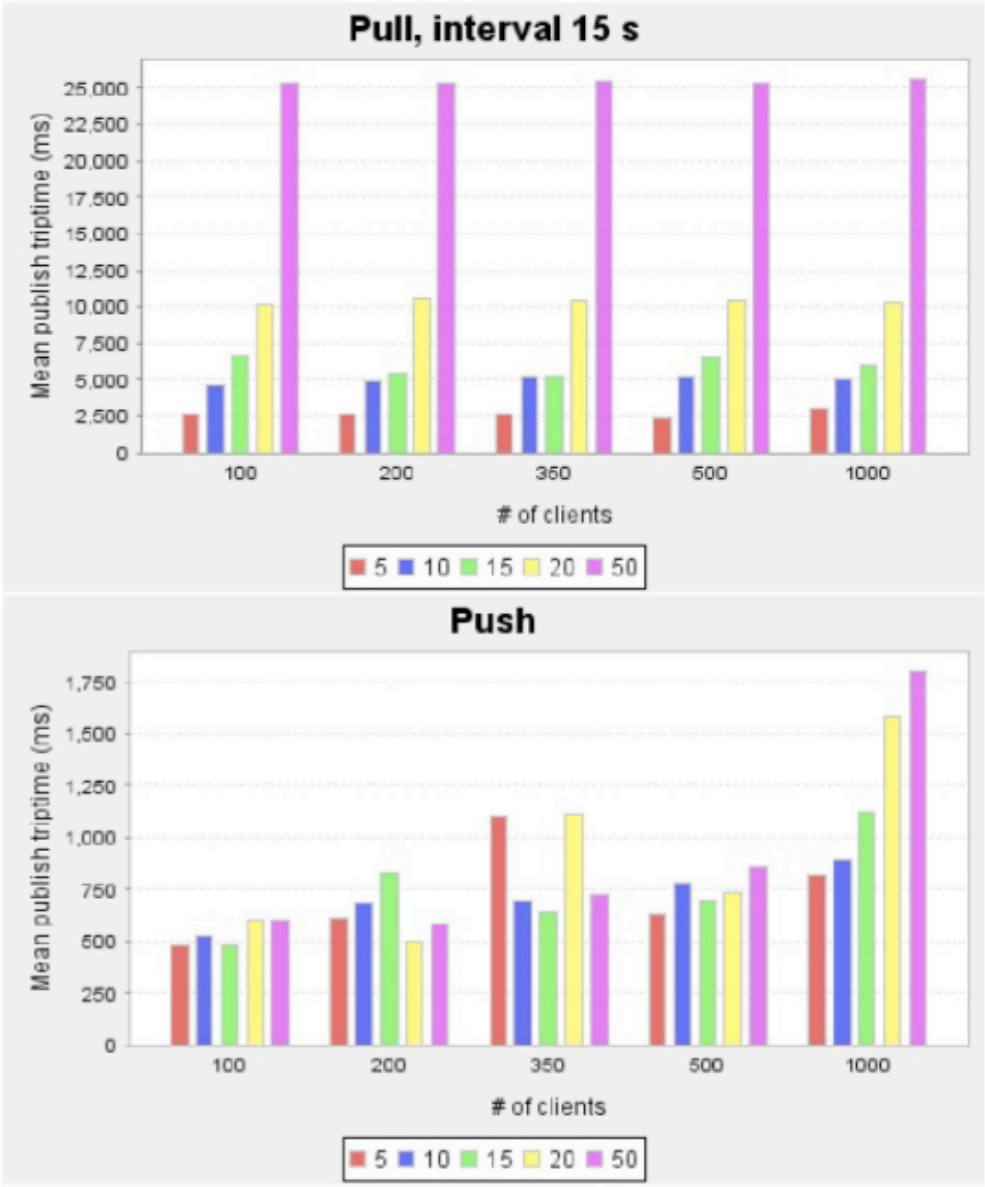
constante verbinding is voor de programmeur zelf in te vullen, deze afbeelding dient voornamelijk om een gevoel te krijgen van de algemene situatie van een Comet implementatie.



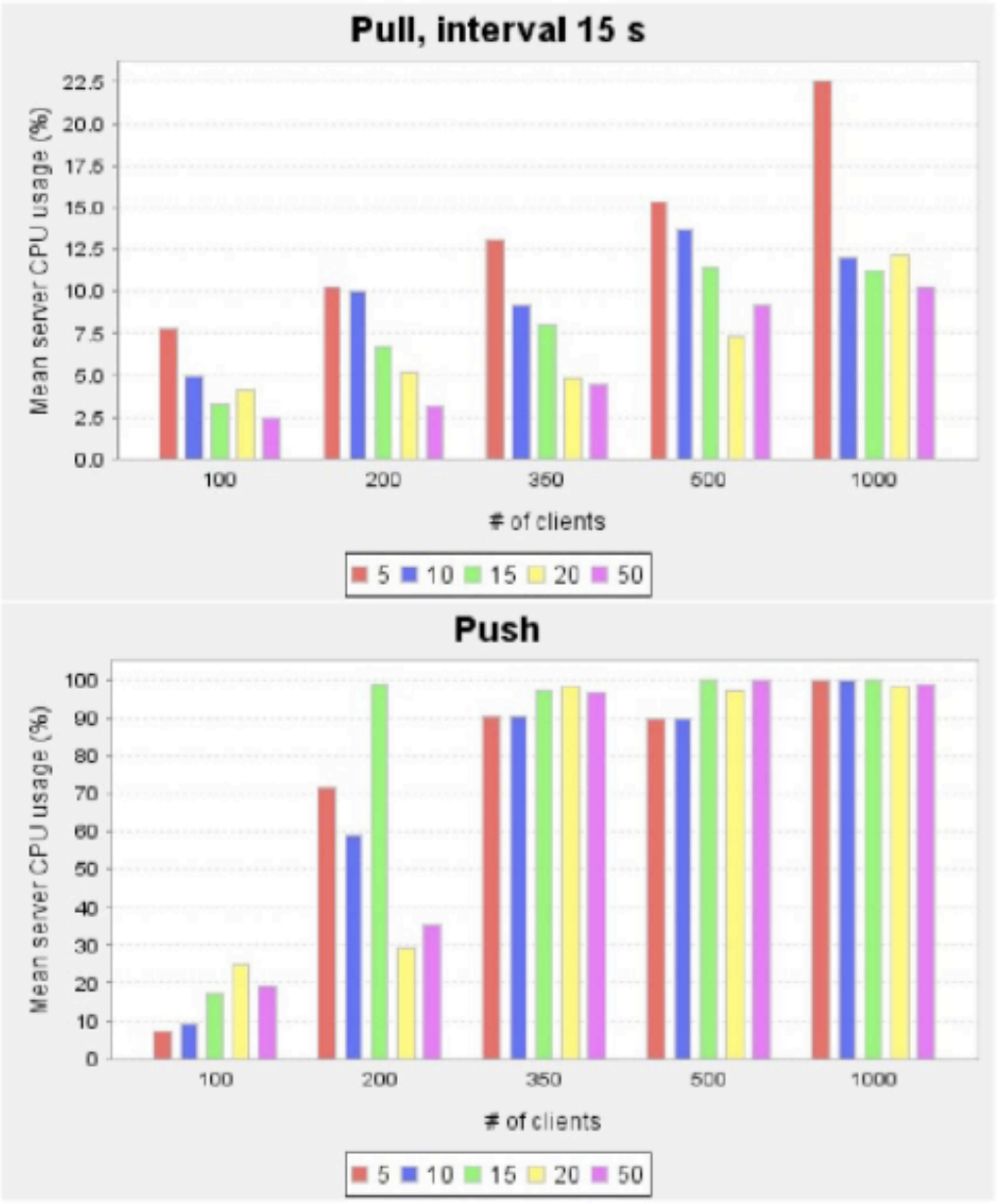
Figuur 1. "Ajax push" volgens ICEFaces [1]

Een van de belangrijkste nadelen die ontstaan is het maximum van 2 verbindingen met een web-server. Maar per applicatie is het verschillend in hoeverre dit een probleem wordt. Ook is dit via verschillende wegen op te lossen. [2]
Een zeer interessant onderzoek naar de verschillen in resultaat en resource gebruik van push en pull technieken in web-applicaties is is een artikel van E. Bozdag et al.[3] Hierin worden bepaalde interessante punten vergeleken als een vergelijkbaar "real-time" web-applicatie dan wel push of dan wel pull technieken gebruikt.
Belangrijkste uitkomsten hier zijn dat hoe vaker de data verandert en hoe minder deze frequentie te voorspellen is, hoe vaker push beter is. We zien dat in dat geval de tijd die het kost om de data bij de gebruiker te krijgen vele malen korter is. Verder zien we dat CPU gebruik interessant is om te bekijken. Doordat de push technieken de connectie openhouden, houden ze ook de threat draaiende die hun request afhandelt. Zie figuur 2 en 3 voor de resultaten, de verschillende kleuren stellen de intervallen in seconden voor van de data verandering op de server.
De noodzaak dat de gebruiker het real-time effect ervaart is een belangrijke reden om te kiezen voor push. Als het zojuist genoemde niet het geval is, zijn

de netwerk en server performance zaken een rede om pull te gebruiken, gezien het feit dat push meer van het netwerk en server vereist.



Figuur 2 gemiddelde verversingstijd [3]



Figuur 3 CPU gebruik [3]

eForms

De toepassing van deze techniek in TAG eForms, zal voor de gebruiker voornamelijk in de inbox terug te vinden kunnen zijn. Dit is in de huidige visie het enige dat real-time gedrag zal moeten hebben. [4]
Als we nu de punten genoemd bij “techniek” naast TAG eForms leggen, zien we dat de frequentie van data verandering op de server, niet te voorspellen is. De informatie bij de gebruiker mag ook enigszins achterlopen, of in afwijkende volgorde binnen komen. Deze laatste veronderstellingen halen we uit het feit dat de applicatie zich als een mail-client moet gedragen en dus een ververs interval van 5 minuten niet ongebruikelijk of hinderlijk is.

Alternatieven

Repeated request
Een mogelijk alternatief voor de eerder genoemde streaming of long-polling, is een timed HTTP-request. Met Javascript kan eenvoudig een module geschreven worden, die op een vast tijdsinterval een Ajax-functie aanroept en zo de informatie in de browser regelmatig ververst. Deze techniek is al vaker gebruikt binnen TAG software, namelijk de TAG dashboard applicatie. Deze applicatie biedt de mogelijkheid de gebruiker niet alleen widgets te plaatsten, maar ook zelf te ontwikkelen. Deze widgets hebben een interval parameter, welke TAG dashboard vertelt om de hoeveel tijd de widget ververst moet worden. [5] Het voornaamste verschil is dat hierbij de request direct afgehandeld worden, bij long-polling, wat hier enigszins oplijkt, wordt de request pas afgehandeld zodra de server bepaalde wijzigingen ondervindt.

Java-applets
Een mogelijk toepassingen waarin Comet gebruikt kan worden is het maken van de formulieren en de workflows. Deze toepassing is echter niet afhankelijk van Comet, maar het kan een communicatie methode zijn tussen de browser en de server. Echter een goed alternatief hiervoor zijn Java-applets. Deze houden namelijk zelf de communicatie met de informatie in eigenhanden en kunnen dus precies naar wens afgesteld worden.
Echter zal binnen TAG eForms eerder een goede javascript library gebruikt worden om dit te bereiken dan Java-applets. Het voordeel hiervan is dat de forms dan al bekend zijn als HTML code.

Toekomst

Onderdelen en gebruikte technieken van TAG eForms dienen gekozen te worden op hun brede inzetbaarheid en toekomstgerichtheid. Niet enkel op toepasbaarheid in het huidige ontwerp.

Comet is zeer interessant om te beheersen voor een gepassioneerde web-applicatie ontwikkelaar. Echter is het op dit moment niet onmisbaar voor toekomst plannen voor TAG eForms.

Conclusie

Ten eerste komen we tot de conclusie dat deze techniek te gebruiken is in de TAG eForms. Zoals we eerder zagen brengt het een real-time inbox update functie. Het genoemde voordeel hier is dat de gebruiker direct de meest actuele informatie voor zich heeft. Verder is het een interessante oplossing voor real-time statistieken, echter hebben we niet de toegevoegde waarde kunnen vast stellen van een real-time monitor aan TAG eForms op dit moment. Wel is het interessant om de rapportage-tool te gebruiken, maar deze heeft geen noodzaak aan Comet, het gaat namelijk een moment opnamen. Ook is het mogelijk later op bepaalde modules Comet toe te passen zonder dat dat in het huidige ontwerp meegenomen moet worden.
Het is dus evident dat de enige interessante locaties voor Comet (zie de paragraaf ‘eForms’), niet afhankelijk zijn van een techniek als Comet. Wat ons dus de keuze laat berusten op andere feiten.
We zagen eerder dat het CPU gebruik bijna exponentieel stijgt t.o.v. een timed pull-interval. Dit is geen sterkt tegen argument de grote van TAG eForms zal niet dermate zijn, ook de hoeveelheid gebruiker zal zelden boven de 200 zitten.
Wat overblijft is de toegevoegde waarde tegenover de extra tijd die nodig is TAG eForms met Comet te implementeren.
Comet zal dus niet toegevoegd worden aan TAG eForms, rekening houdende met de huidige opdracht omschrijving en met, logische, toekomstige uitbreidingen.

Referenties

[1] ICEFaces, http://www.icefaces.org/main/product/ajaxpush.iface
[2] http://en.wikipedia.org/wiki/Comet_(programming) (niet als feiten behandeld, wel interessant om een idee te krijgen van Comet)
[3] A comparison of push and pull techniques for Ajax, E. Bozdag et al, 2007 (TU-Delft)
[4] TAG eForms opdracht omschrijving, TAG software & consultancy, Dordrecht 2008
[5] TAG dashboard beschrijving, TAG software & consultancy, Dordrecht 2007

Onderzoek naar de toepasbaarheid van RUP en MoSCoW voor kleine software development projecten

Vincent Simon van Beek
TU-Delft
vincent@vlogic.nl

Abstract

Het ontwikkelen van software is een complex proces dat een structuur nodig heeft om te slagen. Dit artikel zal proberen aan te tonen dat niet RUP maar XP een geschikte methode is voor het uitvoeren van een klein software development project. Ook zal dit artikel laten zien hoe binnen deze methoden gebruik gemaakt kan worden van ranking methode. Dit artikel is gestoeld op literatuur onderzoek en is voornamelijk gebaseerd op andere artikelen.

1 Inleiding

Met dit artikel zal ik antwoord geven op de vraag: “zijn RUP en MoSCoW verenigbaar en toepasbaar op kleine software development projecten?”. Naast deze hoofdvraag zal er gekeken worden naar alternatieven voor RUP en MoSCoW. Met een klein project wordt bedoeld een project met 1 tot 5 medewerkers [appendix 1][5].

2 Project planning en organisatie

RUP en XP zijn beiden methoden die gebruikt kunnen worden om een software development project te organiseren. RUP staat voor Rational Unified Process, dit is een commerciële variant van UP (Unified Process). XP staat voor Extreme Programming.

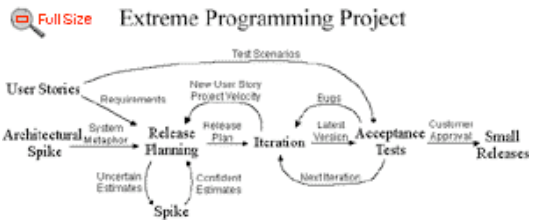
2.1 RUP [3]

RUP is een framework dat beschrijft hoe een software development project georganiseerd kan worden. Het beschrijft welke fasen [7] doorlopen dienen te worden en welke documenten geproduceerd dienen te worden. RUP is volledig use-case gestuurd. Van uit de use-cases worden requirements geformuleerd, deze worden vervolgens omgezet in formele ontwerp documenten. RUP schrijft voor dat vrijwel alle requirements duidelijk zijn voordat er begonnen wordt met het implementeren van de software. Dit betekend dat er veel tijd besteed moet worden aan de eerste en tweede fase ([7] zie appendix 3). Tijdens fase 3 “de constructie fase” wordt het grootste gedeelte van de implementatie gedaan. Hierna wordt er nog getest en wordt in fase 4 het eindproduct opgeleverd. De klant wordt na het verzamelen van de use-cases verlaten en wordt pas weer geïnformeerd op het moment van oplevering. Voor kleine projecten is het uitvoeren van RUP in zijn volledigheid veel te complex [4]. Er wordt dan ook vaak gekozen om een subset van RUP te gebruiken, uit de literatuur blijkt [4] echter dat het aanpassen van RUP aan een klein project veel te complex is, quote: ”Based on our own, and a few other empirical studies on tailoring and introduction of RUP into development organizations we found that there exist few or none (reported) direct success stories. All experiences pull in the same direction; RUP is, out of the box, too complex, however, tailoring it to specific needs is also too complex. Looking at the evolution of RUP itself over the past years and the cases we summarize here we see a clear need for, and movement towards, a more agile process that can bee tailored with less effort.” [4].

2.2 XP

Extreme Programming is een methode om een software development project te organiseren. XP is in de kern gebaseerd op een grote klant betrokkenheid bij het gehele project. Het project start met het opstellen van gebruikers-beschrijvingen, deze beschrijvingen bevatten uitleg van gebruikers over wat het systeem moet kunnen. Aan de hand van deze beschrijvingen worden requirements opgesteld en worden acceptance tests geschreven. Deze tests moeten zo worden opgesteld dat deze minimaal de functionaliteit uit de gebrbuikers-beschrijvingen omvatten. De tests moeten geautomatiseerd worden met bijvoorbeeld unit-testing. Vervolgens wordt een planning gemaakt en wordt er begonnen met het implementeren van de software. De implementatie vind plaats in kleine modules die afgerond worden en naar de klant worden gestuurd om deze te beoordelen en te testen. Naast de acceptance tests worden ook nog unit tests gebruikt om lossen functies te testen. De software mag aan het einde pas opgeleverd worden als alle acceptance en unit tests slagen. Omdat er geen ontwerp wordt gemaakt is het van belang dat programmeurs goed samen werken om de kwaliteit te waarborgen. Dit kan bevorderd worden door gebruikt te maken van pair programming, dit houdt in dat twee programmeurs samen aan 1 werkplek zitten. 1 van de programmeurs hanteert het toetsenbord en is daarmee de baas over de op dat moment te schrijven functie. De ander denkt na over het grotere geheel en geeft aanwijzingen over welke richting de code op moet. Om efficiënt te werken moeten de programmeurs vaak van taak wisselen, minimaal een aantal keer per uur [9]. Naast pair programming is het begrip “Collective code ownership” van groot belang voor de kwaliteit van de software. Collective code ownership houdt in dat iedere programmeur iedere regel code van het project mag wijzigen. Dit principe werkt omdat voor alle code eerst een unit test geschreven wordt. Met behulp van deze tests kan dan worden vastgesteld of de aangepaste code nog voldoet aan de eisen.

XP is een methode die vooral bedoeld is voor kleine projecten met vaak wijzigende requirements. XP is hier uitermate geschikt voor omdat de iteraties in de implementatie fase kort zijn en de klant betrokkenheid groot is.



bron: http://www.extremeprogramming.org/

3 Ranking

Er zullen vier verschillende ranking methoden beschreven worden en er zal worden aangeven voor welke project fase deze het meest geschikt zijn. De volgende vier methoden zullen behandeld worden: Simple Ranking, MoSCoW, AHP en Hundred Dollar Method [2].

3.1 Simple Ranking

Simple Ranking is ogenschijnlijk de eenvoudigste manier om requirements te ranken. Van de requirements wordt een lijst van 1..n gemaakt, hierbij geldt dat requirement 1 de belangrijkste is en n de minst belangrijke. Uit onderzoek blijkt dat deze methode uitsluitend succesvol is bij een gering aantal requirements. Psychologisch onderzoek heeft aangegeven dat 7 (plus of min 2) requirements het maximum is wat een mens in een keer kan overzien [2].

3.2 MoSCoW

MoSCoW is een acroniem dat gebruikt wordt om een hiërarchische verdeling te maken van de requirements.
Must have: deze requirements zijn verplicht om het project te laten slagen.
Should have: features die waar mogelijk zeer gewenst zijn.
Could have: features die waar mogelijk gewild zijn maar minder belangrijk dan should have.
Won't have: dit zijn requirements die wel belangrijk kunnen zijn maar die bij de huidige release niet zullen worden ontwikkeld. Deze requirements zijn tijdens de ontwerp fase wel van belang omdat er in het ontwerp wellicht rekening moet worden gehouden met toekomstige releases.

3.3 AHP

AHP, Analytic Hierarchy Process is een bekend algoritme dat gebruikt wordt bij het nemen van beslissingen. AHP is erg sterk in het vergelijken van alternatieven van verschillende ontwerpen en het vergelijken van verschillende componenten in een systeem. AHP gebruikt pair-wise comparison om tot een oordeel te komen. Om dit te kunnen doen met requirements moet iedere requirement met alle andere requirements vergeleken worden en moet de onderlinge verhouding worden aangegeven met een getal van 1 t/m 9. Met deze waarden wordt een matrix met relatieve afhankelijkheden gevuld. Met deze data kan AHP een een prioriteiten lijst maken. AHP wordt beschreven als tijdrovend en alleen toepasbaar op kleine datasets.

3.4 Hundred Dollar Method

De honderd dollar methode is een simpele methode om een persoon of een groep van belanghebbenden te betrekken bij het bepalen van prioriteiten. De methode werkt in drie simpele stappen:

De belanghebbenden wordt gevraagd zich in te beelden dat ze 100 dollar hebben.

De honderd dollar wordt verdeeld over der requirements.

De verdeling van de 100 dollar wordt geëvalueerd. Als de 100 dollar te gelijkmatig verdeeld is wordt er nog eens 100 dollar verdeeld. Deze tweede 100 dollar moet echter in een deel van 50 dollar en twee delen van 25 dollar worden uitgegeven.

Het verdelen van geld is een hele effectieve manier om een gedetailleerd beeld te krijgen van de prioriteiten van de wensen.

3.5 Toepassing

Het gebruik van ranking moet per project fase apart bekeken worden, het is namelijk zo dat iedere methode meer of minder kennis van de requirements vereist. Een project kan grofweg in drie fasen verdeeld worden, start fase, midden fase en latere fase. Bij de start van een project is er nog weinig kennis over de requirements en het te ontwikkelen product. In deze fase is het dus niet zinvol heel erg gedetailleerd te ranken, daar de requirements nog kunnen wijzigen als de kennis over het product groeit. In deze fase is het MoSCoW model dus erg goed

toepasbaar. Het is simpel toe te passen en bij veranderende requirements is het niet nodig het volledige ranking proces opnieuw te doen.

Tijdens de midden fase komen er eigenlijk geen requirements meer bij het kan hooguit gebeuren dat requirements verder uitgewerkt worden. In deze fase kan het van belang zijn meer onderscheid in de ranking te maken. Dit kan bijvoorbeeld door de groepen requirements uit de MoSCoW ranking verder onder te verdelen met behulp van de Hundred Dollar Method of als het project heel erg eenvoudig en duidelijk is met de Simple Rank methode.

De latere fase is de fase van het project wanneer het grootste deel van het programmeer werk plaats vind. Tijdens deze fase kan het van belang zijn, om een verdergaande ranking te maken van de requirements. Dit zou dan kunnen met behulp van AHP. Deze methode kost wel veel tijd maar het verkeerd besteden van waardevolle resources kan veel erger en uit pakken dan de tijd die besteed wordt aan ranking.

5 Conclusie

De hoofdvraag “ zijn RUP en MoSCoW verenigbaar en toepasbaar op kleine projecten?” moet met nee worden beantwoord. RUP is volgens de literatuur te complex om toe te passen op kleine project. Kleine projecten vragen om een meer flexibele methode. Extreme programming is een flexibele methode die uitermate geschikt is voor kleine projecten. MoSCoW is een ranking methode die vooral geschikt is aan het begin van een project om de algemene prioriteiten van requirements te bepalen. Aangezien XP in de eerste fase een requirements document samenstelt is het goed denkbaar dat MoSCoW wordt gebruikt om deze requirements te ordenen. Tijdens de implementatie fase kunnen de belangrijkste requirements eerst worden geïmplementeerd zodat de klant direct het resultaat hiervan kan zien. Bij het plannen van een iteratie kan de hundred dollar method goed worden toegepast, deze methode is simpel snel en effectief.

Referenties

[1] Christiane Gresse von Wangenheim, Se´ rgio Weber, Jean Carlo Rossa Hauck, Gisele Trentin, Experiences on establishing software processes in small companies, Information and Software Technology 48 (2006) 890–900

[2]Sarah Hatton, Choosing the “Right” Prioritisation Method, 19th Australian Conference on Software Engineering maart 2008

[3] John Smith, A Comparison of RUP® and XP, Rational Software White Paper TP167, may 2001

[4] Geir Kjetil Hanssen, Finn Olav Bjørnson, and Hans Westerheim, Tailoring and Introduction of the Rational Unified Process

[5] © Centraal Bureau voor de Statistiek, Den Haag/Heerlen, Overzicht aantal bedrijven naar aantal medewerkers , 13-10-2008

[6] http://www.extremeprogramming.org/ Copyright (c) 1999, 2000, 2001 Don Wells. All Rights reserved.

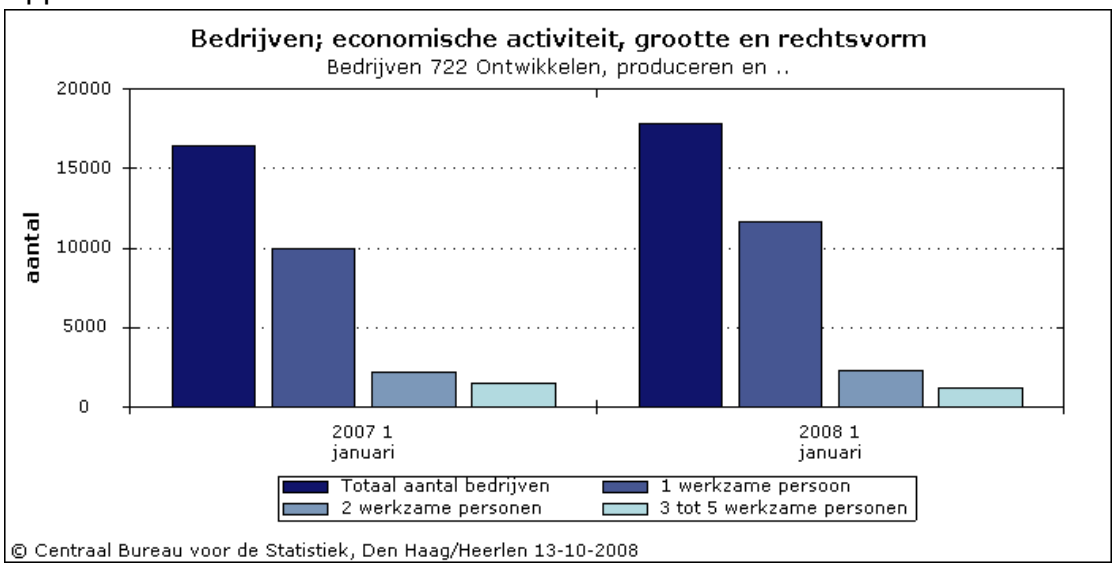
[7]http://www.ibm.com/developerworks/webservices/library/ws-soa-term2/rup.jpg

[8] Rational Unified Process Best Practices for Software Development Teams, Rational Software White Paper TP026B, Rev 11/01

[9] *Strengthening the Case for Pair Programming,* Laurie Williams, Robert R. Kessler, Ward
Cunningham, Ron Jeffries, IEEE Software, Vol. 17, No. 4, 2000, pp. 19-25

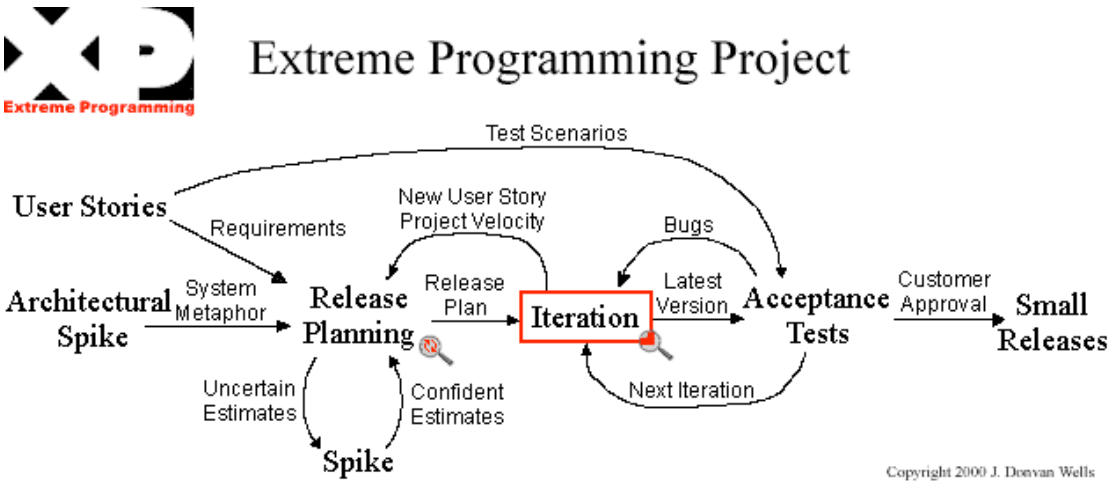
Appendix

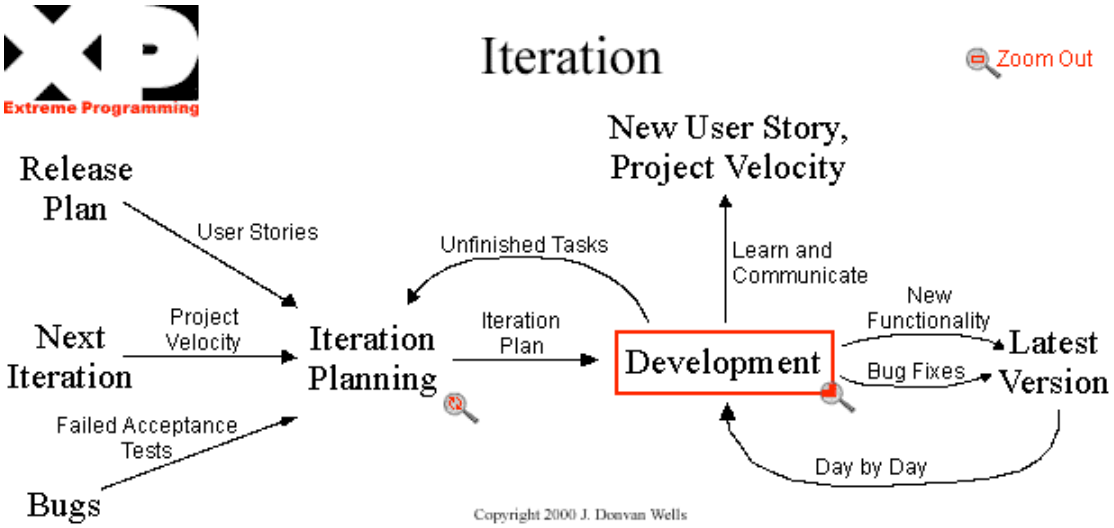
Appendix 1

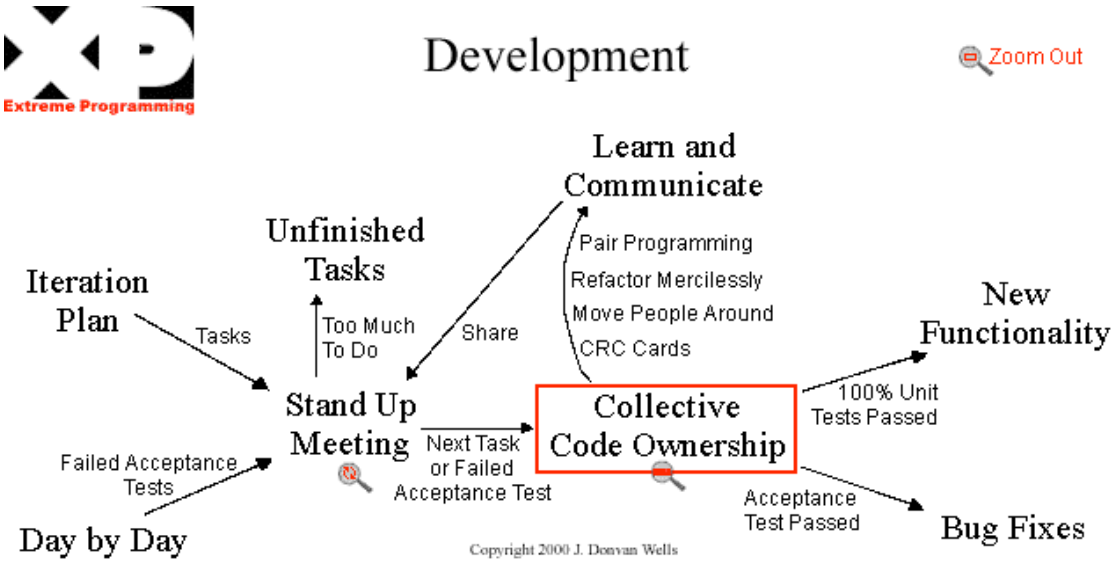


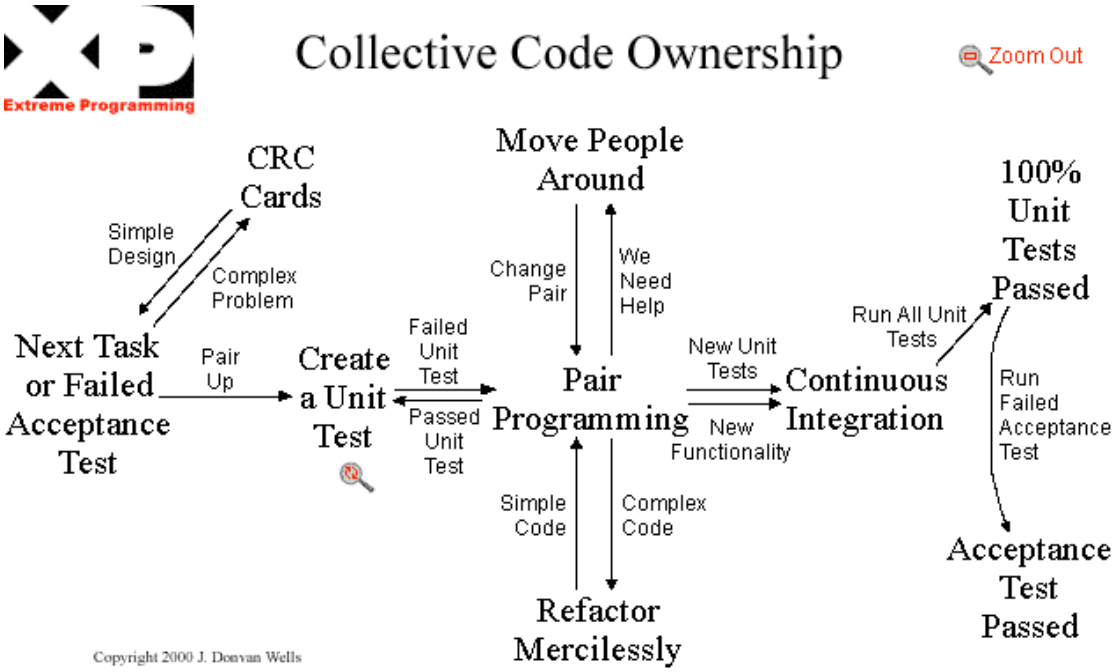
722 Ontwikkelen, produceren en ..
722 Ontwikkelen, produceren en uitgeven van software; software consultancy
Deze groep omvat:
- analyseren, ontwerpen en programmeren van gebruiksklare systemen:
* analyseren van de behoeften en problemen van de gebruiker en verlenen van advies over de beste oplossing;
* ontwikkeling, productie, levering en samenstelling van documentatie over standaardprogrammatuur en over speciale programmatuur op bestelling;
  * schrijven van programma's volgens richtlijnen van de gebruiker;
- interim systeembeheer;
- ontwerpen van homepages voor internet, webdesign;
- ontwerpen, programmeren van websites.
Eventueel in combinatie met:
- detachering van systeemontwerpers en -analisten.
Deze groep omvat niet:
- reproductie van standaard programmatuur (2233);
- uitgeverijen van cd-roms, diskettes en dergelijke met software, spellen, boeken, encyclopedieën en dergelijke (2215).

Appendix 2 [6]

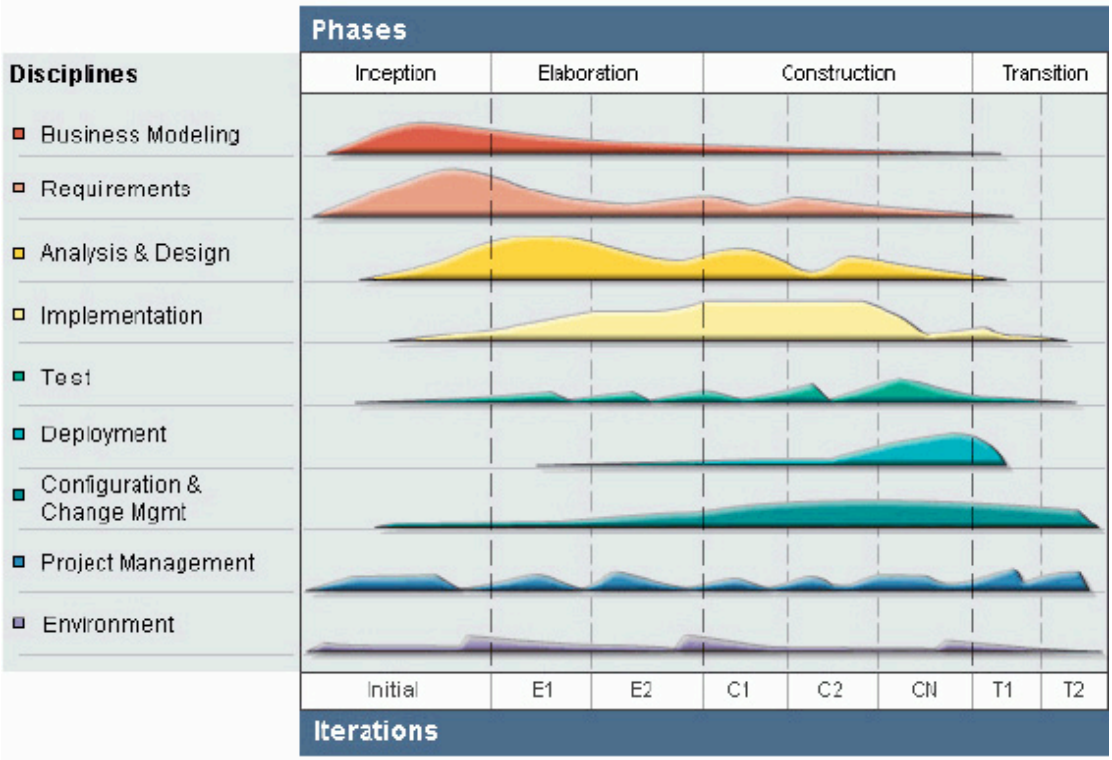

Extreme Programming
Extreme Programming Project
Test Scenarios
User Stories
Requirements
New User Story
Project Velocity
Bugs
Architectural Spike
System Metaphor
Release Planning
Release Plan
Iteration
Latest Version
Acceptance Tests
Customer Approval
Small Releases
Uncertain Estimates
Confident Estimates
Spike
Next Iteration
Copyright 2000 J. Donvan Wells


Extreme Programming
Iteration
Zoom Out
Release Plan
New User Story, Project Velocity
User Stories
Unfinished Tasks
Learn and Communicate
New Functionality
Next Iteration
Project Velocity
Iteration Planning
Iteration Plan
Development
Bug Fixes
Latest Version
Failed Acceptance Tests
Bugs
Day by Day
Copyright 2000 J. Donvan Wells


Extreme Programming
Development
Zoom Out
Learn and Communicate
Unfinished Tasks
Pair Programming
Refactor Mercilessly
Move People Around
CRC Cards
Iteration Plan
Tasks
Too Much To Do
Share
New Functionality
Stand Up Meeting
Next Task or Failed Acceptance Test
Collective Code Ownership
100% Unit Tests Passed
Failed Acceptance Tests
Acceptance Test Passed
Day by Day
Bug Fixes
Copyright 2000 J. Donvan Wells


Extreme Programming
Collective Code Ownership
Zoom Out
CRC Cards
Simple Design
Complex Problem
Next Task or Failed Acceptance Test
Pair Up
Create a Unit Test
Failed Unit Test
Passed Unit Test
Move People Around
Change Pair
We Need Help
Pair Programming
Simple Code
Complex Code
Refactor Mercilessly
New Unit Tests
New Functionality
Continuous Integration
Run All Unit Tests
100% Unit Tests Passed
Run Failed Acceptance Test
Acceptance Test Passed
Copyright 2000 J. Donvan Wells

Appendix 3 [7]


Phases
Disciplines
Inception
Elaboration
Construction
Transition
Business Modeling
Requirements
Analysis & Design
Implementation
Test
Deployment
Configuration & Change Mgmt
Project Management
Environment
Initial
E1
E2
C1
C2
CN
T1
T2
Iterations

H. Artikel Adeptive user interface in web application

Adaptive user interface in web-applicaties

Vincent S. van Beek
Technische Universiteit Delft
Mekelweg 4, Delft
V.S.vanBeek@student.tudelft.nl

Mark Zijdemans
Technische Universiteit Delft
Mekelweg 4, Delft
M.Zijdemans@student.tudelft.nl

abstract
Een gebruikers interface dient een gebruiker te helpen bij het gebruik van een programma. Echter moet het zo zijn dat het gebruik dusdanig veel eenvoudiger wordt dat het programma gebruikt kán worden. Nu is niet alleen de eenvoud belangrijk, ook moet de juiste informatie in beeld komen voor de gebruiker.
Wij laten 2 manieren zien waarop een gebruikersinterface dynamisch ontworpen kan worden, door te kijken naar het apparaat waarmee de gebruiker werkt en door te kijken naar het gedrag van de gebruiker.
Ook laten we zien dat het bestuderen van de gebruikers, je veel vertelt over hoe je de 2 manieren dient toe te passen. Het is namelijk niet altijd een noodzakelijke bijdrage om een interactieve helper agent toe te voegen, soms is een voorbeeld al voldoende. Een helper agent kan in zo'n situatie misschien alleen irritatie opwekken, zoals later zal blijken.

inleiding

Dit artikel zal de vraag beantwoorden of de gebruikers interface van TAG eForms afhankelijk dient te zijn van het gebruikte apparaat en/of de gebruiker die deze bedient.

device dependent interface

inleiding
Bij het ontwerpen en implementeren van een gebruikersinterface moet er met verschillende zaken rekening worden gehouden. Een daar van is het device waarmee de gebruikersinterface wordt weergegeven. TAG eForms wordt een web-applicatie, deze moet dus met behulp van een browser worden weergegeven. Deze browser kan op verschillende platformen werken, denk bijvoorbeeld aan desktop computers, laptops, pda's, en smartphones. Het verschil in capaciteit van het platform bepaalt voor een deel ook het verschil in capaciteit van de browser. Zo kan een smartphone vaak geen Flash afslepen terwijl een browser op een laptop dat wel kan. Naast dit verschil is er nog een belangrijk verschil tussen de platformen, de monitoren van de platformen kunnen erg verschillen van elkaar. Zo heeft een smartphone vaak een zeer kleine monitor en een desktop computer juist een relatief grote monitor. Ook de netwerk capaciteit moet worden meegenomen bij het ontwerp van de gebruikers interface.
Naast de eigenschappen van de browser zijn er ook nog andere trends op het gebied van de ontwikkeling van web-applicaties die het ontwikkelen van gebruikers interfaces beïnvloeden.
Deze paragraaf zal een aantal van deze aspecten belichten, aan het einde van de paragraaf zal de toepassing er van op het eForms project worden beschreven.

Trends op het gebied van web-applicatie development

Web-applicatie development is een zeer ruim begrip, bij het beschrijven van een aantal trends zal het domein beperkt worden tot trends die invloed hebben op het eForms project. De volgende domeinen zullen beschreven worden: browser trends, netwerk trends en fat client vs device independence.

Browser trends
De ontwikkeling van de browser van een simpele tekstfile interpreter tot een geavanceerde render engine en applicatie platform heeft er voor gezorgd dat web-applicaties nu meer zijn dan een platte weergave van informatie. Deze ontwikkeling van de browser lijkt door te zetten. De door Apple geïntroduceerde *canvas* tag [1] bijvoorbeeld maakt het mogelijk om 2d beelden in de browser te renderen. Voorheen was het alleen mogelijk om beelden op de server te renderen, wat bij een groot aantal gebruikers tot performance problemen kon leiden. In de Firefox roadmap [2] en de WHATWG [3] specificatie wordt er al gehint dat er een volledige 3d hardware geaccelereerde render omgeving in de browser komt. Deze ontwikkeling zal er toe leiden dat een groter deel van de web-applicatie naar de client gaat schuiven. Deze ontwikkeling wordt mede mogelijk gemaakt doordat desktop en laptop platformen tegenwoordig voldoende capaciteiten hebben om 3D graphics te renderen.

Netwerk trends
Door de desktop-to-web trend [4] zal er een steeds grotere vraag komen naar continue verbinding met het internet. De desktop-to-web trend is de trend dat er steeds meer desktop applicaties worden her ontwikkeld als web-applicaties, denk aan Google documents en Outlook web-access. WHATWG beschrijft een set van events die een applicatie developer kan gebruiken om de netwerk status te checken. De applicatie zou werk van een gebruiker in het geval van het verlies van de verbinding lokaal kunnen opslaan en op het moment dat er weer verbinding is kunnen versturen. Een andere ontwikkeling is het integreren van P2P in de browser. Zowel Opera [5] als Firefox [6] werken op dit moment aan deze functionaliteit. De integratie van P2P in de browser kan er toe leiden dat er minder opslag ruimte nodig is op de server omdat grote files uitsluitend nog via P2P zullen worden verzonden. De stabiliteit en snelheid van de internet verbinding zullen wel cruciaal zijn om de gebruiker een goede beleving van de applicatie te kunnen geven.

Fat client vs device independence [4]
Het fat client model heeft de laatste jaren veel aanhang gekregen door de ontwikkeling van de pc en de ontwikkeling van de browser. De trend dat er steeds meer gebruik wordt gemaakt van mobile devices zoals pda's en smartphones vraagt om een aanpak van device onafhankelijke ontwikkeling. Sterker nog er zijn mobile devices met extra features die een pc niet heeft en die voor de applicatie kunnen worden aangewend. Zo zijn er op dit moment mobile devices met een touchscreen en bewegingssensoren. Deze extra features kunnen ten goede worden aangewend bij de ontwikkeling van een applicatie.

Automatisch gegenereerde gebruikers interface
Een manier om device onafhankelijke web-applicaties te maken is door de interface automatische aan de hand van een model te laten genereren [12]. De eigenschappen van de device kunnen dan meegenomen worden bij het genereren van de gebruikers interface. Het is dan dus ook mogelijk om in het model te specificeren welke functionaliteit op welk platform gebruikt kan worden. Zo is het denkbaar dat een applicatie een functie bevat waarbij de muis een vereiste is (PC). Deze zelfde applicatie zou een functie kunnen hebben waarbij plaatsbepaling vereist is (smartphone met GPS).
Het volledig genereren van een gebruikers interface vanuit een model is iets dat tot nu toe niet wordt toegepast op grote schaal. Dit komt doordat er betere resultaten worden behaald door menselijke designers. Wel worden er goede resultaten behaald met het ontwikkelen van verschillende interfaces voor verschillende devices. Om dit succesvol te kunnen doen is het van belang om de applicatie logica en de interface van elkaar te scheiden. Dit maakt het dan mogelijk om de zelfde logica op verschillende manieren aan te wenden.
Bij de ontwikkeling van web-applicaties zie je dat er om een scheiding van interface en logica te krijgen steeds vaker MVC modellen [7] worden toegepast. Verder worden template engines steeds populairder.

TAG eForms

De eForms applicatie is grofweg op te splitsen in twee delen, namelijk het beheer gedeelte waar formulieren gemaakt worden en het gebruikers deel waar de formulieren gebruikt worden. Het ontwerpen van een formulier is een complexe taak waarbij er veel gebruik gemaakt wordt van de UI. Eigenijk zou dit deel van de applicatie gezien kunnen worden als een desktop-to-web applicatie. Een ontwikkeling die op dit deel van de applicatie invloed zou kunnen hebben is de mogelijke implementatie van een module om de netwerk status te kunnen lezen. Een andere ontwikkeling die van invloed zou kunnen zijn in een toekomstige release van de applicatie is de mogelijke ontwikkeling van een 3D render engine voor de browser.

Het gebruikers deel van de applicatie, het deel waar de formulieren gebruikt worden zal veel te maken krijgen met de trend van device onafhankelijkheid. In de toekomst zal dit deel van de applicatie namelijk toegankelijk moeten worden voor handheld devices (smartphones, PDA's, blackberrys, etc). Aangezien de formulieren gegenereerd worden aan de hand van een model uit de database zal het mogelijk zijn om pc te voorzien van een versie met JS validatie en zal er voor een handheld device een versie gegenereerd kunnen worden met server-side validatie.

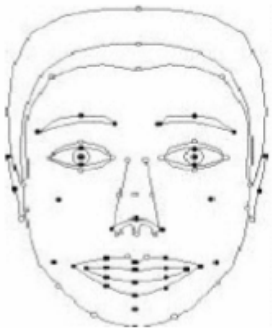
user dependent interface

inleiding
Een gebruikersinterface hoeft niet alleen afhankelijk te zijn van het apparaat dat gebruikt wordt om de web applicatie te bekijken. Ook het gedrag van de gebruiker is een interessante factor om mee te nemen. Het vakgebied Human-Computer interaction (HCI) houdt zich bezig met de studie van de interactie tussen mens en machine. In deze paragraaf zal worden uitgelegd welke subdomeinen en ontwikkelingen van HCI in een programma als TAG eForms gebruikt kunnen worden. Eerst zullen de laatste ontwikkelingen op het gebied van HCI beschreven worden.
De belangrijkste reden dat er gekeken wordt naar een gebruiker ondersteunende GUI, is het feit dat gebruikershandleidingen zeer weinig gebruikt worden. Het onderzoek van Novick en Ward [11] laat zien dat gebruikers zelfs liever het product niet gebruiken dan de handleiding erop na te slaan. De reden voor dit gedrag ligt vaak in de grootte en de onoverzichtelijkheid van deze handleidingen, maar ook bij de tijd die het de gebruiker kost zich bekend te maken met de geschreven stukken over het onderwerp van zijn probleem.
Het onderzoek van Novick en Ward [11] laat zien dat gebruikers de navigatie in een digitale handleiding al prettiger vinden, maar dat ook deze niet altijd als behulpzaam gezien wordt. Het doel is dus een interface of help-functie te ontwikkelen die de gebruiker op een juiste, maar vooral prettige wijze ondersteund.

Gebruikers en problemen
In het werk van Novick en Ward [11] wordt ook verteld op welke wijze gebruikers dan wel met problemen omgaan die ze tegenkomen met de gebruikte software. Zij stellen vast dat als de gebruiker de software echt gebruikt, deze dus toch echt op onderzoek gaat naar de oplossing, het liefst wordt er naar de oplossing vraagt bij bijvoorbeeld een collega of mede-student. Is er niemand met de juiste kennis in de buurt dan wordt de handleiding geraadpleegd. Zelf zien we dat in de tussentijd een willekeurige zoekmachine op het Internet wordt gebruikt om te kijken of andere mensen dezelfde problemen tegenkomen. Dit is echt alleen mogelijk met niet op maat gemaakte software.

Human-Computer interaction
Een van de belangrijkste doelen van HCI is het meer bruikbaar maken van computer systemen/software. [8] Dus dit betekent dat bijvoorbeeld de software, die geschreven moet worden voor duizenden gebruikers, voor iedere individuele gebruiker geschikt is. [8]
Een zeer natuurlijke wijze om vast te stellen wat de gebruiker nodig heeft, is gebruik te maken van gelaatsuitdrukkingen of zithouding. In het werk van Bahr et al. [9] wordt dit gebruikt om de Microsoft helper agent te verbeteren (Microsoft's Clippit ®).
Deze GUI helper agent is speciaal gekozen, omdat deze na introductie al vrij snel de standaard instelling kreeg van "uitgeschakeld". Hij gaf of onnodige informatie, of kwam tevoorschijn wanneer dit zeer ongewenst was.
Bahr et al. willen dergelijke helper agents verbeteren door te bepalen of de gebruiker actief bezig is of niet en of de gebruiker globaal of lokaal bezig is. Dit laatste houdt in dat bijvoorbeeld het gehele scherm afgezocht wordt naar een optie, of dat slechts 1 window gebruikt wordt voor het bekijken van instellingen.
Het bepalen van de werkhouding gebeurt door sensoren in de stoel, deze bepalen de zithouding van de gebruiker. Zo kan worden geconstateerd of de gebruiker afgeleid is, of juist intensief aan het werk. Ook de webcam helpt hierbij, deze kan ook bepalen of de gebruiker lokaal of globaal aan het werk is. Een extra optie die met behulp van en webcam kan worden bepaalt is de gemoedstoestand van een gebruiker. [9]

Zie figuur 1, waarin een face video map is weergegeven. De afstanden tussen de dots, ten opzichte van de neutrale gelaatsuitdrukking, bepalen of de gebruiker, vrolijk, boos, gefrustreerd, geconcentreerd of bijvoorbeeld kalm is.



Figuur 1. Face Video Map

Buiten de hierboven genoemde wijzen, zijn er meer mogelijkheden om te toestand van de gebruiker te achterhalen. Het principe blijft echter het zelfde, namelijk door de toestand van de gebruiker te analyseren wordt bepaald welke interface of informatie op dat moment moet worden weergegeven.
Ook is de wijze waarop de gebruiker ondersteund wordt in zijn werk niet eenzijdig. Er zijn meerdere manieren om een gebruiker te ondersteunen. Een aparte helper agent is een mooie gadget, maar kan te veel van het goede zijn voor een kleinere applicatie. Deze zijn misschien meer geholpen met kleine pop-ups of voorbeelden. Maar ook de interface zelf kan afhankelijk zijn van de gebruiker, de genoemde helper GUI die verbeterd werd door Bahr et al. is eigenlijk een programma op zich, wat inhoud dat het programma zijn eigen interface aanpast aan de gebruiker, dit is dus ook toe te passen op de hoofdapplicatie zelf.

TAG eForms
TAG eForms is een web-applicatie, welke niet zal beschikken over meer input dan slechts de muis en het toetsenbord. Het is dus niet mogelijk functies te gebruiken als gelaatsuitdrukking en gestures (gebaren).
Wat is er eigenlijk bekend over de gebruiker van TAG eForms? Deze vraag heeft een eenvoudig antwoord, namelijk: zeer weinig. Echter verdient het wel enige overdenking of dit echt waar is. In eerste instantie is het een logisch antwoord, het product is zeer breed inzetbaar en zeker het invullen van de formuleren moet door iedereen gedaan kunnen worden.
Maar we zien al dat we nu een verschil maken tussen het invul gedeelte en het beheerders gedeelte. We kunnen eigenlijk 3 soorten gebruikers definiëren in TAG eForms, namelijk de beheerder, de systeembeheerder en de gebruiker [10]. De gebruiker is slechts gemoeid met het invullen van de juiste formulieren en deze te verzenden. Zijn kennis van dit systeem of zijn affiniteit met computers mag zeer laag zijn, hier moet dus rekening mee gehouden worden tijdens het ontwerp en implementatie van het invul gedeelte. Daartegenover krijgen de beheerders van TAG eForms een training en andere consultancy diensten, hun kennis van het systeem zal hierdoor dus groeien.
Een aparte taak binnen het systeem is er weggelegd voor een systeembeheerder. Buiten het installeren van de software, is er bijvoorbeeld de mogelijkheid datasources in te voeren [10]. Datasources zijn eigenlijk koppelingen naar andere databases of systemen, zodat het mogelijk wordt informatie uit andere systemen in de formulieren te gebruiken. Wat we hiermee vaststellen is dat de systeembeheerder de hoogste technische kennis heeft.
Het volgende is dus het geval: het invulgedeelte moet eenvoudig zijn, maar ook geen aparte helper functie bevatten, het systeem moet binnen het bekende ritueel passen van een

formulier invullen en deze via de mail verzenden. Voor het beheer van de formulieren mag de interface complexer zijn, hier is bij het ontwerpen van de formulieren en hun workflow wel ruimte voor een helper functie. We zien dat het interessant kan zijn voornamelijk tips te verstrekken met betrekken tot hetgeen waar de gebruiker mee bezig is.
Voor de systeembeheerder is het interessant dat hij de mogelijkheid heeft voorbeelden op te vragen bij de te nemen stappen, bijvoorbeeld het weergeven van een voorbeeld bij het invoeren van een databasesource die verbindingen maakt met een externe medewerkersdatabase.
Binnen TAG is er opgemerkt dat, net zoals eerder genoemd, de handleidingen niet vaak genoeg gelezen worden. Wat resulteert in een geïrriteerde gebruiker, maar ook in overbodig contact met de helpdesk, deze 2 punten zijn de drijfveer achter het creëren van een intuïtieve interface.

discussie & conclusie
Voor een product als TAG eForms is het zeer interessant om de gebruikerskant device afhankelijk te maken. We zien terug dat een eForm op locatie in te vullen moet zijn, denk aan checklists ed. Voor de beheerszijde van het programma is het overbodig, omdat ten eerste enige rekenkracht vereist wordt van de client browser en omdat ten tweede TAG zijn klanten adviseert dat beheer en systeembeheer vanaf een reguliere pc plaats vindt.
Zoals eerder bleek is het voor de invul-kant geen toevoeging dat het systeem de toestand van de gebruiker achterhaalt, een belangrijke reden hiervoor is dat het doel van het product het maken en verwerken van formulieren is. Deze kunnen oneindig veel vormen aannemen en er is dus geen helper-functie te maken voor het formulier zelf. In het voorlopige plan van het systeem is het de bedoeling dat de reguliere gebruiker niet veel mogelijkheden heeft, het voordeel hier is dat deze dan ook een in verhouding een eenvoudige interface heeft.
In de design fase van TAG eForms zal rekening gehouden worden met de kennis die bij een bepaalde gebruikersrol verwacht kan worden. Naast een digitale handleiding, de intuïtieve interfaces en een mogelijkheid tot het snel opvragen van specifieke informatie voor de complexere handelingen is er geen noodzaak tot bijvoorbeeld een helper-functie. De invulzijde zou op zijn minst een mobiele versie moeten hebben, naast de conventionele GUI.
We zien dus dat een device dependant interface in beperkte mate interessant is voor TAG eForms. Hiernaast is het niet interessant om tijdens het gebruik van het programma door de gebruiker, vast te stellen tot welk type gebruiker de gebruiker hoort. Wel is het zeker belangrijk dat er goed nagedacht wordt welk type gebruikers er in welke rol (TAG eForms gebruikersrol) voornamelijk zitten. Zodat er rekening gehouden wordt met interface die bij die rol voornamelijk gebruikt wordt.

referenties
[1] Canvas Tag http://en.wikipedia.org/wiki/Canvas_(HTML_element)
[2] Firefox Feature Brainstorming, 2006.
[3] http://www.whatwg.org/
[4] Some Trends in Web Application Development, Mehdi Jazayeri
[5] Widgets, BitTorrent, content blocking: Introducing Opera 9 Beta, http://www.opera.com/press/releases/2006/04/20/
[6] Firefox Extension Promises Private P2P, Ed Oswald, 2006 http://www.betanews.com/article/Firefox_Extension_Promises_Private_P2P/1156452762
[7] Web-Application Development Using the ModelViewController Design Pattern, Avraham Leff, James T. Rayfield IBM T. J. Watson Research Center
[8] User Modeling in Human-Computer Interaction G Fischer (UMUAI 10th Anniversary Issue)
[9] Nonverbally smart user interfaces: postural and Facial expression data in human computer interaction, Bahr et al., 2007
[10] TAG eForms requirements definition (preliminary)
[11] Why Don't People Read the Manual?, David G. Novick en Karen Ward, 2006
[12] adaptiver user interface generation for web services, Jiang He en I-Ling Yen, 2007

15. Bronnen

[1] Onderzoek naar de toepasbaarheid van RUP en MoSCoW voor kleine software development projecten, VS van Beek dec 2008

[2] ExtJS http://www.extjs.com

[3] JSON (JavaScript Object Notation) http://www.json.org

[4] Firebug http://www.getfirebug.com

[5]http://unrev.stanford.edu/introduction/body_introduction.html (appendix)

[6] Extreme Programming (rapid development for web-based applications), Frank Maurer and Sebastien Martel, IEEE internet computing 2002, IEEE

[7] Software testing and analysis (process, principles and techniques, Mauro Pezzè and Michal Young, 2008, Wiley

[8] http://sourceforge.net/projects/nusoap/

[9] http://www.extremeprogramming.org/ (appendix)

[10] http://www.simpletest.org/