Afstudeerverslag

Herinrichting beheerorganisatie Peinemann

& Ontwikkeling configuratie- en incidentbeheerapplicatie

Rotterdam, 11 januari 2008

Student:	P. van Dam
	20030871
Examinatoren:	E.M. van Doorn
	T.H.M. Spaan
Opdrachtgever:	B. Quispel, Peinemann BV

REFERAAT

Auteur: Paul van Dam
Titel: Herinrichting beheerorganisatie Peinemann
Onderwerp: Herinrichten beheerorganisatie en ontwikkelen configuratie- en incidentbeheerapplicatie

In dit verslag wordt de afstudeeropdracht van de auteur omschreven bij Peinemann BV. Het volledige proces wordt van begin tot eind weergegeven met de bijbehorende gemaakte keuzes en uitleg hierover. Dit verslag heeft betrekking tot de volgende onderwerpen:

Trefwoorden:

- Herinrichting beheerorganisatie
- Configuratiebeheer
- Incidentbeheer
- ASP.NET 2.0
- C#
- SQL
- WQL (WMI Query Language)
- WMI (Windows Management Instrumentation)
- Stored Procedures
- Requirements Management
- Versiebeheer
- Intranet
- Webbased
- MVC Architectuur
- Unified Process
- Webservices
- OpenUP

Voorwoord

In dit afstudeerverslag doet de auteur verslag van de afstudeerperiode bij Peinemann BV. De afstudeerperiode is onderdeel van de afsluiting van de opleiding Informatica aan de Haagse Hogeschool.

De opdracht bestond gedurende 15 weken uit het herinrichten van de beheerorganisatie bij Peinemann en het daarbij ontwikkelen van een bijpassende webapplicatie, waarmee het mogelijk is om efficiënter dan daarvóór configuratiebeheer en incidentbeheer te implementeren.

Hierbij wil de auteur nog de volgende mensen bedanken voor hulp, ondersteuning en begeleiding:

Peinemann BV:

- Dhr. B. Quispel
- Dhr. T. Kanters

Haagse Hogeschool:

- Dhr. T. Spaan
- Dhr. E. van Doorn

Inhoudsopgave

1. Inleiding

Dit verslag is geschreven door Paul van Dam in het kader van het afstuderen aan de Haagse Hogeschool, afdeling ICT & Media, opleiding Informatica. Het verslag richt zich op het gevolgde proces en opgeleverde producten voor het afstudeerproject “Herinrichten beheerorganisatie” bij Peinemann BV in Hoogvliet.
De afstudeerperiode besloeg 18 weken, van september 2007 tot en met half januari 2008.

De bedoeling van dit verslag is, dat de lezer geleidelijk een duidelijker beeld krijgt van wat er tijdens deze 18 weken gebeurd is bij het werken aan de afstudeeropdracht. Ook hoopt de auteur zijn keuzes (en die van de opdrachtgever) in dit verslag kenbaar te maken en te verduidelijken.

Het verslag is primair bedoeld voor de examinatoren van de Haagse Hogeschool en de gecommitteerde, maar verder voor een ieder die geïnteresseerd is in het beheer van zowel incidenten als configuratie-items binnen een groeiende IT-omgeving.

Na deze inleiding wordt in het tweede hoofdstuk een beeld gegeven van de organisatie Peinemann. Hierna volgt een opdrachtomschrijving, waarna in drie hoofdstukken wordt beschreven hoe de auteur de opdracht heeft aangepakt. In het laatste hoofdstuk wordt de opdracht (zowel product als proces) geëvalueerd.

In het gehele verslag schrijft de auteur descriptief in de derde persoon, met één uitzondering. Het laatste hoofdstuk (evaluatie) is in de eerste persoon geschreven, om de eigen mening en ervaring te benadrukken.

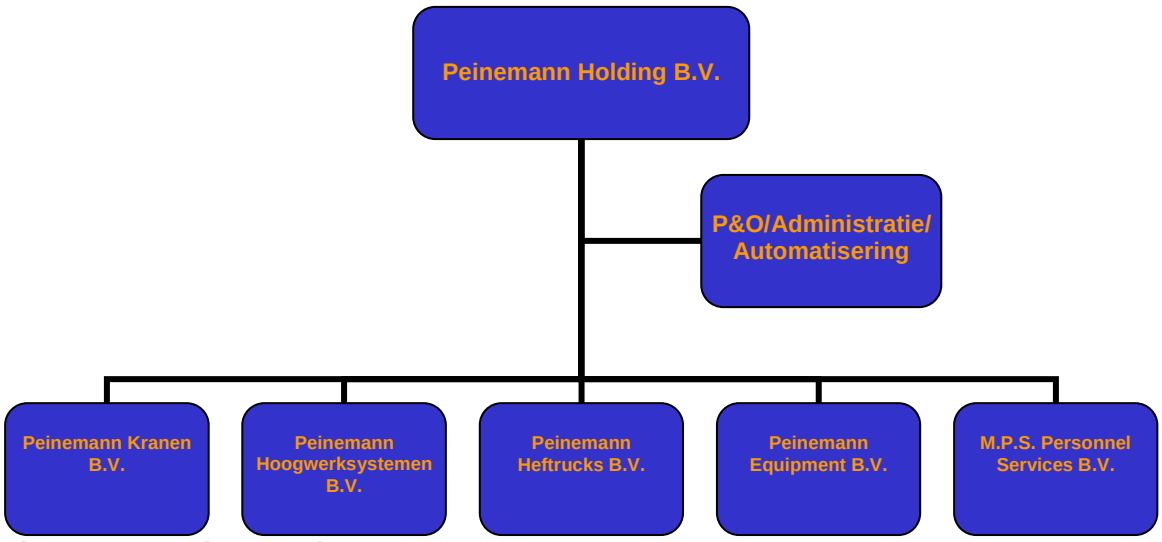
2. Peinemann BV

In dit hoofdstuk wordt de organisatie beschreven waar de afstudeeropdracht is uitgevoerd. Naast de organisatie wordt ook een toelichting gegeven over de afdeling waar de uitvoering van de opdracht heeft plaatsgevonden.

Peinemann B.V. is een bedrijf dat gericht is op het verhuren en in mindere mate verkopen van kranen, heftrucks en hoogwerksystemen. De klanten van Peinemann bevinden zich over het algemeen in de petrochemische industrie.
Peinemann is nu zo'n 50 jaar een begrip in de wereld van horizontaal en verticaal transport. Het bedrijf telt ongeveer 500 werknemers en het hoofdkantoor is gevestigd in Hoogvliet. Naast de vestiging in Hoogvliet, waar Peinemann uit totaal drie panden bestaat, heeft Peinemann Heftrucks nog een vestiging in Veendam. In Amsterdam heeft Peinemann een gecombineerde vestiging van Peinemann Heftrucks en Peinemann Hoogwerksystemen.
Per 1 januari 2008 is Crepa BV volledig geïntegreerd in de Peinemann groep. Crepa is een bedrijf dat zich vooral richt op het importeren, verhuren en verkopen van heftrucks.
Crepa heeft de hoofdvestiging in Rijswijk, met vestigingen in Leeuwarden, Assen, Venlo en Londerzeel (België).

2.1 Organigram

In figuur 1 is de organisatie van Peinemann BV schematisch weergegeven. Omdat de toevoeging van Crepa aan de Peinemann groep zo recent is, is nog geen combinatie ontstaan van Peinemann en Crepa in de organisatiestructuur. Zo is de IT afdeling naast de directie eigenlijk de enige afdeling die met beide bedrijven te maken heeft.



Figuur 1: Organigram Peinemann

Aan het hoofd van de Peinemann Holding staat Peter Peinemann als hoofdverantwoordelijke vanuit het familiebedrijf. Financieel directeur is Martin Houkes. Aan het hoofd van de verschillende BV's staan de managers, die rapporteren aan de Holding en voor ondersteuning afhankelijk zijn van de stafafdelingen Personeel & Organisatie, Administratie en Automatisering.

2.2 Afdeling Automatisering

Tot voor kort was de afdeling Automatisering een aanhangsel van de afdeling Administratie. Het hoofd administratie was tevens hoofd IT. Met de laatste ontwikkelingen in de groei van het bedrijf zijn de afdelingen echter gesplitst en is er naast het hoofd IT een andere persoon in dienst als hoofd administratie.

De afstudeeropdracht heeft plaatsgevonden bij de afdeling automatisering in Hoogvliet. Deze afdeling bestaat uit het hoofd IT, de systeembeheerder en de afstudeerder.

De afdeling is verantwoordelijk voor het onderhouden van de telecomapparatuur, het netwerk en de verschillende clients, servers en printers op alle locaties van de Peinemann groep. Met de toevoeging van Crepa aan de Peinemann groep is de afdeling verantwoordelijk geworden voor dezelfde onderdelen van Crepa BV.

Voor het beheer van deze onderdelen (configuratiebeheer, incidentbeheer, change management) wordt geen gebruik gemaakt van een specifieke methode. Om de organisatie transparant te houden voor de gebruikers, wordt een eenvoudig systeem van incidentmelding en het oplossen hiervan gebruikt. Geen van alle werknemers heeft een ITIL-opleiding gehad, waardoor deze methode niet gebruikt wordt. Hiervoor zou in de toekomst bij een verdere groei nog gekozen kunnen worden, maar omwille van transparantie voor een relatief kleine gebruikersgroep met min of meer gestandaardiseerde hardware en software is dit nu nog niet aan de orde.

2.3 De verschillende Peinemann bedrijven

Voor verhuur en verkoop van de verschillende productgroepen zijn de Peinemann BV's in het leven geroepen.

Peinemann kranen is primair gevestigd in Hoogvliet en houdt zich vooral bezig met het verhuren van kranen, inclusief bemanning. Peinemann Kranen biedt kranen aan met een hefvermogen tot 650 ton.

Peinemann hoogwerksystemen heeft een vestiging in Hoogvliet, op korte afstand van de hoofdvestiging van Peinemann BV. Peinemann hoogwerksystemen houdt zich bezig met het verhuren en verkopen van hoogwerkers en het uitvoeren van onderhoud aan dergelijke apparaten.

Peinemann heftrucks is min of meer evenredig verdeeld gevestigd in Hoogvliet en Amsterdam, met voor de regio Noord de vestiging in Veendam. Deze divisie houdt zich vooral bezig met het verhuren en verkopen van heftrucks. Met de overname van Crepa BV (dat zich alleen bezig houdt met heftrucks) is de samenwerking tussen Crepa en Peinemann heftrucks uitgebreid.

Peinemann Equipment is een gespecialiseerd bedrijf dat zich bezig houdt met verhuur en verkoop van bundeltrekkers en zogenaamde "bundle cleaners" in de petrochemische industrie. Deze divisie is gevestigd in de hoofdvestiging van Peinemann BV in Hoogvliet en is wereldwijd marktleider op het gebied van bundeltrekkers en –cleaners.

Naast de operationele Peinemann bedrijven heeft de Peinemann groep een eigen uitzendbureau voor machinisten, bundeliers, monteurs enzovoorts. MPS Personnel Services BV werkt dan ook nauw samen met de verschillende andere Peinemann bedrijven.

3. Opdrachtomschrijving

In dit hoofdstuk wordt de opdracht toegelicht. Zo wordt er beschreven wat de probleemstelling is met daarbij de uitgangssituatie. Vervolgens worden ook de doelstelling en daarbij op te leveren producten beschreven. Deze opdrachtomschrijving kan afwijken van de originele opdrachtomschrijving (die als bijlage is meegeleverd), in die zin, dat dit verslag geschreven is, terugkijkend op de afgesloten opdracht in plaats van vooruitkijkend op het begin van de opdracht.

3.1 Probleemstelling

Het aantal gebruikers (en daarmee ook het aantal PC's, GSM's, printers, servers, locaties en applicaties) is de afgelopen jaren sterk toegenomen. Hierdoor is het handmatig bijhouden van configuratiebeheer en incidentbeheer een steeds meer tijdrovende klus geworden. Bijkomend feit is dat het bedrijf Crepa is overgenomen door Peinemann. Dit heeft als gevolg voor het systeembeheer van Peinemann dat het aantal gebruikers, PC's, gsm's, printers, lokaties en vestigingen minimaal zal verdubbelen. De huidige organisatie van het beheer van de verschillende systemen is niet geschikt voor een zo grote gebruikersgroep en de bijbehorende extra hard- en software.

De groei van het bedrijf is dus de oorzaak van het tekortschieten van de beheerorganisatie, waardoor de hoeveelheid werk voor de IT-afdeling te groot is geworden. Vooral het handmatig bijhouden van configuratiebeheer en incidenten is nu te veel werk geworden. Het is vaak niet duidelijk (en niet meteen zichtbaar) welk configuratie-item welk probleem heeft en wat de status is van het probleem. Er is geen budget voor het aannemen van extra personeel op de IT-afdeling.

3.2 Uitgangssituatie

Bij het begin van het project was er een standalone programma aanwezig, genaamd Mobuscom, waarin de systeembeheerder en het hoofd IT informatie van PC's, gebruikers, mobiele telefoons, printers, lokaties en vestigingen op hebben geslagen.
Alle informatie die hierin opgeslagen is, is handmatig ingevoerd door de systeembeheerder en het hoofd IT (wat veel werk oplevert bij wijzigingen of toevoegingen).
Bij de herinrichting van de beheerorganisatie moest de functionaliteit en inhoud van deze applicatie in ieder geval meegenomen worden.

Ook was er aan het begin van het project een server aanwezig waarop IIS 5 en SQL Server 2000 waren geïnstalleerd. Om versiebeheer te kunnen toepassen heeft de afstudeerder op deze server tevens een Subversion server geïnstalleerd en ervoor gezorgd dat er voldoende back-ups van zowel de Subversion repository, de database server en de web server werden gemaakt.

Deze server bestaat uit meerdere SCSI RAID 1 array's, een Pentium III Xeon 933 MHz processor, 916 MB geheugen en als besturingssysteem Microsoft Windows 2000 Server SP4 Standard. In de toekomst zal deze server vervangen worden door een actueler model, hier wordt tijdens de ontwikkeling van een oplossing rekening mee gehouden.

Als werkstation werd gebruik gemaakt van een desktop pc met Pentium 4 2,4 GHz processor en 1,2 GB geheugen. Hierop geïnstalleerd waren: MS Visual Studio 2005 en MS SQL Server Management Studio 2005, op een basis van MS Windows XP Pro SP2. Ook MS Office 2003 en MS Visio 2002 waren bij voorbaat op het systeem aanwezig.

3.4 Doelstelling

De doelstelling van de afstudeeropdracht was een herinrichting van de beheerorganisatie, waaruit een Webapplicatie voortkwam waarmee het hoofd IT en de systeembeheerder inzicht kunnen krijgen in de status van de gehele IT-infrastructuur en waarmee de werkprocessen van de IT-afdeling efficiënter en eenvoudiger gemaakt kunnen worden. Computersystemen die aan het netwerk toegevoegd worden, kunnen van afstand worden uitgelezen op status van hard- en software.

3.5 Producten

Aan het einde van dit project dienden de volgende producten opgeleverd te zijn:

- Heringerichte beheerorganisatie
- Nieuwe configuratie- en incidentbeheerdatabase
- Nieuwe configuratie- en incidentbeheerapplicatie
- Alle bijbehorende broncode, modellen en overige documentatie
- Handleidingen voor het gebruik en beheer van de webapplicatie

3.6 Methoden en technieken

Er was vanuit de organisatie geen enkele verplichting voor het gebruik van bepaalde vastgelegde methoden en technieken. Uiteraard zou het uitvoeren van een dergelijk groot project zonder enige vorm van methodische aanpak problematisch worden, dus er is wel degelijk methodisch gewerkt.

Door de afstudeerder is ervoor gekozen om de ontwikkelmethode UP toe te passen in de verschillende vormen die hierin bestaan. Voornamelijk is OpenUP gebruikt omdat dit een open source oplossing biedt waarvoor voldoende documentatie is. Over OpenUP kan kostenloos geschreven worden en het kan kostenloos gebruikt worden, omdat de methode valt onder de Eclipse Public License.
De UP methode biedt de mogelijkheid tot een iteratieve werkwijze en geeft de ontwikkelaar de mogelijkheid om een selectie uit de op te leveren producten te nemen. Hierdoor was het mogelijk om alleen producten op te leveren die iets toevoegden aan het ontwikkeltraject of een eventueel vervolgtraject en niet te vervallen in min of meer nutteloze documentatie.

Dit ten opzichte van een methode zoals bijvoorbeeld IAD, die meer gericht is op ontwikkeling met behulp van pilots. Binnen dit project konden na requirements analysis duidelijk twee verschillende iteraties benoemd worden, wat het project buitengewoon geschikt maakte voor een UP-ontwikkeltraject.

Buiten de toepasbaarheid van UP op deze situatie, was de afstudeerder ook bekender met de UP-methode vanuit de opleiding, aangezien daarin veruit de meeste aandacht werd besteed aan UP.

Daarnaast is UP een flexibele methode, waarin niet per se van tevoren alle requirements duidelijk hoeven te zijn, hieraan kunnen tijdens het project wijzigingen optreden.

Voor het onderdeel “Requirements management” is “Managing Software Requirements” van D. Leffingwell en D. Widrig gebruikt. Dit is een boek en niet zo zeer een methode, maar de aanwijzingen die in dit boek gegeven worden zijn als leidraad gebruikt bij het opstellen en beheren van de requirements. Wat dat betreft zou het dus als een methodiek omschreven kunnen worden.

Bij het uitvoeren van de acceptatietesten is uitgegaan van de richtlijn die TMap biedt. Naamgeving van tests en indelen van de testomgeving hoort hierbij. Wat de auteur betreft kan TMap alleen maar omschreven worden als een richtlijn en niet als een methode, omdat in TMap nergens echt een werkwijze omschreven wordt, enkel conventies wat betreft naamgeving en de benadering van een testtraject.

Als modelleertechniek is door de afstudeerder gekozen voor UML, Unified Modeling Language. Omwille van de onderhoudbaarheid van de aan het eind van het project opgeleverde applicatie door een derde, was de keuze voor een techniek die als een branchestandaard beschouwd wordt snel gemaakt.
UML is tevens zeer geschikt om objectgeoriënteerde applicaties te modelleren. Vanuit Microsoft (Peinemann is volledig gericht op Microsoft software) wordt het gebruik van UML ten zeerste aangemoedigd, zo ook vanuit de opleiding. Daarnaast heeft de systeembeheerder ook kennis van UML en gebruikt hij dit veel bij zijn eigen projecten, dus kan hij de UML modellen lezen, begrijpen en in de toekomst eventueel bewerken.

4. Voorbereiding op het ontwikkeltraject

In hoofdzaak zou de opdracht bestaan uit twee hoofdbestanddelen. Ten eerste het herontwikkelen van een beheerorganisatie (en het daarbij herinrichten van het beheerproces). Hieruit ontstond daarna een webapplicatie die volledig aansloot bij het nieuwe beheerproces. Op deze manier kon, door middel van de nieuwe webapplicatie, het werken via het nieuwe beheerproces worden afgedwongen.

In de voorbereiding op het ontwikkeltraject van het nieuwe beheerproces en de webapplicatie is door de afstudeerder besloten om het proces van verzamelen en analyseren van requirements niet binnen het ontwikkeltraject te laten vallen, maar dit in een eerdere fase uit te voeren. Hier werd dus bewust van de vastgelegde UP-werkwijze afgeweken. Dit is zo gedaan om “Requirements Elicitation” (het “loskrijgen” van requirements bij de verschillende stakeholders) als losstaand te kunnen blijven beschouwen ten opzichte van het ontwikkeltraject.

Voordeel hiervan is, dat vóór het werkelijke begin van de werkzaamheden aan de opdracht bepaald kon worden hoe het project er zou gaan uitzien. Naast het vergelijken van verkregen requirements met features die al aanwezig zijn in commercieel verkrijgbare producten, of onder licentie bruikbare methodes zoals bijvoorbeeld ITIL, bood deze constructie de mogelijkheid om de requirements in te delen in verschillende iteraties binnen het ontwikkeltraject, in volgorde van prioriteit. Deze eerste fase kon hiermee gebruikt worden om een duidelijk afgebakende scope voor het project te definiëren, die een in het “Vision Document” gedocumenteerde visie opleverde.

Om de werkzaamheden binnen het project overzichtelijk te houden, is door de afstudeerder in overleg met de opdrachtgever dus gekozen voor een traject waarin volgens UP-normen gewerkt werd met een aantal basisiteraties, in dit geval twee (en vervolgens in feite drie, hierover later meer). Voordat het UP-proces begon, waren de requirements al grotendeels duidelijk in de voorbereidingsfase en werden er geen alternatieven meer overwogen tijdens het ontwikkeltraject.

Het was de bedoeling dat er aan het einde van iedere basisiteratie een werkend product werd opgeleverd, dat binnen het bedrijf in gebruik genomen kon worden.
In overleg met de opdrachtgever is dan ook bepaald, in welke iteratie aan welke requirements moest worden voldaan. Het werd op deze manier ook mogelijk om de opdrachtgever gedurende het project de mogelijkheid te geven tot het uitvoeren van acceptatietests, zodat constant geverifieerd kon worden of het product (de beheerorganisatie/-applicatie) nog voldeed aan de ideeën die de opdrachtgever hier over had (wat dus in de requirements duidelijk is geworden).

Binnen één iteratie werd aan het eind van iedere iteratiefase een “milestone” bereikt. Het bereiken van een milestone wil feitelijk zeggen dat alles wat vóór het bereiken van de milestone uitgevoerd is, wordt afgesloten en niet meer gewijzigd kan worden tot de volgende iteratie. Voordat een milestone gezet kon worden, doorgingen alle betrokken producten een keuringstraject bij de opdrachtgever. Eventuele onduidelijkheden of ongewenste projectonderdelen konden dan op korte termijn bewerkt of verwijderd worden. Een eventuele fout kost op deze manier zo weinig mogelijk tijd om te herstellen, aangezien deze al in een vroeg stadium wordt afgevangen. Aanverwant werd er, om projectvoortgang te bewaken en eventuele knelpunten zo snel mogelijk voor het voetlicht te brengen, aan het einde van iedere week een voortgangsrapportage opgeleverd aan de opdrachtgever. Hierin werd aangegeven wat er de afgelopen week zoal bereikt was, of dit voldeed aan de planning en in hoeverre er oponthoud ontstaan is door in die week tegengekomen knelpunten.

Binnen het ontwikkeltraject zou doorlopend veel aandacht worden besteed aan requirements management, wat een product opleverde dat volledig voldeed aan de voor de opdracht door alle partijen goedgekeurde requirements.

Door de afstudeerder werd gekozen voor het afnemen van interviews met stakeholders, om duidelijk te krijgen aan welke requirements de nieuwe beheerorganisatie moest gaan voldoen. Zo werden de

opdrachtgever (het hoofd IT), de systeembeheerder (in zijn twee rollen als systeembeheerder en toekomstige applicatiebeheerder van de op te leveren webapplicatie) en enkele gebruikers gevraagd naar hun mening over de huidige beheerorganisatie en de punten waarop zij mogelijkheden zagen tot verbetering. De requirements van de gebruikers werden achterhaald door het uitgeven van enquêteformulieren. De antwoorden werden verwerkt in de requirements.

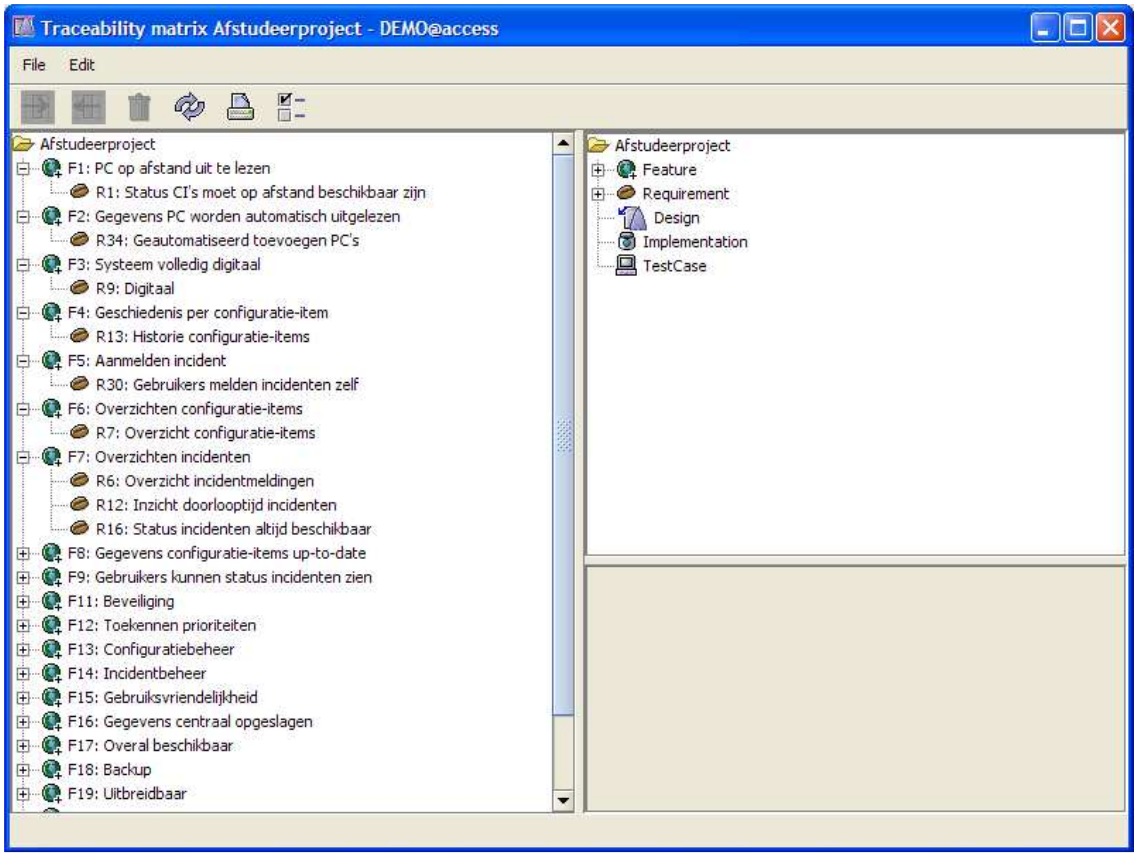
4.1 Requirements & OSRMT

In overleg met de opdrachtgever werd bepaald welke requirements bij de achtereenvolgende iteraties hoorden aan de hand van het toekennen van prioriteiten. Zo werden in eerste instantie de requirements met een hoge prioriteit ingedeeld in iteratie één. De overige requirements werden in iteratie twee ingedeeld.

Door de omvang van het project en de diepgang van sommige requirements en bijbehorende features werd echter al snel in samenspraak met de opdrachtgever besloten dat de requirements met de laagste prioriteit ook in iteratie twee niet aan bod zouden komen. Deze requirements werden dan ook in het vervolg gezien als “could haves” die in een voorlopig fictieve iteratie drie terug naar voren zouden komen.

Er werd dus gesteld dat de ontwikkeling van de beheerorganisatie en bijbehorende applicatie aan het einde van dit afstudeerproject weliswaar een werkbare status zou hebben (en hierbij verbeterd ten opzichte van de reeds bestaande situatie), maar dat er door de toekomstige beheerder nog het één en ander aangepast zou kunnen worden om te voldoen aan de overgebleven requirements. De requirements werden beheerd in de Open Source applicatie OSRMT (Open Source Requirements Management Tool). Voor deze applicatie is door de opdrachtnemer gekozen omdat het gebruik hiervan geen kosten met zich meebrengt. Verder is de applicatie veel eenvoudiger dan bijvoorbeeld de beschikbare pakketten van Rational en diende deze precies het doel van de beginfase van deze opdracht, namelijk:

- Het duidelijk krijgen van de requirements
- Het toekennen van features aan de requirements
- Het toekennen van prioriteiten aan de features en requirements ten behoeve van een planning in iteraties
- Traceability van requirements en features



Figuur 2: Traceability matrix

In bovenstaande afbeelding is (een onderdeel van) de traceability matrix van de applicatie OSRMT weergegeven. Hier is te zien dat een er een duidelijk overzicht is van features en requirements (Items met een F staan voor features en zijn gekoppeld aan items met een R, die staan voor requirements).

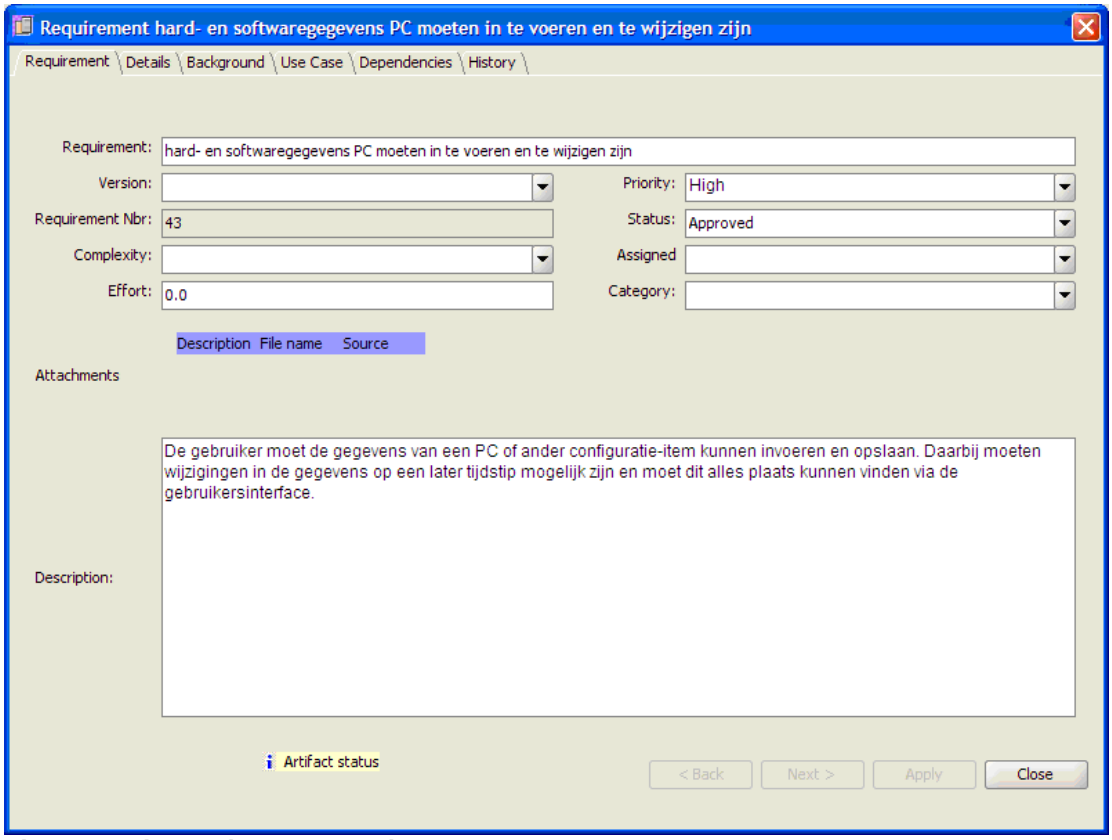
Als iedere requirement in OSRMT is gekoppeld aan een feature, wil dit zeggen dat de som van de genoemde features een oplossing vormt voor het gestelde probleem en in een ideale situatie (waarbij gedetailleerde requirements aanwezig zijn) de enige mogelijke oplossing is die aan alle requirements voldoet. Deze laatste mogelijkheid is echter meer theoretisch van aard, omdat het zo goed als onmogelijk is om van tevoren 100% van de requirements volledig duidelijk te hebben.

Op het moment dat de traceability matrix volledig ingevuld was, was er in feite een theoretische oplossing voor 80% aanwezig, die enkel nog in praktijk gebracht diende te worden.

Natuurlijk kwamen er dus in de loop van het project zo nu en dan requirements bij. Naar het onmogelijke doel van 100% duidelijkheid in requirements werd overigens wel gestreefd om toch de kosten (in tijd) van een laat in het project toegevoegde requirement (en de bijbehorende feature) te drukken.

Hierbij is met de opdrachtgever afgesproken dat een tijdens een iteratie aan het licht gekomen requirement niet meer bij de huidige iteratie ingedeeld kon worden, maar op zijn vroegst in de daaropvolgende iteratie kon worden behandeld.

OSRMT biedt ook de mogelijkheid om een uitgebreide omschrijving per requirement of feature in te vullen, alsmede een prioriteit, zoals in onderstaande afbeelding te zien is.



Figuur 3: Eigenschappen requirements

Op deze manier is dus te allen tijde duidelijk vastgelegd wat de inhoud is van een requirement en kan dit ook voor alle betrokken partijen (uiteraard na wederzijdse goedkeuring) gebruikt worden om eventuele afwijkingen hiervan aan de kaak te stellen.

Hoewel OSRMT over meer functionaliteit beschikt dan bovenstaand wordt getoond, is er door de afstudeerder voor gekozen om gebruik te maken van het minimaal nodige in OSRMT. Op deze manier werd het overzicht behouden en werd er minder tijd besteed aan relatief minder belangrijke details. Een naam, omschrijving en prioriteit waren voldoende om een requirement en/of feature accuraat te omschrijven.

Omdat use cases niet altijd gelijk lopen met requirements en sommige use cases bij meerdere requirements horen en andersom zou het een te warrig geheel worden om requirements en features ook nog aan use cases te koppelen. Ook hier is dus omwille van het behouden van een mate van overzichtelijkheid van afgezien.

Uiteindelijk werden de requirements vastgesteld op onderstaande lijst:

Figuur 4: Requirements

Aan de hand van deze requirements kon een vergelijking worden gemaakt met commercieel verkrijgbare alternatieve methoden en applicaties.

In overleg met de opdrachtgever werd sowieso besloten dat een veelgebruikte beheermethode als ITIL te ingewikkeld en te uitgebreid zou zijn voor het gebruik bij Peinemann. Hoewel de beheerorganisatie met de groei van het bedrijf steeds meer geprofessionaliseerd werd, liet de omvang in personeel van de automatiseringsafdeling het voldoen aan ITIL niet of alleen in sterk verminderde mate toe.
Daarom werd dan ook door de opdrachtgever besloten dat de beheerorganisatie ten opzichte van ITIL versimpeld diende te worden. Er moest uitgegaan worden van efficiëntie van incident- en configuratiebeheer. De eerste UP-iteratie was dan ook volledig geënt op het herinrichten van het

configuratie- en incidentbeheer naar een nieuwe situatie, die al bruikbaar zou zijn voor Peinemann (naast de bestaande beheerorganisatie in een productieteststadium).

In deze fase werd ook in overleg met de opdrachtgever definitief besloten dat er geen commercieel verkrijgbaar alternatief gebruikt zou worden, hij wilde specifiek een voor en door Peinemann ontwikkeld systeem dat volledig in de op Microsoft-gerichte Peinemann infrastructuur zou passen en daarnaast aan zou sluiten op de Peinemann huisstijl. Bovendien was er geen budget beschikbaar voor de aanschaf van licenties op applicaties.

Als voorbeeld van een alternatief product is er wel gekeken naar de applicatie TOPdesk, die voorziet in een op ITIL gebaseerd incidentbeheer en configuratiebeheer (bij aanschaf van deze modules). TOPdesk is echter geen goedkoop pakket en bovendien geënt op omvangrijkere automatiseringsafdelingen. Het bijwerken van gegevens in TOPdesk zou uiteindelijk evenveel of zelfs nog meer tijd kosten als in de bestaande situatie werd besteed. Het gebruik van applicaties zoals TOPdesk werd daarom door alle partijen in deze fase van de hand gewezen.

Aan de requirements “configuratiebeheer” en incidentbeheer zou dus aan het einde van de eerste iteratie al voldaan moeten zijn. Bepaalde andere requirements konden daar ook al bij horen, maar waren niet noodzakelijk voor het “live gaan” van de applicatie en de bijbehorende nieuwe beheerorganisatie (in ieder geval voor de automatiseringsafdeling, waarbij de gebruikers hun meldingen nog steeds telefonisch of via mail indienden, waarna de systeembeheerder de gegevens in de applicatie bijwerkte).

Requirements die voor diepgang in de opdracht zorgden en verfijningen aan de applicatie, werden geplaatst in de tweede iteratie en requirements die pas in de toekomst aan de orde zouden komen en dus een lage prioriteit hadden, kwamen terecht in iteratie drie, een iteratie waaraan binnen dit project verder geen aandacht werd besteed.

4.2 Projectorganisatie iteraties

Leffingwell & Widrig (Managing software requirements, zie verder literatuurlijst) stellen dat een projectteam bij ieder project moet voldoen aan bepaalde vaardigheden. Aangezien het projectteam bij dit project in feite bestond uit één persoon, de afstudeerder, moest deze voldoen aan alle vaardigheden. Een omvangrijk project als dit toekennen aan één persoon heeft voordelen en nadelen.

Één van de voordelen is dat de betreffende persoon de volledige oplossing in één structuur kan inpassen zonder dat er communicatieproblemen optreden. Bijkomend voordeel is dat deze persoon gedurende het hele traject op de hoogte is van alle wijzigingen en hierover de controle heeft.

Een nadeel, naast het feit dat er veel werk op de schouders van één persoon rust, is dat er maar vanuit één standpunt gekeken wordt naar het probleem, waardoor ook de oplossing alleen vanuit dat standpunt aangedragen kan worden. Eventuele fouten of misvattingen worden op deze manier ook niet snel opgemerkt, mede omdat het vervullen van alle rollen binnen een project een zware taak is voor één persoon.
Bovendien wordt binnen de meeste UP-methoden niet aangeraden om bijvoorbeeld de rol van ontwikkelaar en de rol van tester aan één en dezelfde persoon toe te kennen.
Om deze laatste reden is ook gekozen voor de tijdelijke incorporatie van extra projectleden tijdens de testfase, namelijk de opdrachtgever en de uiteindelijke gebruikers.

De vaardigheden die Leffingwell & Widrig noemen:
- Analyzing the problem
- Understanding user and stakeholder needs
- Defining the system
- Managing scope
- Refining the system definition
- Building the right system

Aan al deze vaardigheden kan worden voldaan door veel aandacht te besteden aan requirements management. Hiermee wordt dus het belang van requirements management in een project aangetoond, zeker in het geval van min of meer grootschalige projecten valt hier veel winst te behalen.

Nadat het grootste deel van de requirements duidelijk was en ingedeeld was naar prioriteit (80% van de uiteindelijke requirements wordt door Leffingwell & Widrig gezien als een streefpercentage) kon er pas begonnen worden aan het daadwerkelijke ontwikkeltraject en de bijbehorende iteraties.

4.3 Plan van aanpak

Het plan van aanpak had als doel, om de toekomstige projectactiviteiten zo gedetailleerd en correct mogelijk uiteen te zetten. Zo werden in het plan van aanpak in feite afspraken gemaakt met de opdrachtgever over op te leveren producten en werkwijzen.
Projectorganisatie was ook onderdeel van het plan van aanpak, om vast te leggen wat er van alle personen, die met het project te maken hadden, werd verwacht. Tevens is een analyse van aanwezige en voorziene risico's in het plan van aanpak opgenomen.
Het plan van aanpak herhaalt in het kort nog de opdrachtomschrijving om er zeker van te zijn dat er geen last-minute wijzigingen in de opdracht zijn geweest. Om risico's zo vroeg mogelijk duidelijk te hebben en deze tevens zoveel mogelijk te beperken is een Risk Assessment al opgenomen in het plan van aanpak, in plaats van dit pas in de eerste Inception fase van het project te doen.

De onderdelen van het plan van aanpak die nog niet in het vorige hoofdstuk genoemd zijn, worden hier nog in het kort omschreven.

4.3.1 Stakeholders

In het plan van aanpak moesten alle stakeholders opgenoemd worden, om een duidelijk beeld te krijgen van de personen die met de beheerorganisatie te maken hadden en zouden gaan krijgen. Deze werden vastgelegd in de volgende lijst, met hun bijbehorende verantwoordelijkheden:

Afstudeerder
Gedurende de afstudeerperiode werkte deze aan het ontwikkeltraject, de bijbehorende herinrichting van de beheerorganisatie en het eindverslag. De afstudeerder legde voor het project verantwoording af aan de opdrachtgever.

Hoofd IT
Het hoofd IT bij het afstudeerbedrijf was opdrachtgever voor het afstudeerproject. Hij is tevens één van de belangrijkste gebruikers van de uiteindelijk opgeleverde applicatie. Het geven van feedback op de, door de afstudeerder opgeleverde, producten was binnen dit project zijn hoofdverantwoordelijkheid.

Systeembeheerder
De persoon die de uiteindelijke applicatie het meest zou gaan gebruiken. Ook was de systeembeheerder vraagbaak gedurende het project, en applicatiebeheerder van de opgeleverde software. Het leveren van feedback behoorde bij zijn verantwoordelijkheden.

Eindgebruikers
De personen die de applicatie zouden gaan gebruiken voor incidentmelding. Het geven van feedback op producten of gerichte vragen van de afstudeerder was binnen dit project hun hoofdverantwoordelijkheid, naast een gebruikerstest.

4.3.2 Risk Assessment

In de Risk Assessment zijn de risico's van het project beschreven. Deze risico's zijn opgesteld naar aanleiding van ervaringen met eerdere projecten. Hieronder worden nog in het kort de risico's doorgenomen, die op dit project van toepassing waren.

1. Eventuele kostbare aanschaf van licenties voor benodigde software. Het project zou hierdoor beperkt kunnen worden in de voortgang
2. Beperkte tijd die beschikbaar was voor het uitvoeren van het project.
3. Hardwarematige uitval.
4. Afhankelijkheid van één persoon, de afstudeerder.
5. De opdrachtgever had het druk met andere projecten.
6. Ter zake doende kennis van de opdrachtnemer betreffende het onderwerp zou onvoldoende aanwezig kunnen zijn geweest.

Om deze risico's te beperken en zo mogelijk teniet te doen, zijn onderstaande maatregelen getroffen:

1. Aan het begin van het project is er zoveel mogelijk gezorgd dat benodigde licenties aanwezig waren (bijvoorbeeld Office, Visual Studio etc.). Mochten er lopende het project nog bepaalde tools nodig zijn geweest, werd er gezocht naar oplossingen die binnen een General Public License vallen, waardoor deze licentiekostenvrij gebruikt kunnen worden. Voorbeelden hiervan zijn de Requirements Management tool OSRMT en de gehele client-server structuur van Subversion en TortoiseSVN.
2. Er is gewerkt met iteraties. Iedere requirement en bijbehorende feature kreeg een prioriteit toegekend, waardoor duidelijk werd welke functionaliteiten het belangrijkste waren voor de verschillende stakeholders. Mocht de afstudeerder dan in tijdnood zijn gekomen, kon in ieder geval een product worden opgeleverd dat voldeed aan de belangrijkste requirements.
3. De opslagruimte voor alle projectdocumentatie, modellen, database en broncode bevond zich op een server die allereerst van een RAID1-oplossing was voorzien, wat betekende dat gegevens op twee fysieke harde schijven tegelijk werden opgeslagen (mirroring). Als er dan één schijf zou zijn uitgevallen, zou de data nog van de andere schijf bereikbaar zijn. Verder wordt binnen het bedrijf iedere nacht een back-up gemaakt van belangrijke bestanden op alle servers. Als laatste maatregel om dit risico te beperken is er gebruik gemaakt van versiebeheersoftware, in dit geval een Subversion server. Op meerdere systemen werd consequent een update uitgevoerd zodat de laatste versie van alle bestanden altijd nog op een andere lokatie aanwezig was mocht er bijvoorbeeld brand uitbreken in de serverruimte. Verder werd op willekeurige momenten tijdens het project een volledige back-up van de projectbestanden gemaakt op een USB-geheugenstick.
4. Tegen het risico van ziekte of een andere vorm van langdurige afwezigheid bij de opdrachtnemer bestond niet echt een waterdichte maatregel. Wel was er voor noodgevallen een VPN verbinding met het bedrijf beschikbaar, zodat eventueel vanaf een andere locatie gewerkt zou kunnen worden.
5. Door drukte van de opdrachtgever met andere belangrijke projecten, had hij af en toe niet veel tijd om de voortgang van dit project te controleren of te beïnvloeden. In gevallen dat de opdrachtgever weinig aanwezig was, verliep de meeste communicatie telefonisch of via e-mail. Er werd voor gezorgd dat er te allen tijde een correcte en volledige informatiestroom naar de opdrachtgever liep, door middel van wekelijkse voortgangsrapportages. Door te zorgen dat de belangrijkste requirements al vanaf het begin duidelijk waren, was veel verdere inbreng van de opdrachtgever niet langer noodzakelijk.
6. Eventuele problemen betreffende onvoldoende kennis en kunde bij de opdrachtnemer zijn tijdens het project opgelost door het uitvoeren van onderzoeken, zo nodig naar alternatieven. Ook was de aanwezige systeembeheerder een belangrijke bron van informatie, aangezien hij veel ervaring had op hetzelfde gebied.

4.3.3 Planning

Belangrijk onderdeel van het plan van aanpak was de planning voor het volledige project. Algemeen gezien is deze planning redelijk tot goed gevolgd, met slechts enkele afwijkingen, te lezen in de voortgangsrapportages (zie bijlagen).

Week	Activiteit(en)
1	Inrichten werkplek (hard- en software), inlezen beginsituatie, beginnen met Inception fase 1 (vaststellen requirements), Plan van Aanpak
2	Inception fase 1, Requirements Elicitation/Analysis
3	Inception fase 1, Business Analysis, Business Modeling
4	Inception fase 1, Business Modeling, Use Cases nieuwe situatie, Lifecycle Objective Milestone
5	Elaboration fase 1, UML Modellen (Class Model, Data Model)
6	Elaboration fase 1, UML Modellen, Architectural proof of concept
7	Elaboration fase 1, Lifecycle Architecture Milestone, Construction Fase 1, UML modellen, programmeren applicatie
8	Construction fase 1, programmeren applicatie
9	Construction fase 1, programmeren applicatie, testcases, testplan
10	Construction fase 1, testrapport, onderzoeken van testbevindingen en eventueel aanpassingen maken aan applicatie, Initial Operational Capability Milestone, beginnen aan Transition fase 1
11	Transition fase 1, uitrollen applicatie, schrijven van handleidingen en eventueel ander lesmateriaal, geven van uitleg aan gebruikers, Product Release Milestone, beginnen aan Inception fase 2
12	Inception fase 2, Use Cases, Lifecycle Objective Milestone, Vision Document, Requirements Elicitation/Analysis, eventueel Plan van Aanpak.
13	Elaboration fase 2, UML modellen, eventueel Architectural Proof of Concept, Lifecycle Architecture Milestone
14	Construction fase 2, UML modellen, programmeren applicatie, testen van nieuwe functionaliteit, Initial Operational Capability Milestone, beginnen aan Transition fase 2, uitrollen applicatie
15	Transition fase 2, bijwerken handleidingen, instrueren gebruikers, Product Release Milestone

Figuur 5: Planning

5. Iteratie 1: Configuratie- en incidentbeheer

Voor iteratie 1 was dus besloten om vooral te werken aan de onderdelen configuratiebeheer en incidentbeheer, om zoveel mogelijk van de nieuwe beheerorganisatie te verwerken in de eerste iteratie.

Een UP-iteratie bestaat doorgaans uit de volgende fases:

- Inception
- Elaboration
- Construction
- Transition

Om een passende applicatie te schrijven bij de nieuwe beheerorganisatie, moest na het schrijven van het plan van aanpak eerst het beheerproces gewijzigd worden, dit was voor dit project het begin van de eerste Inception fase.

5.1 Inception Iteratie 1

In de Inception fase van een UP project dienen verschillende doelen bereikt te worden.
Allereerst diende de afstudeerder te weten wat voor oplossing er bedacht moest worden.
Dit bereikte deze, door middel van Requirements Analysis en bijbehorend contact met opdrachtgever en eindgebruikers en was in feite voor het grootste deel al duidelijk in de voorbereidingsfase.
Ook kon door middel van Requirements Analysis bepaald worden, welke requirements het belangrijkst waren voor de opdrachtgever en overige stakeholders.

Door het opstellen van een use case model kon nagedacht worden over een oplossing waarbij de wijzigingen in sommige bedrijfsprocessen aan de orde kwamen. Uiteindelijk moest binnen deze fase duidelijk zijn wat het project zou gaan kosten, zowel in geld als in tijd. Daarbij moesten de opdrachtgever en afstudeerder aan het einde van deze fase op de hoogte zijn van de risico's waaraan het project onderhevig is. In het geval van dit project waren de risico's al bepaald en geanalyseerd in de voorbereidingsfase.

Bijkomend bij een project als dit, waarbij niet alleen een informatiesysteen gebouwd moet worden, maar in feite meerdere business processes opnieuw moesten worden gedefinieerd (in die zin, dat verschillende use case definitions er anders uit zouden gaan zien), was het belangrijk dat de bestaande situatie eerst diepgaand geanalyseerd werd, waarna een duidelijk beeld zou moeten ontstaan van de bestaande beheerorganisatie. Dit werd bereikt door het gebruik van de UML Business Extensions.

Producten die uit deze fase voort moesten komen, zijn als volgt:

- Project Glossary
- Risk Assessment (Bij dit project reeds opgeleverd als onderdeel van plan van aanpak)
- Vision Document
- Business Activity Diagram
- Business Case
- Use Case Model

5.1.1 Project Glossary

Dit document gold als belangrijk gereedschap bij de communicatie tussen de verschillende bij het project betrokken personen. In dit document werden alle in overige documentatie gebruikte termen, vooral die van technische aard, op een duidelijke manier uitgelegd zodat een enigszins technisch onderlegde leek deze zou kunnen begrijpen. Dit was vooral nodig omdat er binnen het project veel gebruik zou worden gemaakt van nieuwe technologieën waar binnen het bedrijf nog geen kennis over was.

5.1.2 Vision Document

Het Vision document geeft vooral een verduidelijking vanuit de opdrachtnemer naar de opdrachtgever over de te ontwikkelen oplossing. Hiernaast werd het Vision document ook gebruikt als naslagwerk voor de opdrachtnemer.
De inhoud van het Vision document is deels ontstaan door de analyse van requirements die in interviews en enquêtes met de opdrachtgever, de systeembeheerder en enkele gebruikers naar voren kwamen.
In het Vision document kwamen onder andere de volgende zaken naar voren:

1. Project scope
2. Positionering product (probleemomschrijving)
3. Stakeholders en de bijbehorende needs, features & requirements
4. Product overview
5. Constraints
6. Documentation requirements

Al deze informatie kunt u terugvinden in de externe bijlage “Vision document”. Hieronder wordt nog in het kort aangegeven wat er met bovenstaande termen precies bedoeld wordt.

1. Een omschrijving van waar het project precies voor bedoeld is.
2. De aanleiding van het starten van het project, tevens probleemomschrijving: Een toename van het aantal gebruikers en het aantal te beheren configuratie-items en lokaties maakt een herinrichting van de beheerorganisatie noodzakelijk.
3. Een gedetailleerde omschrijving van alle stakeholders (belanghebbenden, eindgebruiker, hoofd IT, systeembeheerder en applicatiebeheerder) binnen het project wordt gegeven, met per stakeholder de betreffende initial needs, features & requirements.
4. Hierin werd omschreven wat er met de oplossing zoal bereikt moest worden, de zogeheten applicatiecompetenties.
5. Na de definitie van een scope voor het project is het nodig om de scope te begrenzen door middel van constraints.

De documentatie bij het project moest leesbaar zijn voor een eventuele opvolger, die de applicatie verder zou kunnen ontwikkelen. Daarom moest de applicatie leesbaar en volledig zijn en een helder beeld schetsen van de geproduceerde oplossing en de stappen die daarvoor genomen zijn. Ook documentatie voor gebruikers (handleidingen) moest kwalitatief hoogstaand zijn, om onbegrip bij het gebruik van het nieuwe systeem te voorkomen.

Het configuratiebeheeronderdeel moest uiteraard alleen toegankelijk worden voor de beheerders en zo ook het volledig functionele incidentbeheer. Beheerders moesten configuratie-items kunnen toevoegen, wijzigen en verwijderen en incidenten kunnen aanmaken, wijzigen, verwijderen en afmelden.
Eindgebruikers moesten incidenten kunnen aanmelden, de status van hun incidenten kunnen bekijken en ook de historie van opgeloste incidenten moest eenvoudig voor de gebruiker zichtbaar zijn.

Deze onderdelen zijn in feite de belangrijkste onderdelen voor de herinrichting, aangezien hierin alle programmaonderdelen voor de eindgebruikers vertegenwoordigd zijn. Het gedeelte voor eindgebruikers bestaat uit:

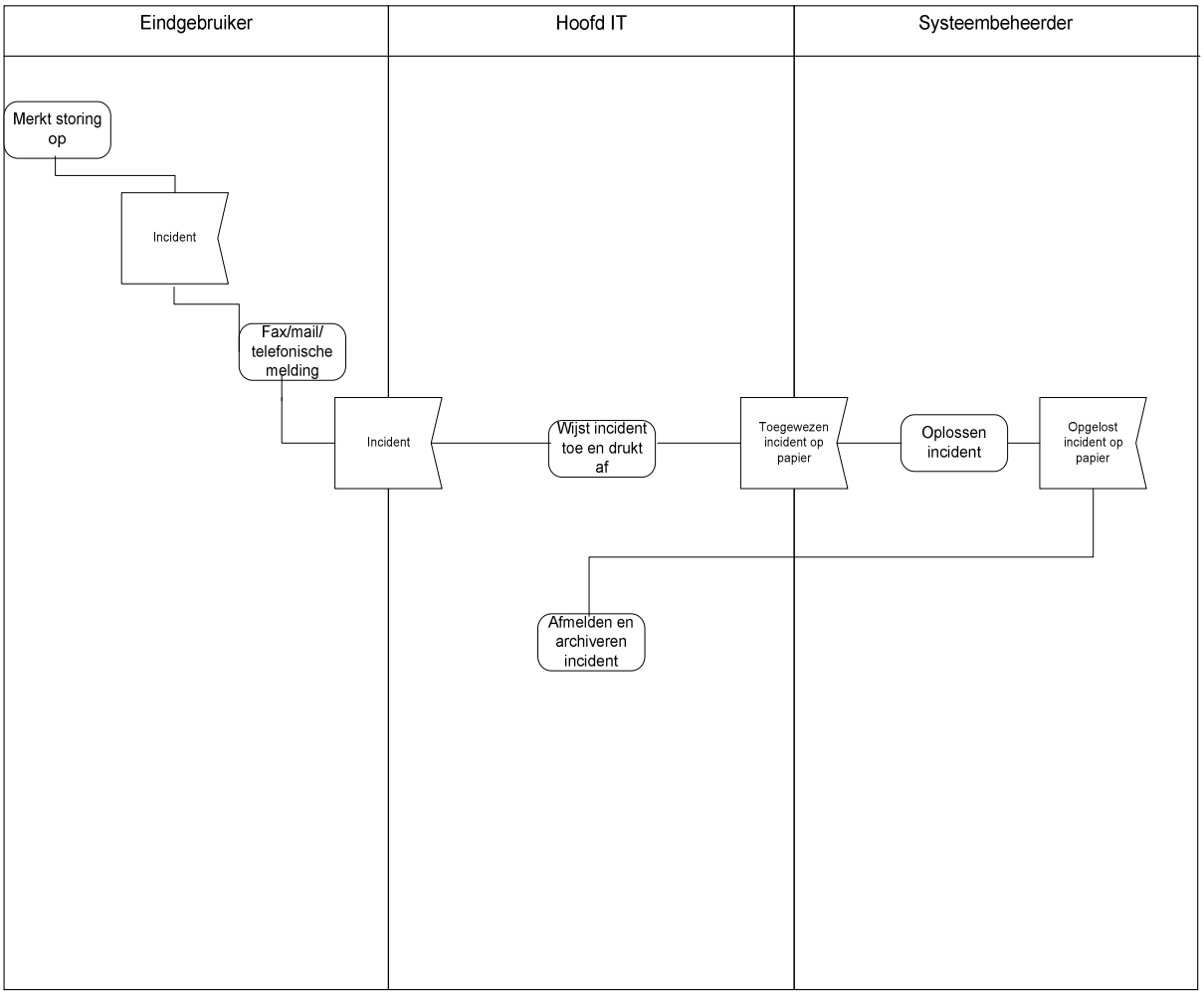
- Aanmelden incidenten
- Openstaande incidenten
- Incidenthistorie

Voor beheerders:

- Aanmelden incidenten (uitgebreid)
- Openstaande incidenten (uitgebreid)
- Incidenthistorie (uitgebreid)
- Configuratiebeheer

5.1.3 Business Activity Diagram

Om een duidelijk beeld te schetsen van de bestaande situatie binnen de Peinemann beheerorganisatie, was het noodzakelijk om de huidige processen van incidentmelding in kaart te brengen. Een goed gereedschap daarbij bleek het Business Activity Diagram, onderdeel van de UML Business Extensions. Het Business Activity Diagram vindt u terug in onderstaande afbeelding.



Figuur 6: Business Activity Diagram

Hieruit bleek dat tijdwinst mogelijk was door het inkorten van communicatielijnen en door het verminderen zo niet afschaffen van handmatig documentbeheer.

In het diagram is te lezen dat er in het geval van bijvoorbeeld een telefonische melding eerst nog het één en ander moet gebeuren bij het hoofd IT voordat de incidentmelding aankomt bij de systeembeheerder.
Zeker als het hoofd IT druk is met andere projecten en daardoor niet op zijn plaats aanwezig is, duurt het soms enkele dagen totdat een incidentmelding de systeembeheerder bereikt.

Door ervoor te zorgen dat een incidentmelding van een eindgebruiker direct in het incidentbeheer terechtkomt is er dus al een behoorlijke tijdwinst te behalen. In het geval van afwezigheid van het hoofd IT kan de prioriteit van een incident dan bepaald worden door de systeembeheerder zelf.
Ook het afmelden van een incident kan efficiënter door hier een geautomatiseerde actie van te maken. De archiefmappen zijn in de bestaande situatie al jarenlang gegroeid waardoor zoeken hierin lastig geworden is en het archief neemt flink wat ruimte in beslag. Door dit alles te digitaliseren kan de ruimte bespaard blijven en kan het zoekproces verkort worden.

In feite werd hiermee dus bepaald dat een incident drie verschillende statussen kon hebben, namelijk “Aangemeld”, “In Behandeling” en “Afgemeld”. Dit zou de kern worden van het incidentbeheer in de nieuwe situatie.

5.1.4 Business Case

Bij de analyse van de processen achter de huidige beheerorganisatie was het opstellen van een Business Case noodzakelijk, waarin nog eens duidelijk gesteld is waarom het nodig is om een applicatie te ontwikkelen voor het doel van de herinrichting van de beheerorganisatie en wat hier zoal voor nodig is.
Ook is de Business Case een nuttig hulpmiddel bij het vastleggen van het doel dat een informatiesysteem nastreeft ten opzichte van een niet geautomatiseerd systeem.

Hiernaast wordt in de Business Case vooral nog omschreven welke mogelijkheden en beperkingen er zijn op het gebied van hardware, software en licenties.

Onderdelen van de business case worden hieronder nog beschreven.

Productomschrijving
Hierin werd vastgelegd hoe de applicatie er qua componenten uit zou gaan zien, de applicatie zou bestaan uit de onderdelen configuratiebeheer en incidentbeheer. Hierin werd ook omschreven dat informatie over hardware en software op een PC uitgelezen zou kunnen worden op afstand.

Business context
De omschrijving van het type bedrijf (de context waarin het project gestart is), is als volgt:
De applicatie is bedoeld voor een bedrijf dat zich in de kern richt op het verhuren en importeren van kranen, heftrucks, hoogwerkers en dergelijke producten. In feite heeft de applicatie niet zoveel te maken met het type bedrijf, maar meer met het feit dat alle kantoormedewerkers en sommige logistieke en technische medewerkers gebruik maken van computers en het bijbehorende netwerk, servers, printers en software. In dit geval komen daar ook nog de mobiele telefoons bij.
Het bedrijf is één van de weinige middelgrote bedrijven in de non-ICT tak die alle hardware, het volledige netwerk en de meeste software in eigen beheer hebben. In deze bedrijfstak is dat tegenwoordig redelijk uitzonderlijk.
Om deze reden was het dan ook wenselijk om een goed functionerend en overzichtelijk configuratiebeheer te hebben. Gezien de beperkte bezetting op de afdeling IT en de uitbreiding van de rest van het bedrijf is besloten om de procedure van incidentmelding en –beheer te gaan aanpassen, om zo voor het hoofd IT en de systeembeheerder en makkelijkere en overzichtelijkere werksituatie te creëren die voor de eindgebruikers een oplossing oplevert die even punctueel en kwalitatief goed is als de oplossing in de oude situatie en zo mogelijk nog sneller en beter.

Product objectives
De in de opdrachtomschrijving genoemde doelen moesten met het uitvoeren van het project behaald worden. Daarbij dienden de requirements, die in de voorbereidingsfase waren opgesteld, gestalte te worden gegeven door middel van het aanbrengen van bijpassende features. Naar ratio van prioriteit moest met het opgeleverde product (zowel na de eerste als na de tweede iteratie) voldaan zijn aan deze requirements.

Constraints
Het project en product dienden te voldoen aan een aantal constraints die golden binnen het idee van hoe de applicatie moest gaan werken, en eruit moest gaan zien,
Ten eerste diende de te ontwikkelen software (en de tools die gebruikt worden bij de ontwikkeling) te draaien op het Microsoft Windows client-server platform, met eventueel gebruik van Microsoft tools voor ondersteuning, zoals Internet Information Services, Visual Studio, SQL Server en Office.

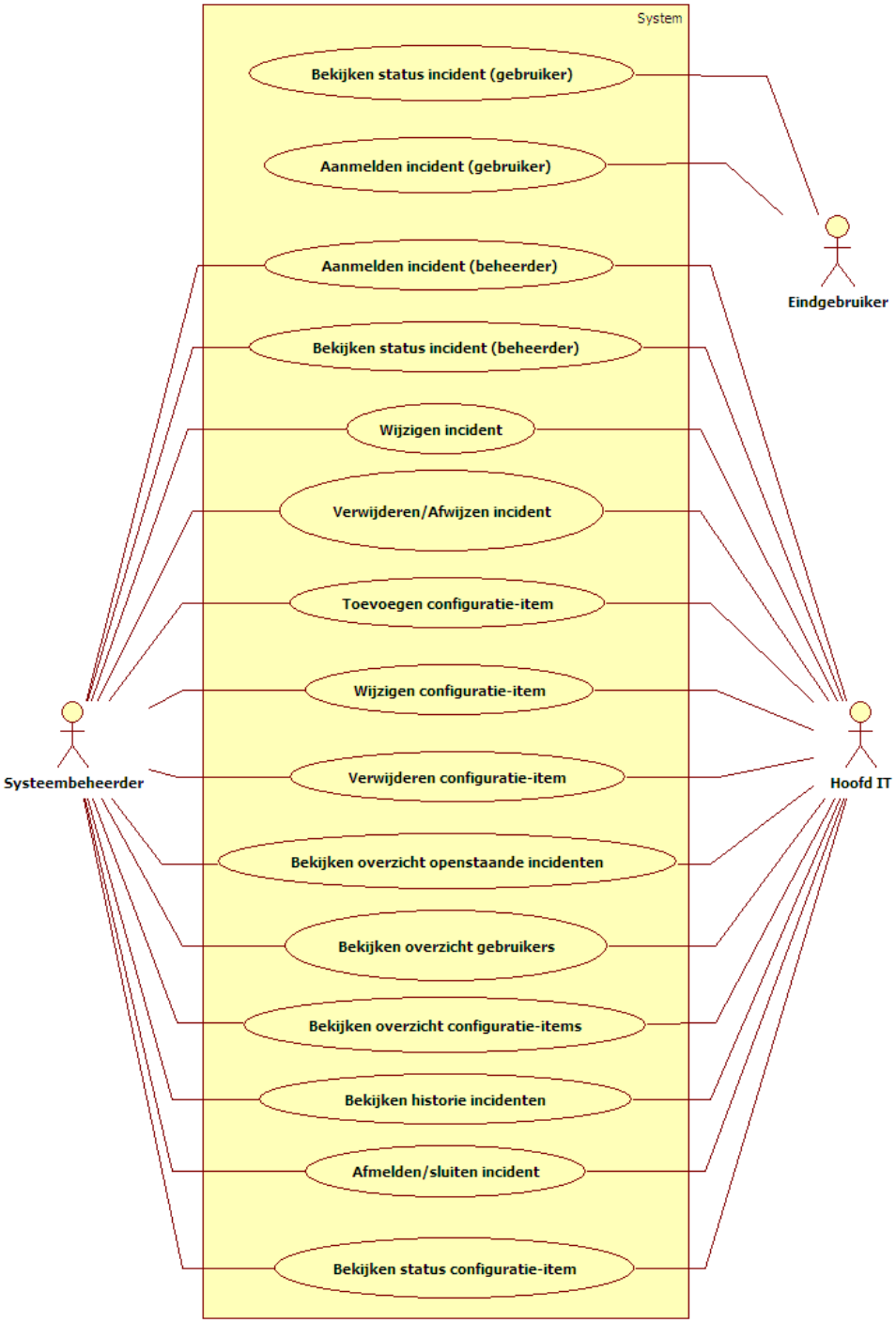
Ook was het voor dit project in principe niet de bedoeling dat er extra hardware werd aangeschaft. Bij de migratie van Crepa is al genoeg geld en tijd gemoeid aan het up-to-date brengen van de hardware-infrastructuur daar. De applicatie en alle ondersteunende software diende dus te draaien op bestaande hardware binnen de infrastructuur van Peinemann.

Licenties voor alle bovenstaande software zijn aanwezig, mochten er meer licenties nodig zijn, dan was het in overleg mogelijk om deze aan te schaffen, maar in principe werd ervan uitgegaan dat de aanwezige licenties voldoende waren.

De applicatie diende bovendien volledig aan te sluiten op de intranetsite van Peinemann, wat betekende dat de te ontwikkelen applicatie een webapplicatie moest zijn. Om een perfecte aansluiting te hebben op de site en op de geïnstalleerde software zou de applicatie ook in ASP.net geschreven moeten worden.

5.1.5 Use Case Model

Nu de stakeholders en de vision voor het nieuwe systeem duidelijk waren, was het zaak om een use case model te produceren waarin alle functionaliteiten, die voor de stakeholders belangrijk zijn, werden opgenomen. Het use case model vindt u in onderstaande afbeelding:



Figuur 7: Use case model

Na overleg met stakeholders bleek dat bovenstaande use cases voldoende waren voor de eerste iteratie en dat deze direct omgezet konden worden naar systeemfunctionaliteit.
Uit dit use case model zijn verschillende scenario's voortgekomen, beschreven in use cases.
Met de oplevering van het use case model kon de Inception fase worden afgesloten en hiermee was dus de Lifecycle Objective Milestone voor deze iteratie bereikt.

5.2 Elaboration Iteratie 1

De Elaboration fase heeft tot doel om de architectuur van de oplossing te bepalen en neer te zetten. Optioneel hierbij is het produceren van een "Architectural Proof of Concept". Om een duidelijk beeld te kunnen schetsen van de uiteindelijke applicatie die bij de heringerichte beheerorganisatie ging horen, is er dan ook een Architectural Proof of Concept aan de opdrachtgever opgeleverd.

Om het systeem op afstand bereikbaar te maken vanaf alle lokaties binnen de Peinemann bedrijven, is er in overleg met de opdrachtgever gekozen voor een webapplicatie als oplossing die de nieuwe beheerorganisatie gestalte moest geven.

Aangezien het hier gaat om een webapplicatie is er in deze fase ook een Navigation map opgeleverd om de schermstructuur aan de opdrachtgever duidelijk te maken.
Verder behoren use cases op specificatieniveau, sequentiediagrammen en klassendiagrammen tot deze fase. Van ieder van deze producten wordt tenminste één voorbeeld gegeven. Verdere informatie hierover is terug te vinden in de bijlagen.

5.2.1 Use cases

De uitwerkingen van de use cases uit het eerdergenoemde use case model (Inception fase) zijn ingepast in het template dat OpenUP hiervoor biedt. De bedoeling van een use case specificatie is, om aan te geven wat de stappen zijn die een actor in een bepaalde use case doorloopt bij zijn interactie met het systeem. Een use case heeft altijd een preconditie en een postconditie. Als voorbeeld is hier de use case “Aanmelden incident (beheerder)” gebruikt.

Use Case: Aanmelden incident (beheerder)

Omschrijving
Een storing of probleem aan een configuratie-item treedt op en er wordt een incident aangemaakt door de systeembeheerder of het hoofd IT.

Actor(en)
Deze usecase geldt voor de systeembeheerder en het hoofd IT. De actor wordt in de eventflow neergezet als “gebruiker”.

Precondities
Het PC werkstation is gestart en ingelogd, de hoofdpagina van het intranet is geopend.

Basic Flow of Events
1. De gebruiker klikt met de muis op “Incident melden”.
2. Het formulier “Incident melden” wordt getoond op het scherm.
3. De gebruiker vult het getoonde formulier in: Gebruikersnaam waarvoor het incident wordt aangemeld, type apparaat (indien toepasbaar), naam van het apparaat (indien toepasbaar), onderwerp en omschrijving van de storing, prioriteit en geeft door middel van een checkbox aan of dit incident in een eventuele latere rapportage dient te worden meegenomen.
4. De gebruiker klikt op “Incident aanmelden”.
5. De pagina met openstaande incidenten wordt getoond, met daarin (onder andere) de gegevens van het zojuist aangemaakte incident.

Postcondities
Een incidentmelding is succesvol aangemaakt.

Specificatie Requirements
Het moet mogelijk zijn om incidenten te melden via de applicatie, die geïntegreerd moet zijn in het bestaande intranet.

Figuur 8: Voorbeeld use case

In de latere Construction en Transition Fases waren deze use cases (en de manier waarop zij ingedeeld zijn) ook zeer bruikbaar als test cases voor de acceptatietest.

5.2.2 Conceptueel datamodel

Om een helder beeld te krijgen van de verschillende klassen binnen de te produceren applicatie moesten er verschillende soorten klassendiagrammen worden opgesteld.

Deze klassendiagrammen vielen in de loop van het project onder verschillende noemers, van Conceptueel datamodel en Domeinklassendiagram tot het Implementatieklassendiagram.

Het conceptueel datamodel is vooral bedoeld om klassen te definiëren met hun bijbehorende relaties en multipliciteit. Speciale relaties zoals associatieklassen worden hierin niet opgenomen, evenmin als attributen en operaties.

Bij de totstandkoming van het conceptueel datamodel moest vooral nagedacht worden over de objecten die hierin zouden worden opgenomen als klassen. Er kon hierbij worden uitgegaan van de objecten die golden als configuratie-item en de objecten waaraan deze configuratie-items gekoppeld waren of zouden moeten zijn.

De verschillende configuratie-items waarover beheer moest plaatsvinden, zijn:

- Pc
- Printer
- Server
- Gsm

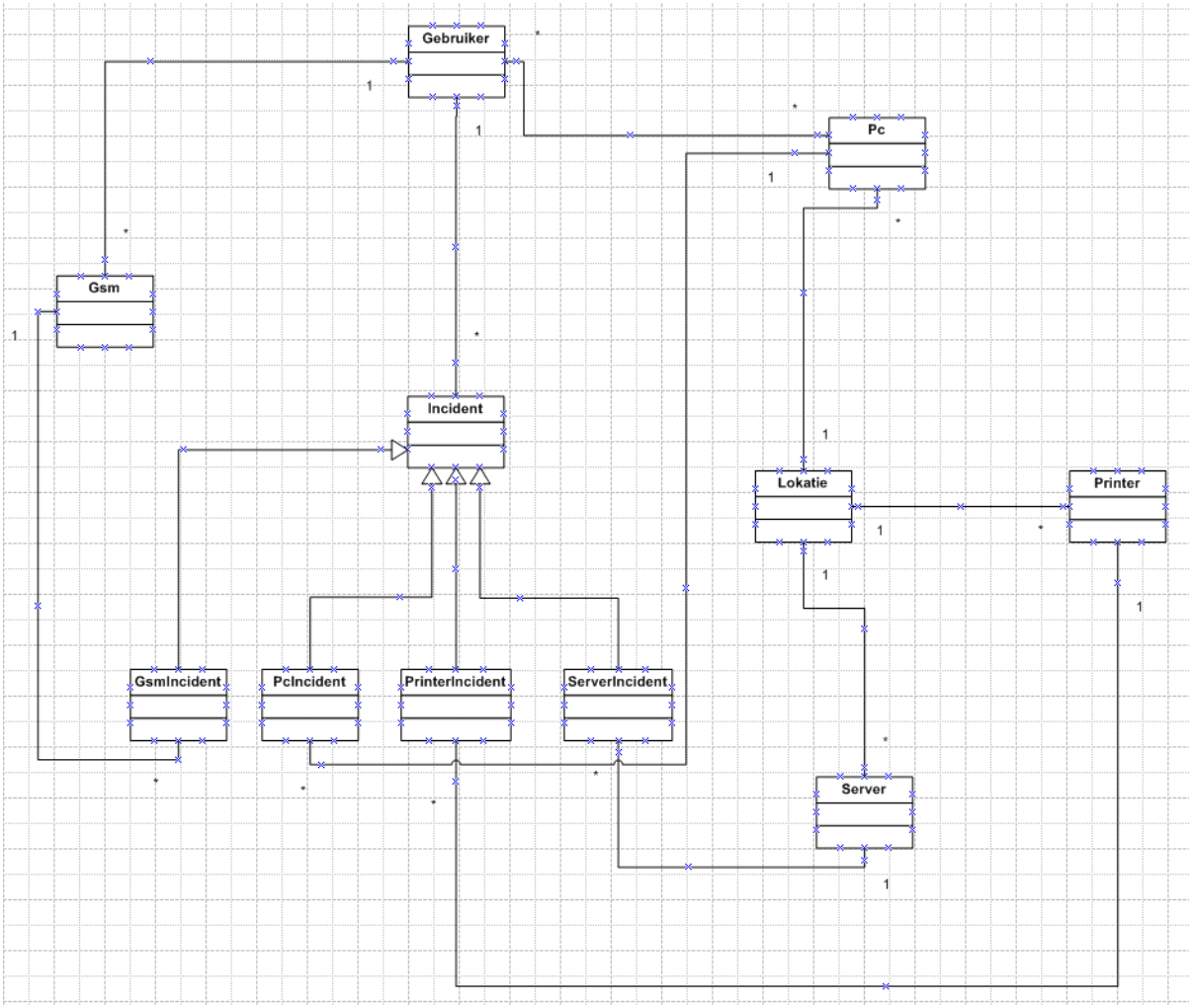
Verder vond beheer plaats van gebruikers en locaties, welke gekoppeld moesten zijn aan de verschillende configuratie-items:

- Locatie
- Gebruiker

Binnen het onderdeel incidentbeheer was het de bedoeling dat er onderscheid gemaakt zou worden tussen verschillende types incidenten, in verband met het rekening houden met de toekomstige situatie waarin rapportages vanuit het nieuwe systeem werden opgeleverd, waarbij de incidenten in verschillende types ingedeeld zouden zijn. Vandaar de verschillende subklassen van “Incident”:

- Incident
- Pc-incident
- Gsm-incident
- Printerincident
- Serverincident

In onderstaand figuur vindt u het conceptueel datamodel ter illustratie.



Figuur 9: Conceptueel datamodel

Zoals uit het model duidelijk wordt, heeft een printerincident of PC-incident (enzovoorts) altijd dezelfde eigenschappen als de klasse incident. Een dergelijke subklasse van incident kan niet bestaan zonder de superklasse incident. Er kan dus bijvoorbeeld geen Pc-incident bestaan zonder dat deze gekoppeld is aan een incident.

Een incident is altijd gekoppeld aan een gebruiker, in het algemeen de gebruiker die het incident aanmeldt.

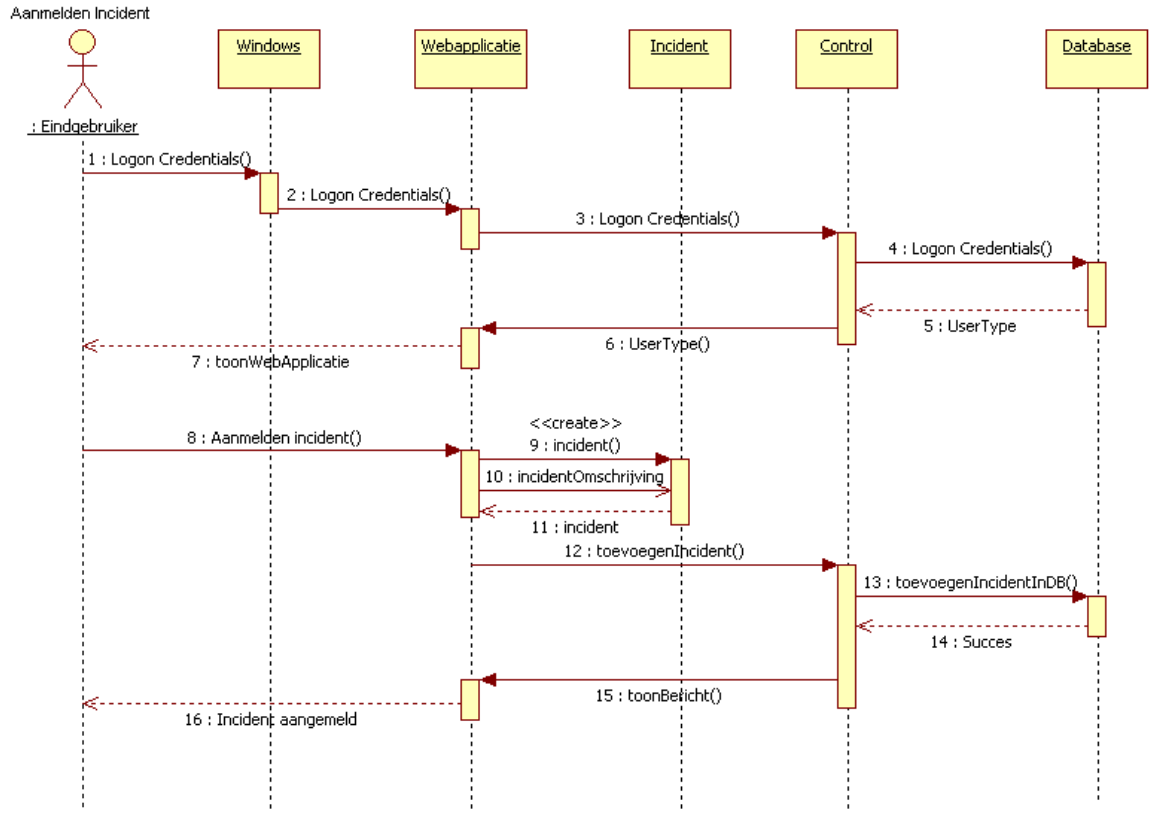
Servers, printers en pc's zijn gekoppeld aan een locatie. Gsm's en Pc's zijn gekoppeld aan een gebruiker, waarbij één gebruiker meerdere pc's en gsm's kan hebben. Een pc kan tevens bij meerdere gebruikers horen, terwijl een gsm maar bij één gebruiker kan horen.

Dit bleek een correcte weergave te zijn van de werkelijkheid binnen de organisatie en zou dus zonder bedrijfstechnische problemen in een applicatie verwerkt kunnen worden.

5.2.3 Sequentiediagrammen

Bij het omschrijven van een use case (zoals in 5.2.1) is het lastig om een overzicht te krijgen van de interactie tussen de verschillende systeemonderdelen.
Om een duidelijker beeld te scheppen van de interactie tussen gebruiker, systeem en de verschillende onderdelen hiervan, kunnen use cases schematisch weergegeven worden in sequentiediagrammen. Hierdoor wordt ook een verdere verdieping bereikt in de use cases.

Onderstaand wordt het sequentiediagram van de use case “Aanmelden incident (Eindgebruiker)” getoond, deze is in feite hetzelfde (met uitzondering van enkele invulvelden) als het met “beheerder” corresponderende sequentiediagram. De overige sequentiediagrammen zijn opgenomen in de bijlagen.



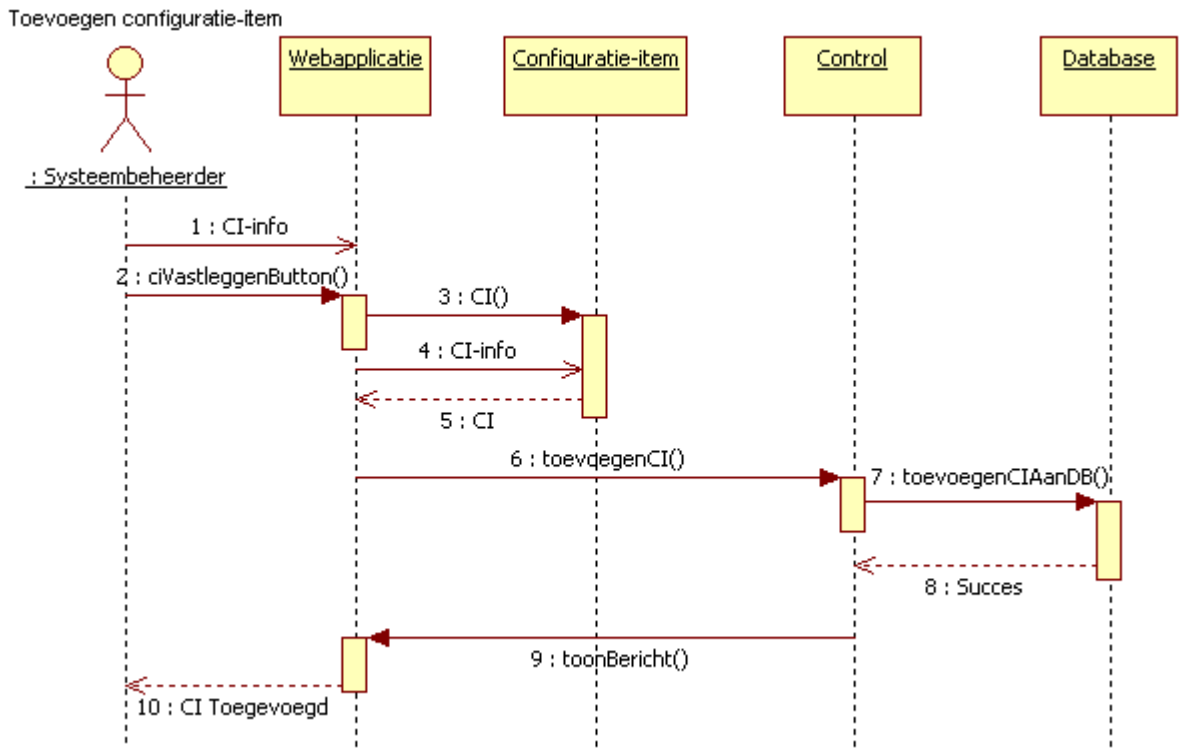
Figuur 10: Sequentiediagram “Aanmelden incident”

Bij het lezen van de sequentiediagrammen dient verder het volgende in acht genomen te worden:

- De applicatie gaat uit van de Logon credentials van Windows bij iedere actie die een gebruiker doorvoert in de database. Dit is alleen opgenomen in het sequentiediagram “Aanmelden incident”. Voor de rest van de Use Cases (en dus sequentiediagrammen) is dit in feite hetzelfde, dit is hier echter omwille van efficiëntie niet in de diagrammen opgenomen.

- In de sequentiediagrammen is sprake van CI. Dit staat voor configuratie-item. Wanneer er in het diagram sprake is van een configuratie-item, dan kan dit één van de volgende items betekenen:
    - Gebruiker
    - PC
    - Server
    - Printer
    - GSM
- Het uiteindelijke bericht dat aan het einde van een sequentie wordt getoond aan de gebruiker, is in feite geen softwarematige return message, maar een visuele aanwijzing voor de gebruiker dat de uitgevoerde actie geslaagd is.
- “Windows” als besturingssysteem is opgenomen in het sequentiediagram, maar niet als klasse in de klassendiagrammen omdat het besturingssysteem standaard op alle machines binnen het bedrijf aanwezig is. De afhandeling van logon credentials wordt na de eerste aanmelding van de gebruiker volledig afgehandeld door Windows, Active Directory, ASP.NET en IIS. De applicatie controleert echter bij iedere actie van de gebruiker, of deze een bij het besturingssysteem (en dus aan het domein) geauthenticeerde gebruiker is.
- “Control” is hier opgenomen ter voorbereiding van de implementatie van een MVC en/of n-Tier architectuur. Voor deze optie is hier gekozen om de klassen die belangrijke gegevens (zoals methoden waarmee databasetoegang bereikt wordt) bevatten af te schermen van de eindgebruiker.

Bovenstaande geldt ook voor onderstaand sequentiediagram, dat betrekking heeft tot het toevoegen van een configuratie-item door de systeembeheerder of het hoofd IT:



Figuur 11: Sequentiediagram “Toevoegen configuratie-item”

5.2.4 Architectuur

De volgende stap in het ontwikkelproces was het bedenken van een architectuur die het beste past bij de situatie (alhoewel het conceptueel datamodel alvast bepaalde constraints legt aan een te ontwikkelen structuur en hier in feite de basis van is).

Het was vooral belangrijk om een keuze te maken tussen de verschillende mogelijke programmeertalen om de applicatie mee te gaan schrijven. Tijdens de opleiding was vooral de nadruk gelegd op Java (java beans, jsp, servlets, verschillende databasepakketten zoals Oracle en Sybase) en later ook op ASP.net 1.1 in combinatie met C#, web services en SQL server. Omdat er niet genoeg tijd beschikbaar was om te beginnen aan het eigen maken van een volledig nieuwe taal, moest er dus gekozen worden tussen java en ASP.net met C#.

Binnen het bedrijf wordt gewerkt met een volledig op Microsoft producten gerichte infrastructuur op basis van Windows 2000 en 2003 servers met Active Directory. Ondersteunend hierbij zijn Exchange en op applicatiegebied Microsoft SQL Server en Microsoft IIS. Voor de ontwikkeling van het intranet was er reeds een licentie voor Microsoft Visual Studio 2005 aanwezig.

Omwille van de integratie in het intranet is er daarom ook voor gekozen om de applicatie te schrijven met behulp van ASP.net en C#. Wel werd hierbij de stap gemaakt om gebruik te gaan maken van ASP.net 2.0 ten opzichte van het bij de afstudeerder reeds bekende ASP.net 1.1, vooral door de ondersteuning van ASP.net 2.0 voor zogenaamde “Master Pages”. Bij het gebruik van een master page hoeft de volledige “look” van een webpagina niet op ieder webform toegepast te worden. Een eventuele wijziging in uiterlijk hoeft daarom maar op één lokatie plaats te vinden, de daadwerkelijke inhoud van de pagina wordt binnen deze master page ingeladen. Om te voldoen aan de eis van gebruiksvriendelijkheid en inpassing in de huisstijl, is er voor gekozen om de CSS (Cascading Style Sheet) van de intranetsite te gebruiken en waar nodig aan te passen. Zo werd bereikt dat de grens tussen applicatie en intranetsite voor de gebruiker dusdanig vervaagd werd, dat men ging spreken over “het intranet” als geheel in plaats van de verschillende individuele gekoppelde applicaties. Aangezien de filosofie van Peinemann is, dat in de toekomst liefst alle applicaties door middel van de intranet “portal” te starten zijn en zo veel mogelijk in de huisstijl passen, is dit een stap in de goede richting.

In feite zorgt de combinatie van SQL Server, ASP.net en IIS al voor bepaalde architectuurvormen die min of meer verplicht zijn. Zo is het “common practice” om bij het ontwikkelen van een ASP.net webapplicatie te werken met zogenaamde “code-behind files” waarin de business logic van de applicatie staat. Op de achtergrond is een database aanwezig (die aangesproken wordt door middel van stored procedures), en op de voorgrond is een “.aspx” bestand aanwezig, gevuld met XHTML code, dat zorgt voor de interactie met de gebruiker door middel van het tonen van een webpagina in de browser. In feite wordt hiermee een zogenaamde “Three Tier Architecture” bereikt. In het geval van de opgeleverde applicatie kan gesproken worden van een “n-Tier Architecture”, aangezien er naast de business logic in de code-behind files ook gebruik werd gemaakt van verschillende klassen (behorend bij de verschillende objecten in de klassendiagrammen) waarin business logic specifiek voor die klassen beschikbaar werd gesteld. Omdat alle databasemethodes werden opgenomen in een losse klasse, kan deze klasse zowel gezien worden als onderdeel van de Tier “Application logic” als onderdeel van het “Data Model”. Het gebruik van stored procedures zorgde ervoor dat ook aan de kant van de database een onderdeel van het datamodel werd opgenomen.

Hieronder wordt nog kort aangegeven, welke architectuurvormen toepasbaar zijn op de applicatie en waarom hiervoor is gekozen.

- Client-server. Dit is de essentie van iedere webapplicatie, de client heeft alleen een web browser nodig, alle content wordt aangeleverd vanaf de server. Zonder deze vorm van architectuur is het zo goed als onmogelijk om een webapplicatie neer te zetten die altijd en van iedere lokatie bereikbaar is.
- Database-centric. Het datamodel en de manier waarop toegang tot gegevens plaatsvindt is volledig gericht op interactie van de applicatie met een database. Sturing van deze communicatie vindt op database-engine niveau plaats door middel van stored procedures. Hiervoor is gekozen omdat zo snel wijzigingen in interactie met de database kunnen worden doorgevoerd, zonder aanpassingen te maken aan de applicatie zelf. Verder is de database een snelle en betrouwbare vorm van gegevensopslag, die centraal bereikbaar is voor toevoegingen, wijzigingen en het verwijderen van gegevens.
- Implicit invocation. Door middel van postbacks naar de applicatieserver worden impliciet methodes aangeroepen. Dit is in feite een event-driven architectuur. Hier is niet zo zeer voor gekozen, dit is een manier van werken die geïntegreerd is in het ontwikkelen met ASP.net en webapplicaties in het algemeen.
- Door het gebruik maken van verschillende klassen, ingericht naar objecten wordt CBSE bereikt (Component Based Software Engineering). Om een relatief eenvoudige uitbreidbaarheid van de software te realiseren, is gekozen voor Component Based Software Engineering. In het geval dat er bijvoorbeeld een bepaald type configuratie-item bij zou komen, volstaat het om alleen de methodes aan te maken die bij deze nieuwe “klasse” horen en daarbij de methodes die bij de reeds aanwezige klassen horen ongemoeid te laten. Daardoor verliest de applicatie geen bestaande functionaliteit ten koste van nieuwe functionaliteit, tenzij daarvoor impliciet wordt gekozen.
- MVC (Model, View, Controller). De view stuurt gegevens naar de controller, de controller stuurt gegevens naar het model, de view krijgt gegevens direct terug van het model OF de gegevens doorstaan een validatie door de controller. Hiervoor is gekozen omdat zo duidelijk afgebakend wordt, wat het doel is van bepaalde onderdelen van de applicatie. Bij MVC is in feite sprake van het direct terugsturen van gegevens van het model naar de view. Binnen de applicatie is dit echter niet altijd het geval, daarom wordt maar gedeeltelijk voldaan aan het MVC-model. Deels voldoet de applicatie dus aan MVC en deels aan de “Multi-tier” architectuur.
- Multi-tier (of n-tier) architectuur.
    - Presentation Tier (de webpagina en formulieren die de gebruiker ziet, webserver frontend, static content).
    - Application Tier/Business Logic Tier (Content Processing & Generation, in dit geval de ASP.net codebehind files en losse klassen alsmede ook web services).
    - Data Tier (Backend Database, datasets, stored procedures, SQL Server DBMS).

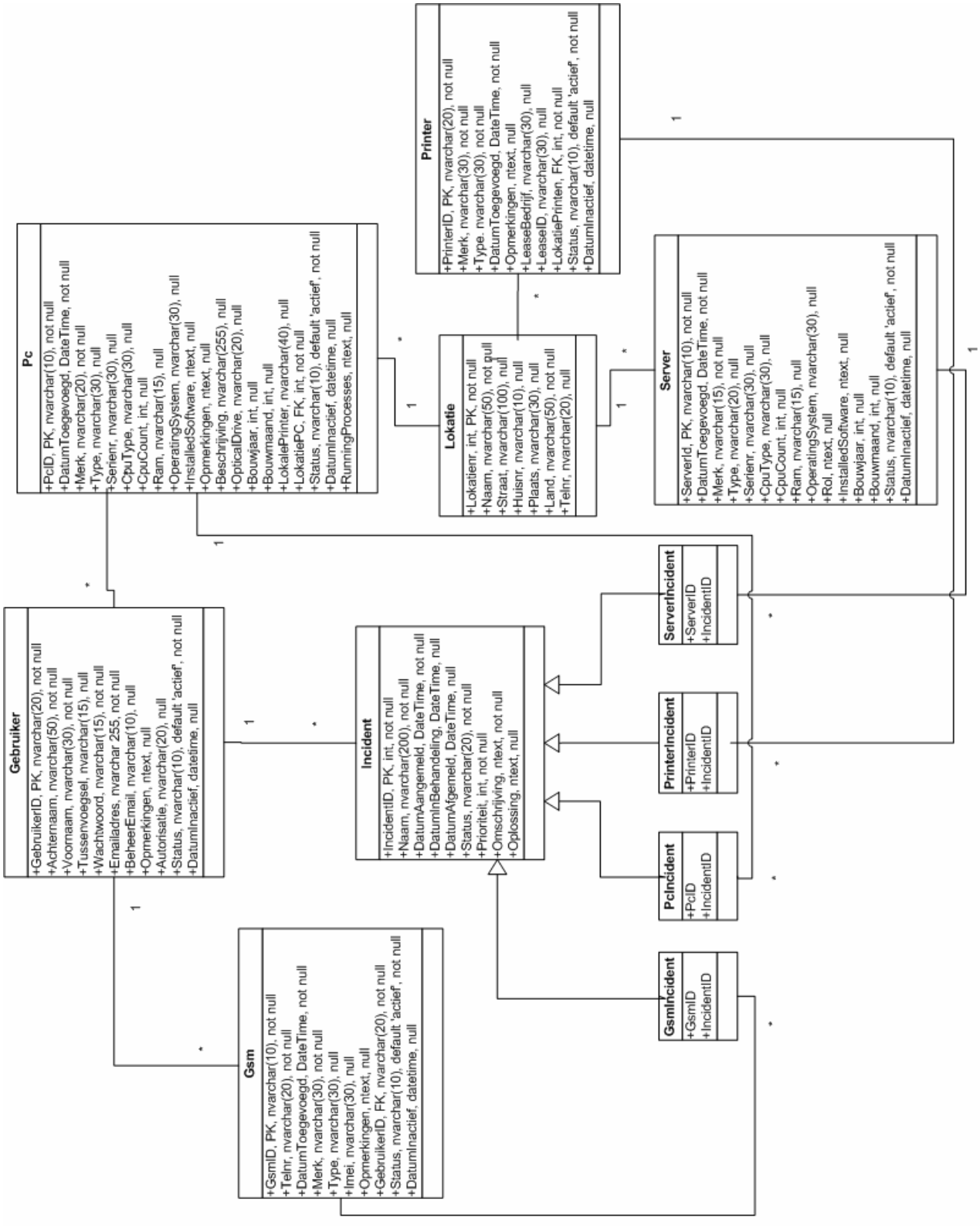
  Bij deze architectuur kan de Data Tier niet direct gegevens uitwisselen met de Presentation Tier. Dit verloopt ten allen tijde via de Application Tier. In de applicatie is omwille van “Rapid Development” echter gekozen om bepaalde stored procedures (uit de database en dus uit de Data Tier) direct aan te roepen vanaf het webformulier (Presentation Tier). Het ging hier om het ophalen van relatief korte gegevensstromen (bijvoorbeeld voor het vullen van een DropDownList). Dit kon eenvoudig bereikt worden door het implementeren van een SqlDataSource. Omdat de gegevens in een correct formaat worden aangeleverd en dusdanig door ASP.net kunnen worden geïnterpreteerd, werd het niet nodig geacht om hierbij de tussenstap via de Application Tier te nemen. Dit zou de applicatie onnodig ingewikkeld maken en bovendien zorgen voor het gevreesde “code bloat”, wat in feite staat voor het gebruik maken van teveel code, die bovendien geen extra functionaliteit oplevert voor de applicatie.

Om de opdrachtgever een idee te geven van hoe deze architectuur er uit zou moeten zien, is er een “Architectural Proof of Concept” opgeleverd. Deze kon gemaakt worden, nadat het domeinklassendiagram met behulp van de vastgelegde architectuur kon worden opgesteld.

5.2.5 Domeinklassendiagram

Het domeinklassendiagram is een verdere uitwerking van het conceptueel datamodel. Het verschil met het conceptueel datamodel is ten eerste dat het domeinklassendiagram veel meer toegespitst is op attributen die benodigd zijn. Dit toont aan dat het domeinklassendiagram de conceptfase voorbij is, aangezien er al uitgebreid nagedacht is over welke informatie het systeem moet bevatten. Met behulp van het domeinklassendiagram kan een groot deel van de structuur van de database al worden aangemaakt. Gedurende het ontwikkeltraject werd het domeinklassendiagram constant bijgewerkt in het geval dat er nieuwe relaties ontstonden of er nieuwe attributen bijkwamen. De voorbereiding op de requirements van interatie 2, waaraan pas later zou worden voldaan, is reeds aanwezig in dit diagram. Dit is terug te zien in de attributen van de klasse “Pc”.

Op de volgende pagina vindt u een deel van het domeinklassendiagram, volledige modellen zijn te vinden in de bijlagen.



Figuur 12: Domeinklassendiagram

5.2.6 Inrichting database

Zoals bovenstaand vermeld, was met de komst van het domeinklassendiagram de mogelijkheid ontstaan om de database te gaan inrichten. Dit is geen verplichte stap binnen het ontwikkeltraject, maar werd door de opdrachtgever en afstudeerder gezien als een goede praktijktest van de in de modellen beschreven architectuur. Belangrijk onderdeel van het inrichten van de database is het opstellen van relationele implementatiemodellen. Met behulp van deze modellen kan de database met de verschillende tabellen aangemaakt worden. Om dit te illustreren wordt onderstaand een voorbeeld gegeven van een relationeel implementatiemodel. De overige modellen zijn allen te vinden in de bijlagen.

```
CREATE TABLE [dbo].[tblPC]

      [PcID] [nvarchar](10) NOT NULL,
      [DatumToegevoegd] [datetime] NOT NULL,
      [Merk] [nvarchar](20) NOT NULL,
      [Type] [nvarchar](30) NULL,
      [Serienr] [nvarchar](30) NULL,
      [CpuType] [nvarchar](30) NULL,
      [CpuCount] [int] NULL,
      [Ram] [nvarchar](15) NULL,
      [OperatingSystem] [nvarchar](30) NULL,
      [InstalledSoftware] [ntext] NULL,
      [LokatiePC] [int] NOT NULL,
      [Opmerkingen] [ntext] NULL,
      [Beschrijving] [nvarchar](255) NULL,
      [TypeBeveiliging] [nvarchar](30) NULL,
      [OpticalDrive] [nvarchar](20) NULL,
      [Bouwjaar] [int] NULL,
      [Bouwmaand] [int] NULL,
      [LokalePrinter] [nvarchar](40) NULL,

PRIMARY KEY [PcID]
FOREIGN KEY [LokatiePC] REFERENCES [dbo].[tblLokatie] ([Lokatienr])
ON UPDATE CASCADE
```

In feite is de database door middel van de gebruikersinterface ingericht, maar is het opstellen van dergelijke implementatiemodellen een goed hulpmiddel bij het begrijpen van de structuur van de database. Niet alle implementatiemodellen zijn met de hand gemaakt, de meeste zijn door middel van reverse engineering vanuit de database aangemaakt. Dit heeft als voordeel, dat bij een catastrofaal falen van de database en het ontbreken van back-ups, de gehele database met behulp van de relationele implementatiemodellen zeer snel weer in de lucht gebracht kan worden.
Dit bleek bovendien een snellere werkwijze die de mogelijkheid bood om de aangemaakte tabellen te controleren op consistentie.

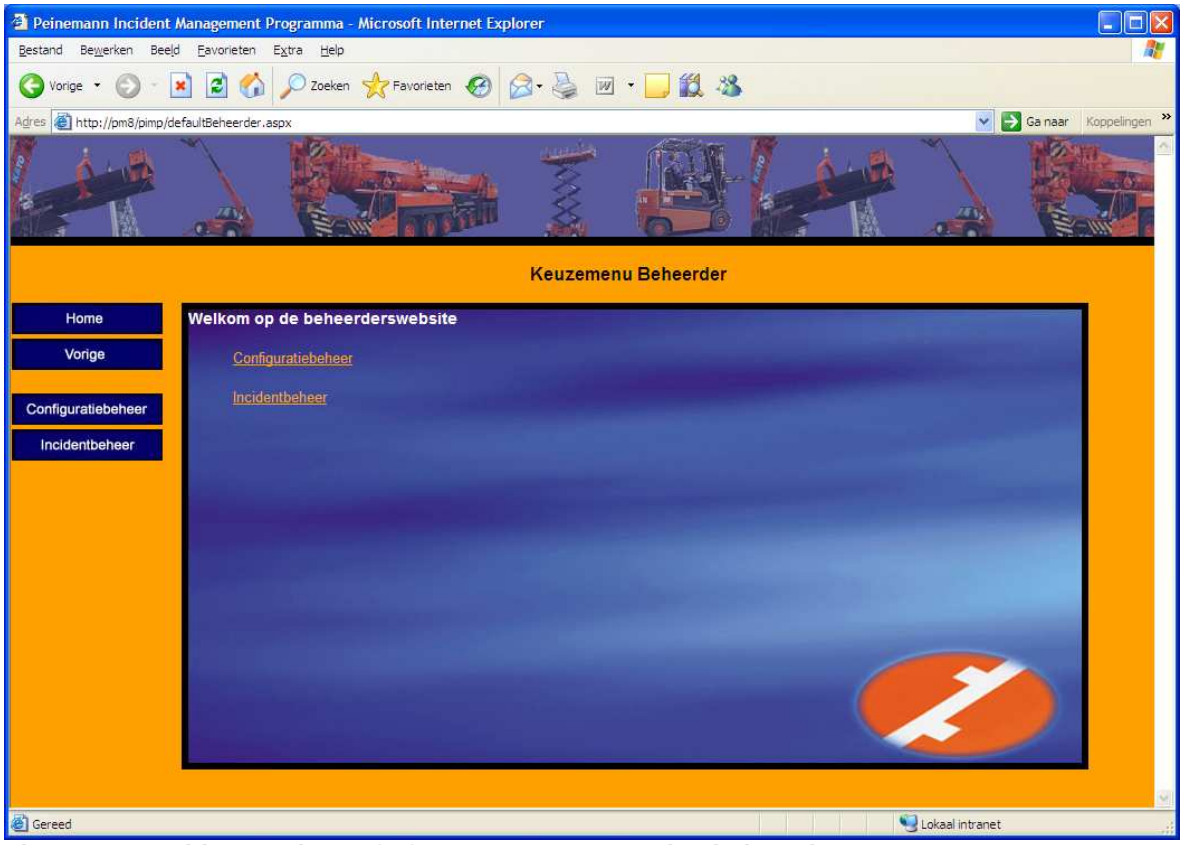
Het uiteindelijke resultaat en doel van deze stap in het ontwikkeltraject was, dat de basis van de databasestructuur aanwezig was, zodat begonnen kon worden aan het “Architectural Proof of Concept”. Dit onderdeel van het ontwikkeltraject moest uiteraard voldoen aan de vooraf bepaalde architectuurkenmerken. Hiervoor was in ieder geval nodig, een Database server met database (SQL Server 2000 + database, data tier), een applicatieserver voor .net applicaties (ASP.net + IIS, application tier) en een manier van presenteren van de gegevens aan de gebruiker (IIS + webbrowser, presentation tier). Met de voorlopig complete inrichting van de database was dit doel bereikt.

5.2.7 Architectural Proof of Concept

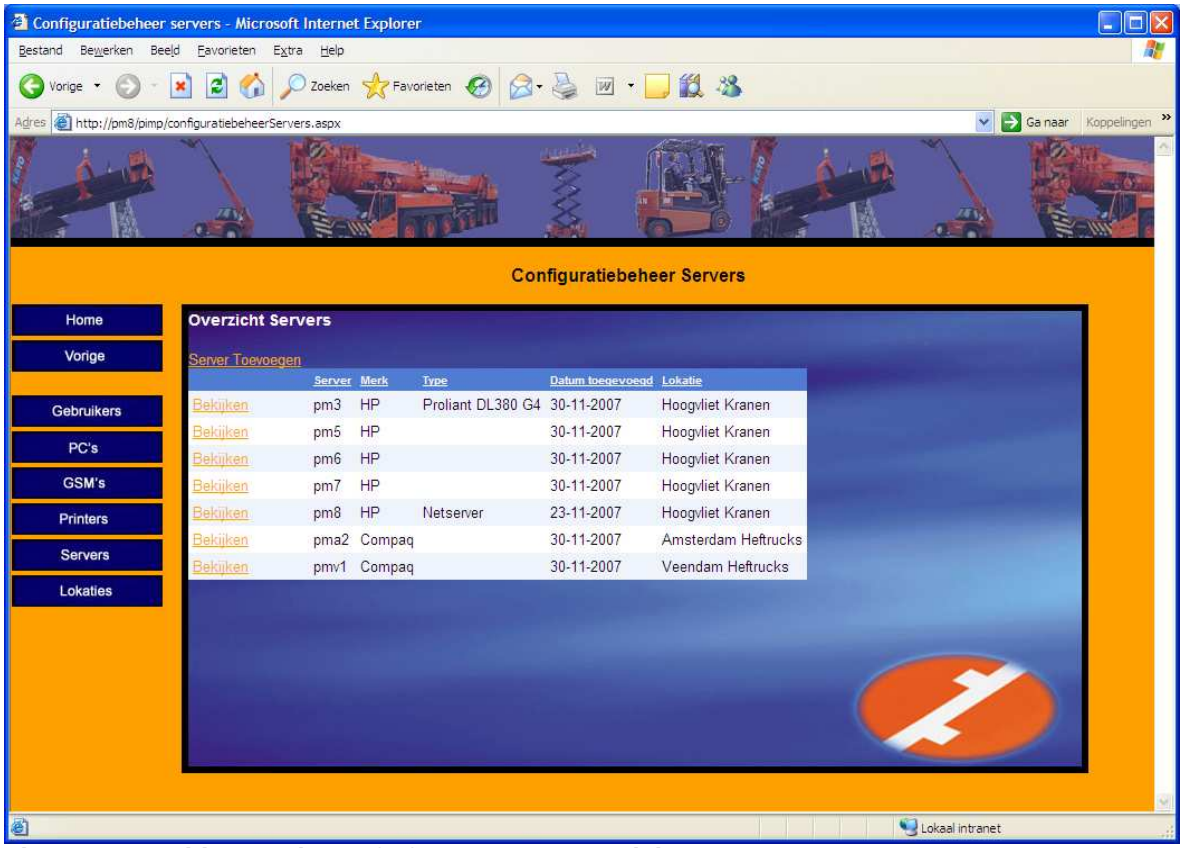
Nu er een solide basis voor de neer te zetten architectuur aanwezig was, kon het nut en de functionaliteit van deze architectuur bewezen worden door middel van het ontwikkelen van het “Architectural Proof of Concept”. De keuze voor het ontwikkelen van een proof of concept in de vorm van een soort prototype werd gemaakt om meerdere redenen:

- Levensvatbaarheid van de gekozen architectuur verifiëren: Is de architectuur toepasbaar op de praktijksituatie? Moeten er nog wijzigingen plaatsvinden zodat de applicatie niet per se hoeft te voldoen aan alle architectuurelementen, zelfs als deze voor het voldoen aan de requirements niet nodig zijn?
- Tonen aan de opdrachtgever en testgebruikers hoe de applicatie er ongeveer uit gaat zien.
- Controle of de gekozen producten/hulpmiddelen wel de juiste functionaliteit boden (ASP.net, IIS, SQL Server).
- Haalbaarheid bekijken van het implementeren van alle geëiste en gewenste functionaliteit.
- Implementatie van bestaande Cascading Style Sheet, wat meteen een test was voor de hieraan gemaakte wijzigingen (zoals verschillen in lettergrootte, de lengte- en breedtematen van de verschillende schermsecties, kleuren van knoppen en links, locatie van koppelingen).

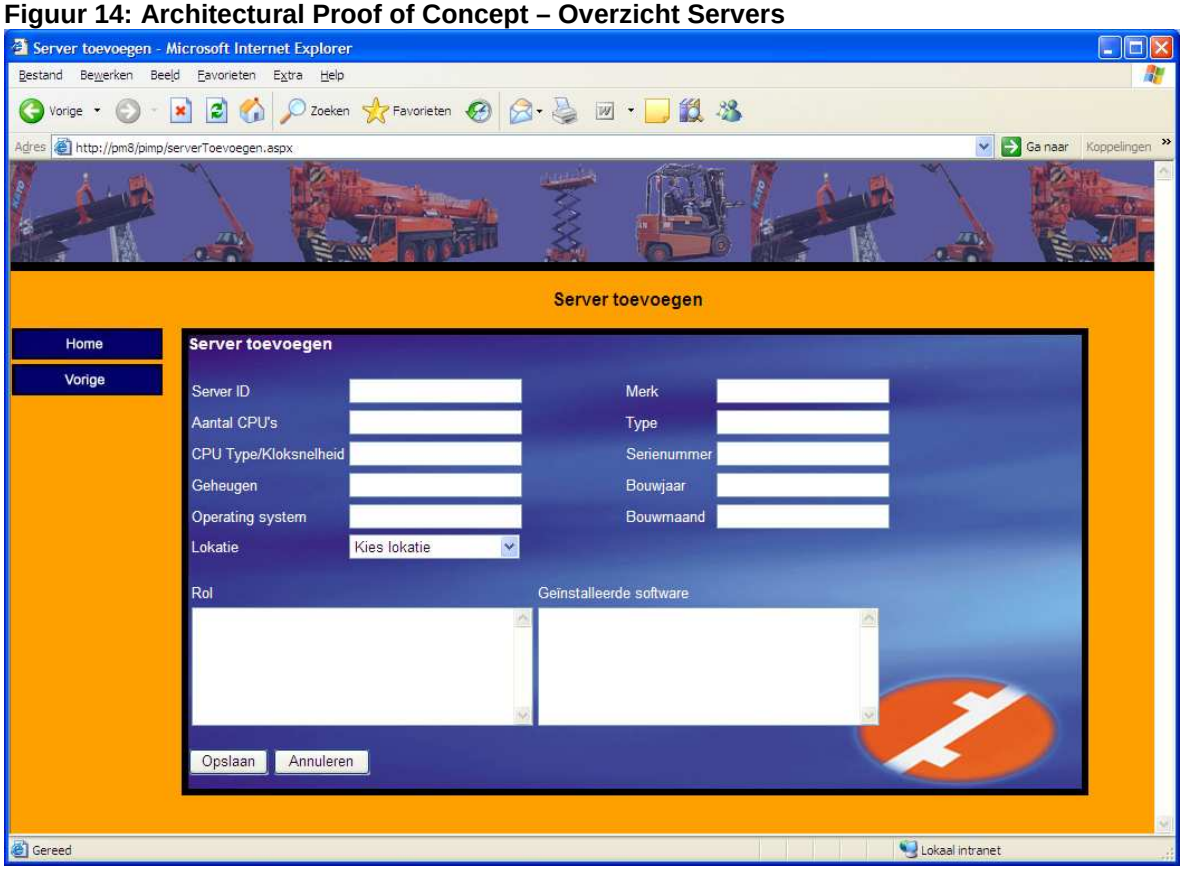
Meerdere screenshots van het “Architectural Proof of Concept” zijn te vinden in de bijlagen, hieronder worden er slechts enkele getoond ter illustratie.



Figuur 13: Architectural Proof of Concept – Startpagina beheerder



Figuur 14: Architectural Proof of Concept – Overzicht Servers



Figuur 15: Architectural Proof of Concept – Server toevoegen

De functionaliteit van het proof of concept beperkte zich tot het toevoegen, wijzigen en verwijderen van één bepaald type configuratie-item. Hiermee werd bij wijze van spreken een tipje van de sluier opgelicht voor wat betreft de toekomstige werking van de applicatie. Het belangrijkste was echter, dat met behulp van het proof of concept duidelijk werd dat de gekozen architectuur de juiste is geweest. Het bewijzen hiervan (of mogelijk van het tegengestelde) is in feite het hoofddoel van het Architectural Proof of Concept.

5.2.8 Implementatieklassendiagram

Met het succesvolle Architectural Proof of Concept bleek dat de gekozen architectuur de juiste was. Op deze manier werden de in het kader van het ontwikkeltraject geproduceerde modellen dus ook meteen gevalideerd. Er bleek snel en gemakkelijk met de applicatie te werken, het bewerken van functionaliteit in alle applicatielagen bleek relatief eenvoudig. Op deze manier kon met vertrouwen een “final” product worden opgeleverd voor deze iteratiefase, namelijk het implementatieklassendiagram. Dit diagram is in feite een compleet klassendiagram, inclusief associatieklassen, relaties, multipliciteit, attributen en operaties. Op basis van dit diagram kon de complete applicatie worden neergezet. Uiteraard werd dit model bij eventuele wijzigingen volledig actueel gehouden. Op de volgende pagina wordt dit diagram getoond.

Met het opleveren van een implementatieklassendiagram wordt normaal gesproken de Elaboration fase afgesloten. Het klassendiagram is op deze manier geheel voltooid voor deze iteratie. Belangrijk hierbij is de associatieklasse die is opgenomen bij de veel-op-veel-relatie tussen “Gebruiker” en “Pc”.

Figuur 16: Implementatieklassendiagram

Met het opleveren van het implementatieklassendiagram werd weer een mijlpaal bereikt, namelijk de afsluiting van de Elaboration fase van deze iteratie. In UP-termen heet dit de Lifetime Architecture Milestone, in dit geval vond de afstudeerder het echter nuttig om in de Elaboration fase nog een Navigation Map toe te voegen, die het bouwen van de navigatiestructuur van de webapplicatie zou vergemakkelijken.

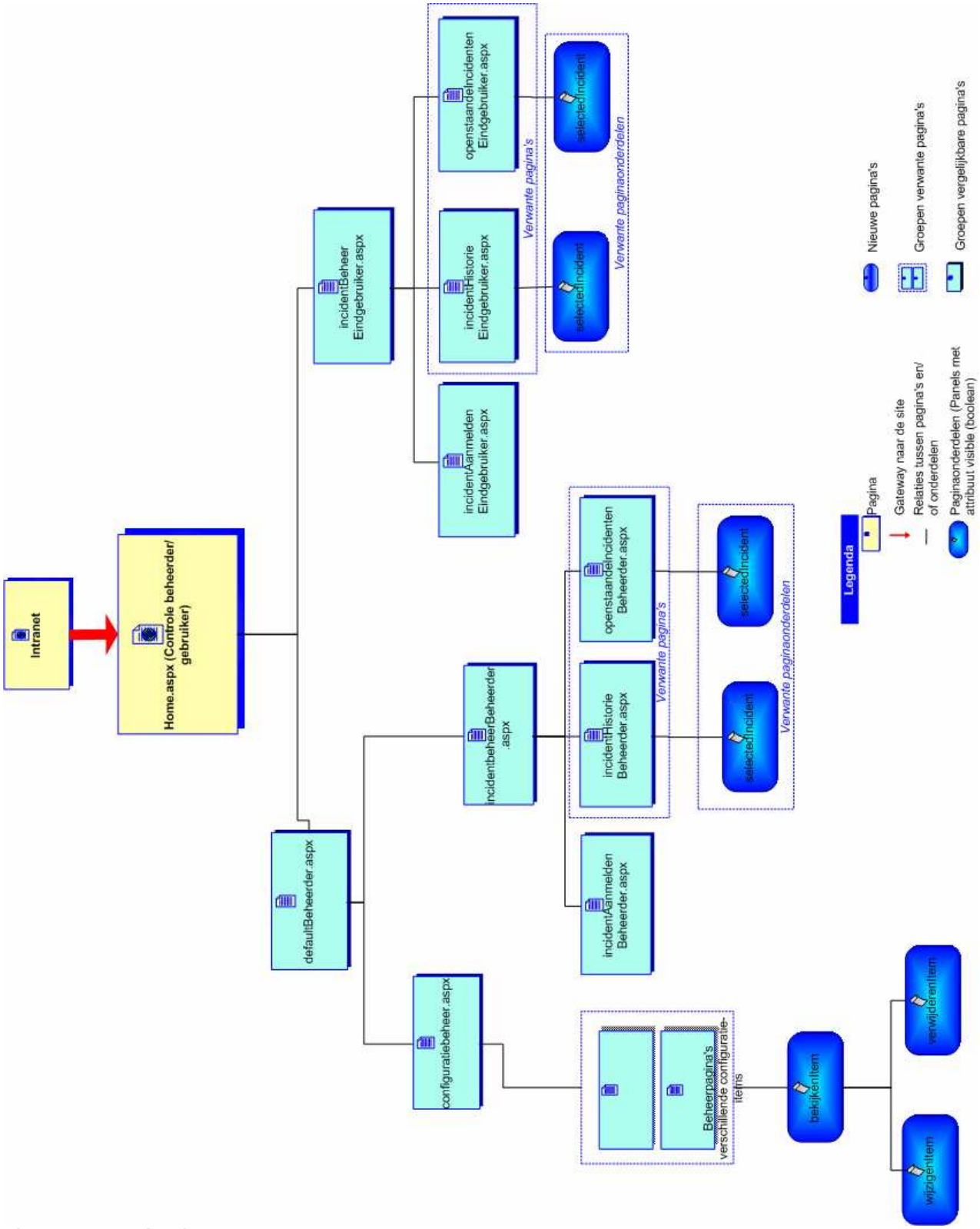
5.2.9 Navigation Map

Om voor de opdrachtgever te verduidelijken welke onderdelen de applicatie zou bevatten en waar deze te vinden zijn, werd er door de afstudeerder voor gekozen om een “Navigation Map” samen te stellen.

De navigation map is ook bij het programmeren van de applicatie (en dan vooral bij het maken van koppelingen tussen verschillende pagina’s) een bruikbaar hulpmiddel gebleken. Met het volgen van de koppelingen tussen pagina’s in de navigation map kon tijdens het ontwikkelen van de applicatie het overzicht behouden worden, zodat bepaalde wijzigingen op de juiste pagina’s werden doorgevoerd.

De navigation map is geen verplicht onderdeel van een UP-ontwikkeltraject, maar kan dus in het geval van het ontwikkelen van een webapplicatie zeer nuttig zijn.

Op de volgende pagina vindt u de navigation map afgebeeld.



Figuur 17: Navigation Map

5.3 Construction Iteratie 1

Na de succesvolle afsluiting van de Elaboration fase kon er overgegaan worden naar de Construction fase. Belangrijk deel van deze fase is het daadwerkelijke programmeren van de applicatie. Vervolgens dient in deze fase begonnen te worden aan de voorbereiding van tests. Unit tests vonden gedurende de Construction fase constant plaats om de kwaliteit van de zich ontwikkelende applicatie te waarborgen. Het voltooien van de fase levert de Initial Operational Capability Milestone op. Tijdens deze fase is niet zoveel documentatie opgeleverd als in de Elaboration fase, het grootste deel van de tijd ging op aan het bouwen van de applicatie die PIMP werd genoemd, het Peinemann Incident Management Programma.

5.3.1 Overzetten gegevens uit oude database

In de bestaande configuratiebeheerapplicatie waren al veel gegevens over gebruikers en pc's aanwezig. De informatie hierin was volgens de systeembeheerder correct en min of meer up to date. Daarom is er in overleg met hem en met het hoofd IT door de afstudeerder besloten om de gegevens over te nemen vanuit de oude database.

Dit was mogelijk omdat de verschillende tabellen in de nieuwe database qua inhoud overeenkwamen met de relevante gegevens uit de tabellen in de oude applicatie. De oude database was in het Microsoft Access bestandsformaat opgebouwd, wat het mogelijk maakte om gebruik te maken van Microsoft Data Transformation Services. Hiermee konden de gegevens op eenvoudige wijze in de nieuwe database worden geïmporteerd.

Gegevens die niet in de oude applicatie waren opgenomen, zoals informatie over locaties, printers, servers en gsm's konden in dit stadium nog niet ingevoerd worden door de afstudeerder. Dit zou teveel tijd kosten en zou niet bij hebben gedragen aan de diepgang van de opdracht. In overleg met de opdrachtgever is besloten om het invoeren van deze gegevens en koppelingen daartussen over te laten aan de systeembeheerder, na afsluiting van het ontwikkeltraject. Wel zijn er ter test enkele configuratie-items gemaakt.

5.3.2 Master pages

Tijdens de ontwikkeling van het "Architectural Proof of Concept" was al duidelijk geworden dat het ontwikkelen van een omvangrijke webapplicatie werd vergemakkelijkt door het gebruik van een "Master Page" in combinatie met CSS (de Cascading Style Sheet was tijdens de Elaboration fase al volledig aangepast maar kwam zoveel mogelijk overeen met de CSS file van de intranetpagina, om uniformiteit in huisstijl te behouden).
Deze master page maakt het mogelijk om functionaliteit op een centrale plaats binnen een webapplicatie te houden en zorgt bovendien voor visuele overerving op de verschillende pagina's van de applicatie. De master page zorgt ervoor dat iedere pagina dezelfde indeling heeft, met het enige verschil in de "ContentPlaceHolder". Binnen de webapplicatie kan na acties van de gebruiker constant een andere inhoud geladen worden in de contentplaceholder, waarbij aan de clientkant (de browser van de gebruiker) de master page en de inhoud van de contentplaceholder worden samengevoegd tot één pagina.

Het alternatief voor het gebruik van de master page was het van de grond af opnieuw opbouwen van iedere webpagina. Dit zou inhouden dat achter iedere pagina steeds dezelfde functionaliteit moest worden aangemaakt, wat in verhouding veel meer tijd zou hebben gekost. Met het gebruik van de master page werd dus ook de onderhoudbaarheid van de applicatie verbeterd.

5.3.3 Controle beheerderstatus

Aangezien het de bedoeling was om bepaalde delen van de applicatie af te schermen voor “gewone” gebruikers (configuratiebeheer, uitgebreid incidentbeheer), moest er bij het gebruik van de applicatie gecontroleerd worden of een gebruiker onder de categorie “eindgebruiker” of “beheerder” viel. Om extra “load” op de database te voorkomen en extra “code bloat” te voorkomen werd besloten om deze controle niet via de database af te wikkelen (de tabel met gebruikersgegevens zou ook weer beheerd moeten worden), maar om een instrument te gebruiken dat reeds binnen de Peinemann infrastructuur aanwezig was, namelijk het Peinemann domein met Active Directory.

Om te voorkomen dat een gebruiker de URL van een bepaalde beheerderspagina invoert en zo alsnog op de beheerderspagina terecht zou komen, werd op iedere beheerderspagina een “check” ingebouwd, die de status van de bezoeker controleert en indien de bezoeker geen rechten heeft voor het bezoeken van de pagina, wordt deze teruggestuurd naar een vergelijkbare pagina waar hij/zij wel rechten op heeft.

De controle werd bereikt door het gebruiken van “Impersonation”, wat feitelijk betekent dat niet de lokale ASP.net user op de server gebruikt wordt voor authenticatie, maar de op het systeem in het domein aangemelde gebruiker. In de controle werd allereerst opgenomen dat pagina’s alleen toegankelijk zijn voor gebruikers die aangemeld zijn in het domein “PEINEMANN”. Daarna wordt op het niveau van ingelogde gebruikersnaam gecontroleerd of de gebruiker in de groep “beheerders” valt.

5.3.4 Stored procedures

Om de applicatie zo beheerbaar mogelijk te houden werd door de afstudeerder besloten om de verwerking van het toevoegen, wijzigen en verwijderen van informatie in de database zoveel mogelijk aan de kant van de database af te handelen. Door gebruik te maken van deels het MVC-model en deels de n-Tier architectuur kon er gemakkelijk voor worden gezorgd dat bijvoorbeeld een toevoeging van een invoerveld systematisch kon worden doorgevoerd in ten eerste de pagina zelf, vervolgens in de klasse die alle methodes van een bepaald object bevat en vervolgens aan databasezijde. Stored procedures aan databasezijde zijn daar bij uitstek geschikt voor. Deze zijn eenvoudig buiten de applicatie om aan te passen.

Bovendien zijn stored procedures een manier om netwerkverkeer en CPU load zoveel mogelijk te beperken, omdat meerdere queries in één databasetransactie kunnen plaatsvinden, in plaats van aparte.

5.3.5 Configuratiebeheer

Om te kunnen beginnen aan het onderdeel incidentbeheer, moest eerst het configuratiebeheer aanwezig zijn. Incidenten dienden namelijk gekoppeld te worden aan gebruikers en aan configuratie-items, deze moesten dus eerst in het systeem aanwezig zijn.

Er werden pagina’s ontwikkeld waarmee de verschillende configuratie-items kunnen worden aangemaakt, bekeken, aangepast en verwijderd. In de klasse “beheerder” werden alle databasetransactiemethoden opgenomen, waardoor aanpassingen in interactie met de database alleen in die klasse doorgevoerd zouden hoeven worden, wat de beheerbaarheid en schaalbaarheid van de webapplicatie zeer ten goede komt. Vanuit de klasse “beheerder” worden stored procedures aangeroepen op de databaseserver.

Van ieder configuratie-item (of gebruiker of locatie) wordt alleen relevante informatie opgeslagen, naast koppelingen tussen locaties en configuratie-items en tussen gebruikers en configuratie-items.

Tijdens het ontwikkelen van het onderdeel configuratiebeheer bleek dat de lijsten waarin het overzicht gegeven werd van items (met behulp van Gridviews) niet erg overzichtelijk waren, omdat sommige types configuratie-items in grote aantallen in de database vermeld stonden. Om deze reden heeft de afstudeerder ervoor gekozen om per pagina maximaal tien items te tonen, waarbij de overige items op volgende pagina's getoond werden. De gebruiker kreeg hierbij de keus om te sorteren naar kolomnaam. Het bleek echter dat zoeken nog steeds lastig was. In overleg met de opdrachtgever kwam er dus voor de volgende iteratie een requirement bij, namelijk het toevoegen van een zoekmogelijkheid in het configuratiebeheer.

Aan het einde van de ontwikkeling van het configuratiebeheer was er voor de gebruikers uit de beheerdersgroep de mogelijkheid om:

- Locaties toe te voegen en te koppelen aan servers, printers en pc's
- Gebruikers toe te voegen en te koppelen aan gsm's en pc's
- Printers toe te voegen en te koppelen aan locaties
- Servers toe te voegen en te koppelen aan locaties
- Pc's toe te voegen en te koppelen aan locaties en gebruikers
- Gsm's toe te voegen en te koppelen aan een gebruiker
- Bovenstaande items te wijzigen
- Bovenstaande items te verwijderen

Een requirement die buiten het project zou vallen, was het creëren van de mogelijkheid om rapportages te genereren. Vooruitdenkend aan deze toekomstige situatie, zou het niet verstandig zijn om een item dat wordt verwijderd, ook daadwerkelijk uit de database te verwijderen. Bij het genereren van een rapportage (ook in verband met toekomstige gekoppelde incidenten) zouden de verwijderde items niet meer terugkomen, wat de consistentie en waarde van de rapportages niet ten goede zou komen.

Daarom werd in overleg besloten dat bij "verwijderen" van items, de items niet daadwerkelijk verwijderd zouden worden, maar in de database een status "inactief" zouden krijgen, waardoor ze in de applicatie niet meer getoond werden, maar nog wel beschikbaar zouden zijn voor rapportages. Dit leverde als nadeel op, dat de database altijd maar zou blijven groeien. Nu is gebleken dat de groei van de database niet heel snel gaat en dat de hardware dit makkelijk aankan, maar toch werd als voorzorgsmaatregel met de opdrachtgever besloten dat in ieder geval na ieder jaar gecontroleerd zou worden of de database opgeruimd kan worden, waardoor de prestaties gewaarborgd blijven. Na een bepaalde periode zouden incidenten toch niet interessant meer zijn. Incidenten voor een pc die al 3 jaar uitgefaseerd is, zouden bijvoorbeeld toch niet interessant meer zijn in rapportages.

5.3.6 Incidentbeheer

Na de voltooiing van het onderdeel "Configuratiebeheer" kon begonnen worden aan het onderdeel "Incidentbeheer". Een eindgebruiker moest een incident kunnen aanmelden en vervolgens een overzicht kunnen krijgen van de status van de door (of voor) hem of haar aangemelde incidenten. Een beheerder moest incidenten kunnen aanmelden, wijzigen, verwijderen, afmelden en bekijken.

Bij dit onderdeel wordt weer de controle op gebruikersstatus toegepast, zodat alleen beheerders toegang hebben tot het uitgebreidere incidentbeheer.

Om de eindgebruiker precies zoveel informatie te geven als deze nodig heeft en niet meer, kan deze alleen de status bekijken van zijn/haar eigen incidenten. Deze worden in een lijstje getoond, met daarbij de status van het incident (zoals eerder genoemd "Aangemeld", "In behandeling" of "Afgemeld") en de datum waarop het incident is aangemeld en afgemeld. Indien een incident de status "Afgemeld" heeft, verschijnt het incident in de "incidenthistorie" van de gebruiker. Hierbij staat dan ook vermeld wat er is gedaan om het incident op te lossen.

Voor beheerders is het incidentbeheeronderdeel een stuk uitgebreider. Naast het aanmelden van een incident kan een beheerder alle incidenten van alle gebruikers bekijken en bewerken. Indien een incident foutief is aangemaakt, kan hij deze verwijderen. Aan een door een gebruiker aangemeld incident kan een prioriteit worden toegekend (doorgaans doet het hoofd IT dit om de volgorde van werkzaamheden van de systeembeheerder te bepalen, maar de systeembeheerder kan dit zelf ook in geval van afwezigheid van het hoofd IT). Ook kan een beheerder bij een incident aangeven of het incident wordt meegenomen in rapportages of niet. Dit kan nuttig zijn in het geval van meerdere meldingen van meerdere gebruikers over hetzelfde probleem (bijvoorbeeld: een printer is kapot). Dan is het niet de bedoeling dat er in de rapportages terugkomt dat diezelfde printer vijf keer kapot is gegaan terwijl dat in werkelijkheid maar één keer was.
Als een beheerder dan alle betreffende incidenten, op één na, zou verwijderen, dan zou dat veel verwarring veroorzaken bij gebruikers aangezien zij het door hen aangemelde incident niet meer terug zouden kunnen vinden.

Een beheerder kan een incident in behandeling nemen. Hieraan wordt een datum gekoppeld, zodat later eventueel vergeleken kan worden wat de doorlooptijd van een incident is.
Ook is het in behandeling nemen een aanwijzing voor de gebruiker dat er aan zijn of haar probleem gewerkt wordt.

Wanneer de systeembeheerder het probleem heeft opgelost, voert hij een oplossing in bij het betreffende incident en meldt hij het af. Vanaf dat moment kan er niets meer gewijzigd worden aan het incident en komt het incident alleen nog terug in de historie. Hier is voor gekozen om de consistentie van de database te behouden, een afmelding is daarom definitief.

5.4 Transition Iteratie 1

De Transition fase bestond vooral uit testen en het wegwijs maken van gebruikers met de applicatie. Hierbij hoorden handleidingen voor zowel het beheer als het gebruik van de applicatie.

5.4.1 Testen

Bij het programmeren met ASP.net in Visual Studio 2005 is het mogelijk (en verstandig) om na iedere wijziging in de programmacode de applicatie te debuggen. Hiermee kan dus alle functionaliteit getest worden door de applicatie in feite in een sandbox (of testomgeving) te draaien als-ware-het-productie. Door middel van constante debugging tests kwamen er weinig fouten tot in de latere acceptatietests.

Er werd vooral getest op kwaliteit van de applicatie ten opzichte van de requirements. Werken de verschillende functionaliteiten zoals vanuit de requirements gezien verwacht mag worden?
Dit was de grootste vraag die de tests moesten beantwoorden.

Aangezien de afstudeerder het debuggen tijdens het programmeren van de webapplicatie zelf had gedaan, was er weinig kans dat hij eventuele resterende tekortkomingen van de applicatie zou opmerken tijdens tests (het testen door de ontwikkelaar wordt binnen OpenUP ook sterk afgeraden), dus werden de systeembeheerder en enkele gebruikers uitgenodigd om de applicatie te testen.

Voor de eindgebruikers die kwamen testen (drie personen in totaal) werd door de afstudeerder de volgende testset aangeleverd (het voorbeeld hieronder is de testset van testcase “Aanmelden incident (eindgebruiker)”:

datum		***datum laatste wijziging***	*16-nov-07*		
auteur	*Paul van Dam*	***laatst gewijzigd door***	*Paul van Dam*		
Te testen onderdelen	**Testfunctie omschrijving**	**Beginsituatie**	**Actie(s)**	**Verwacht resultaat**	**OK/ NOK**
Testgeval					
onderdeel 1	**Openen pagina "incidentAanmeldenEindgebruiker.aspx"**				
onderdeel 1-T01	De gebruiker gaat naar de pagina "Incident aanmelden"	De gebruiker is door de applicatie herkend als zijnde een eindgebruiker	De gebruiker klikt op "Incident aanmelden"	De pagina "incidentAanmelden Eindgebruiker.aspx" wordt geopend	
onderdeel 2	**Foute invoer**				
onderdeel 2-T01	De gebruiker meldt een incident aan zonder daarbij een item te selecteren	De pagina "incidentAanmeldenEindgebruiker.aspx" is geopend	De gebruiker voert een onderwerp en omschrijving in, maar kiest geen type item uit het lijstje	Er wordt een foutmelding gegeven die de gebruiker instrueert om eerst een item te kiezen	

onderdeel 2-T02	De gebruiker meldt een incident aan zonder een onderwerp in te voeren	De pagina "incidentAanmeldenEindgebruiker.aspx" is geopend	De gebruiker kiest een type item en een item, maar voert geen onderwerp in en klikt op aanmelden	Er wordt een foutmelding gegeven (Verplicht veld!)
onderdeel 2-T03	De gebruiker meldt een incident aan zonder een omschrijving in te voeren	De pagina "incidentAanmeldenEindgebruiker.aspx" is geopend	De gebruiker kiest een type item en een item, maar voert geen omschrijving in en klikt op aanmelden	Er wordt een foutmelding gegeven (Verplicht veld!)
onderdeel 3	**Aanmelden incidenten**			
onderdeel 3-T01	De gebruiker meldt een "overig" incident aan	De pagina "incidentAanmeldenEindgebruiker.aspx" is geopend	De pagina "incidentAanmeldenEindgebruiker.aspx" is geopend	Het incident wordt aangemeld en de pagina met openstaande incidenten wordt getoond met daarin de incidenten die in naam van de nu ingelogde gebruiker zijn aangemaakt
onderdeel 3-T02	De gebruiker meldt een "PC" incident aan	De pagina "incidentAanmeldenEindgebruiker.aspx" is geopend	De gebruiker voert een onderwerp en omschrijving in, kiest een gebruikersnaam en kiest als Type item "PC", kiest een PC uit de lijst en klikt op aanmelden	Het incident wordt aangemeld en de pagina met openstaande incidenten wordt getoond met daarin de incidenten die in naam van de nu ingelogde gebruiker zijn aangemaakt
onderdeel 3-T03	De gebruiker meldt een "GSM" incident aan	De pagina "incidentAanmeldenEindgebruiker.aspx" is geopend	De gebruiker voert een onderwerp en omschrijving in, kiest een gebruikersnaam en kiest als Type item "GSM", kiest een GSM uit de lijst en klikt op aanmelden	Het incident wordt aangemeld en de pagina met openstaande incidenten wordt getoond met daarin de incidenten die in naam van de nu ingelogde gebruiker zijn aangemaakt
onderdeel 3-T04	De gebruiker meldt een "Printer" incident aan	De pagina "incidentAanmeldenEindgebruiker.aspx" is geopend	De gebruiker voert een onderwerp en omschrijving in, kiest een gebruikersnaam en kiest als Type item "Printer", kiest een Printer uit de lijst en klikt op aanmelden	Het incident wordt aangemeld en de pagina met openstaande incidenten wordt getoond met daarin de incidenten die in naam van de nu ingelogde gebruiker zijn aangemaakt

Figuur 18: Voorbeeld testset

Helaas kon deze testset alleen in dit document opgenomen worden met wat vervorming van de cellen. De volledige testsets zijn echter allemaal te vinden in de bijlagen.

Opvallend genoeg bleek uit de tests van alle drie de testpersonen dat er zich geen onverwachte resultaten voordeden. Dit gold ook voor de tests voor het gedeelte van de applicatie dat alleen voor beheerders toegankelijk was, wat getest werd door de systeembeheerder en het hoofd IT. Ook hiervan zijn de testsets terug te vinden in de bijlagen.

Uit een testsessie met een gebruiker bleek nog wel dat het voor haar niet duidelijk was hoe zij een incident voor een andere gebruiker kon aanmelden. Uit deze syntactische test bleek dus dat de gebruiksvriendelijkheid bij het aanmelden van een incident tekortschoot. Een nieuwe requirement voor de volgende iteratie werd dus het toevoegen van een “help” knop op het betreffende scherm. Verder was het commentaar van de testers dat de lay-out en stijl van de webapplicatie hen zeer bekend voorkwam en dat de applicatie eenvoudig te begrijpen en te navigeren was.

5.4.2 Handleidingen

Om het gebruik en beheer van de webapplicatie te vergemakkelijken zijn er door de afstudeerder twee handleidingen opgeleverd. Één voor eindgebruikers en één voor beheerders. De handleiding voor eindgebruikers werd via de Peinemann intranetpagina beschikbaar gesteld aan de gebruikers.

In de handleiding voor eindgebruikers wordt allereerst uitgelegd welke functionaliteit er voor de eindgebruikers beschikbaar is in de applicatie en wat zij hier precies mee kunnen. Daarna wordt stap voor stap beschreven hoe de verschillende functionaliteit van de applicatie werkt aan de hand van voorbeelden en screenshots.

De handleiding voor beheerders heeft in feite dezelfde indeling. Van beheerders wordt echter verwacht dat zij technische termen beter begrijpen dan eindgebruikers dat zouden kunnen.

Beide handleidingen zijn bij dit verslag als bijlage beschikbaar.

Met het opleveren van de applicatie en de bijpassende heringerichte beheerorganisatie, waarmee de overdracht van de applicatie naar de beheerders en gebruikers een feit was, kon de Transition fase en daarmee de eerste iteratie worden afgesloten.

6. Iteratie 2: Remote uitlezen en zoekfunctie

De tweede iteratie van het UP-project was vooral bedoeld om de requirements met een wat lagere prioriteit te behandelen. Belangrijkste item hierbij was de functionaliteit voor het remote uitlezen van computersystemen op aanwezige hardware en software.

Naast de vooraf geplande requirements werden in deze iteratie ook de requirements opgenomen die tijdens de eerste iteratie aan het licht waren gekomen. In feite ging het hier dan om het uitbreiden van het configuratiebeheer met een zoekfunctie en de wat minder omvangrijke toevoeging van een “help” knop bij het aanmelden van incidenten voor eindgebruikers.

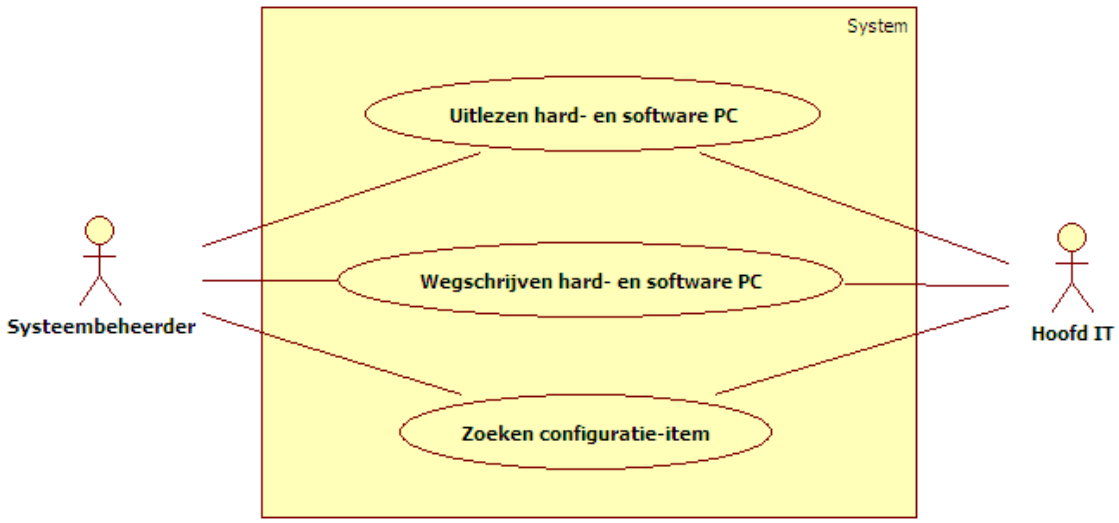
De omvang van deze tweede iteratie bleek niet in verhouding te staan met de eerste iteratie, waarbij in feite aan veel meer requirements is voldaan dan in deze tweede iteratie. In iteratie twee werd vooral functionaliteit uitgebreid, wat de applicatie meer diepgang gaf en het werk van de systeembeheerder en het hoofd IT vereenvoudigde.

6.1 Inception Iteratie 2

Bij het beginnen aan de inception fase van de tweede iteratie bleek dat er voor de toe te voegen functionaliteit weinig hoefde te veranderen aan projectdocumentatie. Het vision document werd aangepast om zo de scope van het project te wijzigen, het project glossary werd aangepast met nieuw te gebruiken termen en er werd een aantal nieuwe use cases bedacht. De belangrijkste wijzigingen werden in deze fase gemaakt in de use cases.

6.1.1 Use Case Model

De wijzigingen in het use case model betroffen het toevoegen van een drietal use cases.



Figuur 19: Use case model iteratie 2

Voor het toevoegen van een knop met “help” voor de eindgebruiker achtte de afstudeerder het niet noodzakelijk om de UML modellen en andere documentatie te gaan herschrijven, aangezien dit een zeer kleine wijziging betrof.

De wijzigingen in de tweede iteratie boden vooral de systeembeheerder en het hoofd IT nieuwe functionaliteit. De mogelijkheid tot het zoeken naar een bepaald configuratie-item door middel van het opgeven van een zoekwoord zou de tijd tussen zoeken en vinden aanzienlijk verkorten.
Bij de in de eerste iteratie opgeleverde applicatie waren de configuratie-items namelijk verdeeld over meerdere pagina's waarbij er per pagina tien werden weergegeven. Het vinden van een bepaald item vereiste dan ook nogal wat bladerwerk.

Om te voorkomen dat er geen informatie beschikbaar zou zijn over pc-systemen die uit staan, moest er ook de mogelijkheid zijn om de gegevens na de eerste uitlezing op te slaan in de database.

Met bovenstaande use cases kon de afstudeerder aan de slag om de nieuwe functionaliteit voor de tweede iteratie te ontwikkelen.

6.2 Elaboration Iteratie 2

In de elaboration fase van de tweede iteratie werden de use cases uitgewerkt tot volledig gespecificeerde use cases. Deze zijn terug te vinden in de bijlagen. Dit geldt ook voor de sequentiediagrammen.

Aangezien de benodigde attributen voor het toevoegen van hard- en software-informatie al aanwezig waren in de database en de verschillende klassen, hoefde alleen nog de user interface aan te worden gepast en moest er gekeken worden naar een toevoeging aan de architectuur voor het uitlezen van pc-systemen.

In overleg met de opdrachtgever en systeembeheerder werd besloten dat alleen pc's van afstand zouden worden uitgelezen en dat dit bij servers niet zou gebeuren. Servers zijn te bedrijfskritiek om risico's tot vertragingen te lopen door het uitvoeren van een remote uitleesactie.

6.2.1 Architectuur

Uit onderzoek bleek dat ASP.net 2.0 de ingebouwde mogelijkheid heeft om met behulp van WMI (Windows Management Instrumentation) een scala aan systeemkenmerken uit te lezen. Er zaten echter wel nadelen aan. Het uitlezen van een systeem over het netwerk kan (bij grote hoeveelheden geïnstalleerde programma's enzovoorts) vrij lang duren, tot ongeveer een minuut. Om dit via de webapplicatie zelf te doen, zou een webapplicatie opleveren die tijdens het gebruik voor remote uitlezen, niet beschikbaar is voor bijvoorbeeld het aanmelden van incidenten. Hiervoor moest dus naar een oplossing worden gezocht.

Het alternatief voor uitlezen via het netwerk zou het bouwen van een applicatie zijn die geïnstalleerd diende te worden op alle pc's binnen het netwerk. Deze applicatie zou dan het uitlezen voor zijn rekening nemen en de gegevens in de centrale database actueel houden. Het voordeel hiervan is dat de webapplicatie zo geen vertraging oploopt. Het nadeel is de beheerbaarheid van de installaties op de clients, bijvoorbeeld in het geval van een update. Een verder nadeel is dat de applicatie op gezette tijdstippen een update naar de database zal moeten schrijven, wat een constante vertraging van het werkstation voor gebruikers oplevert.

Een oplossing die ervoor zorgde dat er geen vertraging in de webapplicatie optrad en waarmee de werkstations van de gebruikers alleen belast worden op momenten dat de systeembeheerder of het hoofd IT de bijzonderheden opvraagt, bleek te liggen in het gebruik maken van web services. Als al het remote uitleeswerk werd gedaan door een web service, zouden de webapplicatie en de werkstations ontlast worden.

Het gebruik van een web service levert een nieuwe dimensie op in de architectuur van de applicatie, namelijk de eerste stap richting een Service Oriented Architecture.

Gebruik maken van zo'n architectuur levert de mogelijkheid op tot bijna eindeloze schaalbaarheid en verdere uitbreidbaarheid van een applicatie. Allerlei functionaliteit kan simpelweg aan de applicatie toegevoegd worden door gegevens naar een web service te zenden, die "iets" uitvoert op basis van die gegevens en vervolgens nieuwe gegevens als resultaat terugstuurt. Het enige wat de applicatie dan nog hoeft te doen is het interpreteren van die gegevens.
Doordat web services zich op andere servers in het netwerk kunnen bevinden dan de applicatie zelf, bieden services de mogelijkheid om rekenkracht te verdelen waardoor de aanvankelijke applicatieserver niet snel overbelast zal raken.

In overleg met de systeembeheerder en het hoofd IT werd daarom besloten om te kiezen voor een web service in combinatie met WMI om de werkstations remote te kunnen uitlezen.

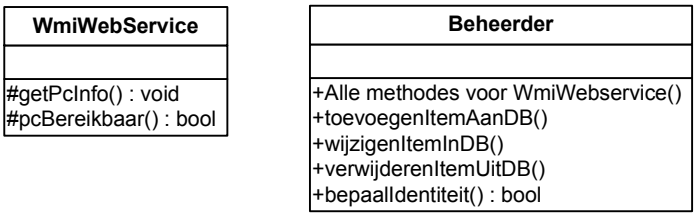
6.2.2 Klassendiagrammen

Aangezien in de eerste iteratie al rekening was gehouden met de attributen die er bij het toevoegen van de functionaliteit "remote uitlezen" bij zouden komen, waren de veranderingen in de klassendiagrammen niet groot.

De attributen van de klasse "Pc" die bij deze tweede iteratie vooral van belang waren, zijn:

- CpuCount (aantal processoren, processorkernen of virtuele processorkernen)
- CpuType (het type processor en de kloksnelheid)
- Ram (de hoeveelheid fysiek systeemgeheugen gemeten in Megabytes)
- OperatingSystem (de naam, versie en service packniveau van het besturingssysteem)
- InstalledSoftware (de software die op het systeem geïnstalleerd is)
- RunningProcesses (de processen die in het besturingssysteem draaien)

De enige klasse die er in het implementatieklassendiagram van de tweede iteratie bijkwam was dan ook de web service, die voor het gemak even "WMIWebService" werd genoemd:



Figuur 20: Toevoegingen aan implementatieklassendiagram iteratie 2

Aangezien de klasse "beheerder" alle communicatie met de database onderhield, was het ook verstandig om deze klasse alle communicatie met de webservice te laten verrichten. Zo waren belangrijke methodes afgeschermd van de eindgebruikers en makkelijk op één plaats te beheren.

"Alle methodes voor WmiWebservice" slaan in dit geval op de methode die een pcID (hostnaam) naar de webservice stuurt en de methodes die de terug ontvangen data interpreteren en doorgeven aan de View, oftewel de webpagina.

Aangezien er alleen WMI gegevens kunnen worden opgehaald van een systeem dat aan het netwerk gekoppeld is en aanstaat, voert de webservice bij iedere aanvraag eerst een controle uit of het betreffende systeem bereikbaar is in het netwerk.

De methodes voor de zoekfuncties bij de verschillende configuratie-itemoverzichten werden afgehandeld door stored procedures aan databasezijde.

6.2.3 Navigation Map

Ook in de navigation map veranderde in de tweede iteratie weinig. De in de eerste iteratie aangemaakte pagina's waren al voorbereid op de in iteratie twee toegevoegde gegevens en ook de zoekfuncties konden op de reeds bestaande pagina's worden toegevoegd.

Ook hier was de webservice dus weer de enige toevoeging.

Figuur 21: WmiWebService als toevoeging aan de navigation map

6.3 Construction Iteratie 2

De focus lag in de Construction fase van de tweede iteratie op een drietal requirements, in volgorde van belangrijkheid:

1. Remote uitlezen van pc's
2. Zoekfunctie configuratie-items
3. Toevoegen "help" knop bij aanmelden incident voor eindgebruiker

Aan de laatste requirement wordt in dit verslag verder geen aandacht besteed, omdat de omvang en diepgang hiervan te gering zijn.

6.3.1 Remote uitlezen pc's

Omdat deze requirement door de opdrachtgever van hogere prioriteit werd bevonden, werd door de afstudeerder begonnen met het werken aan de Web Service voor het remote uitlezen van pc's. Omdat reeds besloten was dat servers om betrouwbaarheid- en performanceredenen niet uitgelezen zouden worden, kon de focus gelegd worden op het uitlezen van pc's.

Bij het gebruik van de Windows Management Instrumentation zijn er vanuit Microsoft Windows in een zogenaamde WMI repository of registry bepaalde Windows Management Classes te vinden. Deze classes bevatten de informatie over het systeem, zoals in dit geval hardware, software en processen. Er is maar één mogelijke manier om het WMI registry uit te lezen vanuit ASP.net code en dat is met het gebruik van WQL, oftewel WMI Query Language. Deze taal is WQL genoemd omdat de syntax vergelijkbaar is met die van SQL. In feite spreekt men de WMI classes dus aan zoals men een database zou aanspreken, met behulp van Select statements en dergelijke.

Om een systeem over het netwerk uit te lezen, heeft Microsoft een beveiliging ingebouwd zodat niet zomaar iedereen dit kan. WMI biedt namelijk niet alleen de mogelijkheid om gegevens uit te lezen, maar ook om gegevens weg te schrijven naar een remote WMI registry. Hoewel dit buiten de scope van dit project valt, is dit belangrijke informatie omdat er gelet moest worden op het rechtenniveau. In de code van de WmiWebservice moest dus impersonation plaatsvinden, de user account die gebruikt werd voor het remote uitlezen moest namelijk lokale administrator rechten hebben op de pc's die werden uitgelezen. Hiervoor is een account in het leven geroepen binnen de Peinemann Active

Directory die rechten heeft om de WMI registry van systemen binnen het domein uit te lezen, maar geen rechten heeft om hier naartoe te schrijven.

Met de aanwezigheid van werkende WQL queries kon een systeem uitgelezen worden vanuit de web service. Als een systeem echter niet aanstond of om een andere reden niet reageerde op een WMI query, werd door de afstudeerder een andere methode gecreëerd in de webservice die eerst een simpele WMI query naar de betreffende host stuurt. Als deze niet binnen een bepaalde tijd reageerde(variabel in te stellen time-out, voorlopig op 10 seconden ingesteld), worden de gegevens over hardware, software en processen niet opgehaald vanaf de host maar worden de gegevens uit de database gehaald, voor zover deze reeds opgeslagen waren.

De gebruikersinterface werd zodanig ingericht dat via WMI opgehaalde gegevens direct met één klik weggeschreven kunnen worden naar de database, zodat de gegevens de eerstvolgende keer dat de host niet bereikbaar is, uit de database gehaald kunnen worden. Hoewel deze gegevens dan mogelijk niet meer actueel zijn, geeft dit de systeembeheerder of het hoofd IT toch wat informatie over de machine.
Ook werd de mogelijkheid toegevoegd om direct te kijken in de database, in het geval van haast. Het ontvangen van resultaten op een WMI query kan namelijk vrij lang duren, tot wel één minuut bij een grote hoeveelheid gegevens en een relatief trage pc als host.

6.3.2 Zoekfunctie configuratie-items

Omdat bij sommige types configuratie-item de overzichten zo lang werden dat het gezochte item lastig gevonden kon worden, werd door de opdrachtgever gevraagd hier een zoekfunctie in te maken. De zoekfunctie bleek bij navraag alleen nuttig bij “Gebruikers”, “Pc’s”, “Gsm’s” en “Printers”.
Bij “Locaties” en “Servers” bleek geen zoekfunctie nodig aangezien de aantallen items in deze categorieën laag zouden blijven.

Omwille van het behouden van een “schone” gebruikersinterface en een overzichtelijke werking van de applicatie werd in overleg met de opdrachtgever besloten dat het zoeken moest werken met één enkele textbox en één enkele zoekknop (zoals bij veel internetzoekmachines gedaan wordt). De zoekmethode bij ieder type item was verschillend:

Bij PC wordt zowel gezocht op pcID als op bijbehorende gebruikersnaam en locatie. Werd bijvoorbeeld de letter H ingevoerd als zoekwoord, dan bestonden de resultaten uit:

- Pc’s waarvan het pcID begint met een H
- Pc’s die gekoppeld zijn aan een gebruikersnaam die begint met de H
- Pc’s die gekoppeld zijn aan een locatie die begint met de H

Dit levert een veel overzichtelijker situatie op en bij het gebruik van specifieke zoekwoorden blijft het aantal ongewenste zoekresultaten tot een minimum beperkt.

Bij het zoeken naar printers wordt gekeken naar printerID, merk en locatie. Bij gsm’s wordt gezocht naar gsmID en bijbehorende gebruikersnaam. Bij gebruikers wordt gezocht op voornaam, achternaam en gebruikersnaam.

6.4 Transition Iteratie 2

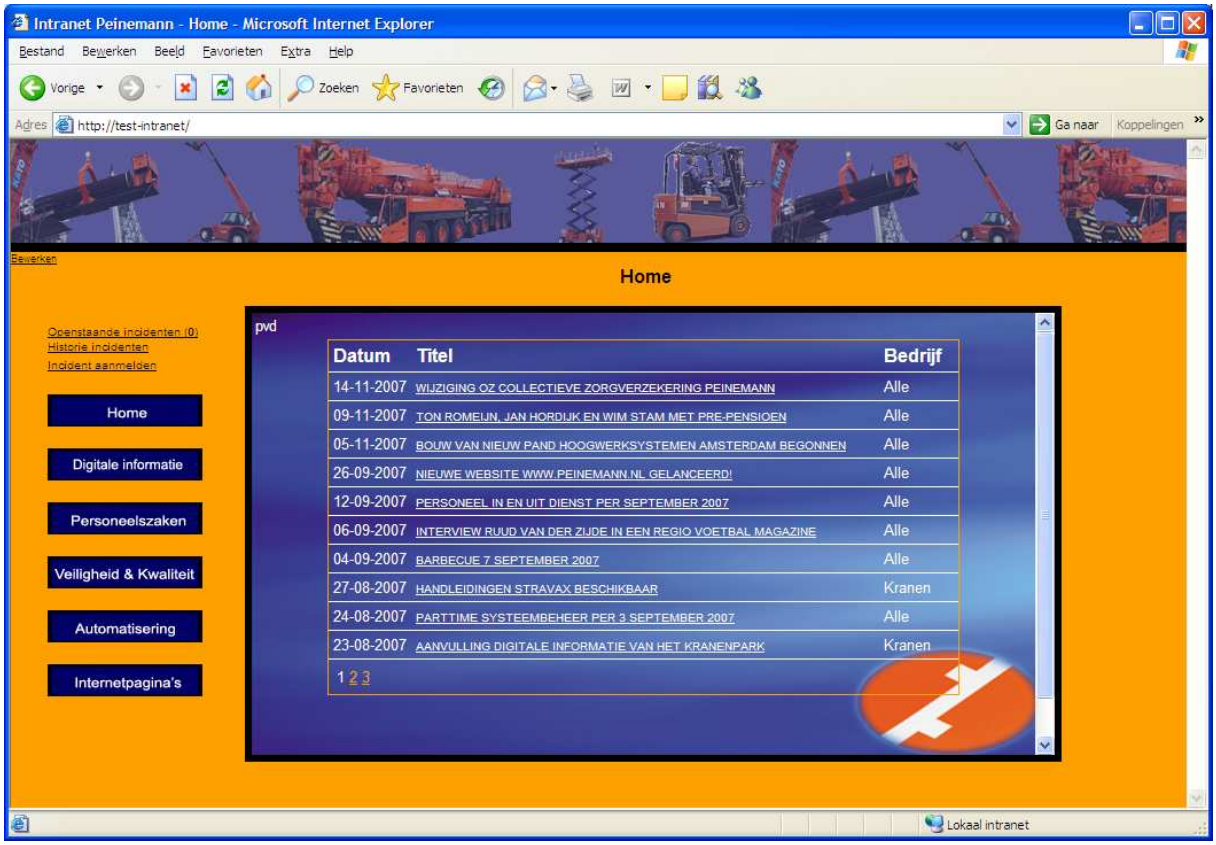
In de transition fase van de tweede iteratie vond het herschrijven van de handleidingen (waar nodig) plaats en het testen van de invloed van de gemaakte wijzigingen op de functionaliteit van de eerste iteratie.

Dezelfde testsets konden gebruikt worden voor deze tweede transition fase en ook hier waren de resultaten allen goed, er werden geen tekortkomingen gevonden en de applicatie kon volledig in gebruik genomen worden.

Tests werden wederom uitgevoerd door de systeembeheerder, het hoofd IT en een drietal gebruikers.

Nu de applicatie volledig af en getest was, kon de integratie met het vernieuwde intranet plaatsvinden en konden de beide projecten “live”(het nieuwe intranet was een project van de systeembeheerder, hij voerde dan ook de integratie van PIMP met de intranetsite uit).

Nu de integratie met het intranet een feit was, kon PIMP bezocht worden via de intranetpagina en niet meer door het invoeren van een URL in de browser.



Figuur 22: Integratie met het Peinemann intranet

Via de koppelingen links bovenin de pagina kan een gebruiker de openstaande incidenten bekijken (het aantal openstaande incidenten wordt getoond op de startpagina), de historie van afgemelde incidenten bekijken en een nieuw incident aanmelden.

7. Evaluatie ontwikkeltraject PIMP en beheerorganisatie

In dit hoofdstuk beschrijf ik de evaluatie van zowel het opgeleverde product als het proces. Deze evaluatie maakt duidelijk of ik gedurende het proces de juiste keuzes heb gemaakt. Ook geef ik in dit hoofdstuk weer of ik een eventueel volgend project weer op dezelfde manier zal aanpakken als ik bij dit project gedaan heb.

7.1 Productevaluatie

Het product is mijns inziens goed gelukt. Als gebruiker is het systeem eenvoudig te gebruiken en laagdrempelig. De systeembeheerder en het hoofd IT hebben nu een waardevol gereedschap om hun dagelijkse werkzaamheden te vergemakkelijken en de resultaten van het systeembeheer te optimaliseren. De doorlooptijd van een incident heeft met behulp van PIMP de potentie om veel korter te zijn dan in de oude situatie. Het papieren archief is vervangen door een digitaal archief en het zoeken naar status van configuratie-items is sterk verbeterd.

Iets dat mij erg goed geholpen heeft bij de ontwikkeling van het product is het feit dat ik toegang had tot alle mogelijke bronnen binnen het bedrijf en een goed overzicht kon krijgen van de bedrijfsprocessen en wie wat nodig had op het gebied van beheer. Dit heeft mij zeer goed geholpen bij het ten eerste op papier zetten van requirements en ten tweede het voldoen aan die requirements door middel van het opleveren van product features. Daarom denk ik dat ik een product overlegd heb, dat perfect past bij Peinemann als bedrijf en de afdeling automatisering als beheerorganisatie.
Met de webapplicatie PIMP als rode draad is het voor de gebruikers en beheerders niet meer nodig om buiten de beheerorganisatie om te werken, de organisatie en applicatie zijn bij alle soorten IT-incidenten volledig toepasbaar.

De keuze voor een maatwerkproduct is mij goed bevallen, ik denk niet dat een ander (in de markt verkrijgbaar) product voor dezelfde resultaten had kunnen zorgen of zo goed bij de organisatie had kunnen passen als het door mij opgeleverde product.

Hierbij moet wel gezegd worden dat het product nog niet perfect is. Het genereren van rapportages is een belangrijk onderdeel van de groeiende beheerorganisatie. Het is steeds meer van belang om door middel van uit de historie verzamelde gegevens te analyseren om zo tot een evaluatie te kunnen komen van zowel de werkzaamheden van de systeembeheerder als de integriteit van de technische infrastructuur. Het mooiste zou zijn geweest als deze functionaliteit reeds opgeleverd had kunnen worden in PIMP. Andere requirements bleken echter prioriteit te hebben, waardoor het onzeker is of verdere functionaliteit nog zal worden ingebouwd. De voorbereidende requirements hiervoor zijn echter al wel aanwezig, waardoor PIMP in feite een open einde heeft.

7.2 Procesevaluatie

Het proces is volledig gericht geweest op UP en dan met de nadruk op OpenUP. Voor OpenUP heb ik tijdens het proces gekozen (ten opzichte van andere UP methodes) omdat OpenUP meer vrijheid geeft in het ontwikkeltraject en goed gedocumenteerd is. OpenUP is wat mij betreft een meer begrijpbare methode dan bijvoorbeeld RUP, waar vanuit de opleiding vooral de nadruk op werd gelegd. Ik zie OpenUP dan ook, zeker omdat het nog volop in ontwikkeling is, als een belangrijke methode in de toekomst, die misschien ook in het onderwijs toegepast zou kunnen worden.

Het volgen van het OpenUP proces was min of meer natuurlijk, de op te leveren producten waren wat mij betreft allemaal erg nuttig bij het ontwikkeltraject(bij sommige UP methodes twijfelde ik aan het nut van het opleveren van bepaalde documentatie, de toepasbaarheid op specifieke projecten is vaak niet hard te maken).

Binnen het ontwikkelproces werd ik ook door de opdrachtgever erg vrijgelaten. Bij Peinemann zijn geen vaste methodes en technieken voor het uitvoeren van IT-projecten. Dit omdat Peinemann in feite geen IT-bedrijf is, de IT bestaat bij Peinemann enkel voor de gebruikers en zeker niet andersom. Ik vond deze manier van werken prettig omdat ik zo zelf de methode en het proces kon kiezen die het beste bij mijzelf en bij dit project pasten.

Het OpenUP proces heb ik volledig gevolgd in twee basisiteraties. Hierbij heb ik gebruik gemaakt van UML als modelleertechniek. Het gebruik van UML wordt namelijk sterk aangeraden bij OpenUP en de verschillende binnen OpenUP aanwezige templates zijn in hoofdzaak gebaseerd op UML. Doordat de verschillende methoden en technieken zo goed bij elkaar passen is het relatief eenvoudig om een goed product op te leveren, als de requirements maar goed geformuleerd zijn en beheerd worden.

Een tekortkoming van de meeste UP methodes (zo ook OpenUP, alhoewel OpenUP er wat mij betreft beter uitkomt dan de andere methodes) is het gebrek aan hulpmiddelen voor requirements management. Het op papier hebben van de meeste requirements en het constante controleren (door middel van requirements traceability) of er aan die requirements voldaan is, wordt onvoldoende gearticuleerd in de UP methodes. Daarom ben ik er dan ook tevreden over, dat ik de periode van “requirements elicitation” buiten het ontwikkeltraject heb gehouden.

Over de testtechniek TMap ben ik minder tevreden. Het testproces in het algemeen zou ik in een volgend project anders aanpakken. Bij het schrijven van een webapplicatie met behulp van ASP.net en Microsoft Visual Studio is het constante debuggen van de applicatie (op technisch gebied) eigenlijk al voldoende om een goede werking van de applicatie te kunnen garanderen. Het enige waarvoor ik TMap in dit project nuttig vond was de integratietest bij inbouwen van nieuwe functionaliteit en de uiteindelijke gebruikers- en beheerdersacceptatietests. De systeemtests kwamen wat mij betreft onvoldoende uit de verf. In een volgend project zou ik in de voorbereidingsfase net als aan requirements management ook aan “test management” veel meer aandacht willen besteden.

Ten slotte ben ik over de gehele periode erg tevreden met mijn werk en het daarbij opgeleverde product en de opdrachtgever heeft de bruikbaarheid van zijn nieuwe beheerorganisatie en bijbehorende webapplicatie. Buiten de hierboven omschreven verbeterpunten, denk ik dat het project als geslaagd gekenmerkt mag worden, mede dankzij de goede begeleiding vanuit Peinemann en de hulpvaardigheid van alle betrokken partijen.

Literatuurlijst

Boeken

- Developing Web Applications with Microsoft Visual C#.net (70-315), Jeff Webb, Onderdeel van MCAD/MCSD self-paced training kit
- Praktisch UML, 3e editie, J. Warmer & A. Kleppe, ISBN 9043008125
- Testen volgens TMap, 2e druk, M. Pol & R. Teunissen & E. van Veenendaal, ISBN 9072194586
- Professional ASP.net 2.0 – Bill Evjen, Devin, Rader & Srinivasa Sivkumar, ISBN10: 0764576100 en ISBN13: 9780764576102
- Managing Software Requirements, A Use Case Approach, second edition, D. Leffingwell & D. Widrig, ISBN 9780321122476

Internet

- http://www.wikipedia.org
- http://www.google.com
- http://msdn.microsoft.com
- http://www.aspnl.com
- http://www.ASP.net
- http://www.codeproject.com
- http://www.w3.org
- http://forums.ASP.net
- http://gathering.tweakers.net
- http://www.eclipse.org/epf/general/OpenUP.pdf

Begrippenlijst

Term / afkorting	Omschrijving
ASP.net	De techniek waarmee PIMP is ontwikkeld
C#	De programmeertaal waarmee PIMP is ontwikkeld
Database	Digitaal gegevensbestand waarin informatie wordt opgeslagen en gewijzigd en verwijderd kan worden
Klasse	Verzameling van objecten met overeenkomstige eigenschappen
PIMP	Peinemann Incident Management Programma
Sequentiediagram	Visuele weergave van een use case waarbij duidelijk wordt welke klassen er gebruikt worden
SQL	Structured Query Language, taal die gebruikt wordt bij interactie met databases
Stored procedures	Databasequeries die aan databasezijde worden opgeslagen en vanuit de code kunnen worden aangeroepen
TMap	Test Management Approach
UP	Unified Process, methode bij ontwikkeling van een softwareproduct
Use Case	Opsomming van de te doorlopen stappen van een functionaliteit

Figurenlijst