Afstudeerverslag

Software t.b.v. kostenregistratie binnen VoIP

in opdracht van

Realisatie : Yong Bing Hu
Studentnummer : 20024721
Afstudeerperiode : 2005.1-1
Opleiding : I&I / IVIT 3
Richting : Infrastructuur Ontwikkeling
Startdatum : 7 februari 2005
Einddatum : 10 juni 2005
Document : afstudeerverslag_20024721_2.7.doc
Versie/Revisie : 2.7
Releasedatum : 10 Juni 2005

Deze pagina is opzettelijk leeg gelaten

Voorwoord

In de vier maanden die ik bij Lijbrandt Telecom heb doorgebracht is er een compleet nieuwe wereld voor me opengegaan op het gebied van telefonie-infrastructuren. Voorheen was ik er niet van bewust hoe telefooncentrales werkten en hoe ingewikkeld het bijvoorbeeld is om tarieven voor gesprekken te implementeren en deze te hanteren voor het verrekenen van gesprekskosten.

Ik wil graag mijn dank betuigen aan Lijbrandt Telecom en mijn opdrachtgever binnen Lijbrandt Telecom en manager techniek, service en bouw: Ing. M.J. Schreuder.
Dit voor het aanbieden van een afstudeeropdracht, medewerking, aansturing, advies en faciliteiten gedurende mijn afstuderen binnen Lijbrandt Telecom.
Tevens wil ik de heer Ing. A. van der Molen bedanken, die in het verleden bij een telecombedrijf werkzaam was en mij informatie en advies heeft verschaft op het gebied van dienstontwikkeling en softwareontwikkelingtrajecten.

Hillegom, Juni 2005

Yong Bing Hu

Referaat

Afstudeeropdracht van Yong Bing Hu. Eindverslag betreft het ontwikkeling van software ten behoeve van kostenregistratie binnen Voice over IP.

Dit verslag beschrijft inhoudelijk hoe de software tot stand is gekomen. Tevens verschaft het inzicht in de keuzes die gemaakt zijn en de manier waarop bepaalde keuzes zijn afgewogen.

Descriptoren:

- Voice over IP
- Kostenregistratie
- SQL
- CGI
- Extreme Programming

Inhoudsopgave

1. Inleiding

In het kader van het vierde leerjaar van de opleiding I&I aan de Haagse Hogeschool te Den Haag, heb ik mijn afstudeeropdracht (Software ten behoeve van kostenregistratie binnen VoIP) verricht binnen Lijbrandt Telecom. Ten behoeve van het afstudeerproject heb ik dit eindverslag geschreven.

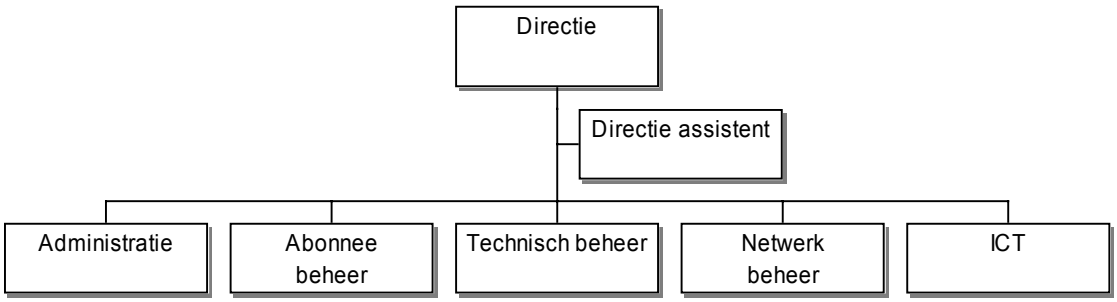
VoIP staat voor Voice Over Internet Protocol en is een wijze van telefonie dat via het internet verloopt in plaats van de conventionele POTS(Plain Old Telephone System). POTS is de conventionele techniek in telefonie dat gebruikt maakt van analoge telefonie via koper draden.

Dit eindverslag beschrijft het uitvoeringsproces van mijn afstudeeropdracht. Ik wil met dit verslag de lezers inzicht bieden over mijn werkzaamheden en werkwijze met betrekking tot mijn afstudeeropdracht binnen Lijbrandt Telecom. Daarnaast geef ik in dit document mijn opvattingen weer over wat er in dit document wordt behandeld. Het is raadzaam om de hoofdstukken achtereenvolgend te lezen, dit om onduidelijkheden te voorkomen.

In hoofdstuk 2 geef ik een beschrijving van Lijbrandt Telecom en mijn positie binnen het bedrijf gedurende mijn afstudeerperiode.
In hoofdstuk 3 hoofdstuk komt mijn opdrachtomschrijving naar voren.
In hoofdstuk 4 vertel ik u meer over hoe ik deze opdracht dacht uit te voeren, dit in de vorm van een plan van aanpak.
In hoofdstuk 5 beschrijf ik de uitgangssituatie van Lijbrandt Telecom.
In hoofdstuk 6 vertel ik hoe ik de klanteisen heb achterhaald.
In hoofdstuk 7 komt de gewenste situatie van Lijbrandt Telecom aan bod.
In hoofdstuk 8 beschrijf ik mijn bevindingen over het ontwerp van de database en software met use case beschrijvingen en toestandsdiagrammen.
In hoofdstuk 9 werk ik mijn globale planning uit tot een releaseplan en stel ik prioriteiten aan de iteraties die ik ga ontwikkelen om zo het essentiële als eerst te ontwikkelen.
In hoofdstuk 10 geef ik een korte beschrijving over de werkzaamheden tijdens het opstellen van de VoIP testomgeving.
In hoofdstuk 11 beschrijf ik hoe ik een prototype heb ontwikkeld om zo te achterhalen of het reëel was om het probleem op deze wijze op te lossen. Tevens beschrijf ik hoe ik de problemen heb weten op te lossen in een workaround.
In hoofdstuk 12 worden de problemen omschreven waar ik tegen aanliep bij het ontwikkelen van de software.
In hoofdstuk 13 komen de acceptatietesten aan bod. In deze acceptatietesten beschrijf ik de wijze waarop ik de acceptatietesten heb uitgevoerd.
Tot slot zal ik in hoofdstuk 14 een project- en procesevaluatie geven over wat ik van de afstudeeropdracht vond en wat ik in het vervolg beter kan doen.

2. Lijbrandt Telecom

Lijbrandt Telecom is een telecommunicatie bedrijf dat zich richt op de exploitatie van telecommunicatie diensten en internet via een eigen glasvezel netwerk door heel Nederland.
Lijbrandt Telecom is gevestigd in Hillegom en heeft circa 30 werknemers in dienst. De leiding bestaat uit twee directeuren en een directie assistent die de dagelijkse leiding in handen heeft. Daaronder vallen de afdelingen: 'Administratie', 'Abonneebeheer', 'Technisch beheer', 'Netwerkbeheer' en 'ICT'.
Ik heb de organisatiestructuur in kaart kunnen brengen met behulp van het hoofd van de afdeling Administratie: R. Schäfer.
Afbeelding 2.1 geeft het organigram van Lijbrandt Telecom weer. Dit is een schematisch overzicht van de organisatiestructuur binnen Lijbrandt Telecom:



Afbeelding 2.1 'Organigram Lijbrandt Telecom'

2.1 Geschiedenis

Voor het achterhalen van de geschiedenis van Lijbrandt Telecom heb ik gesproken met de directie assistente S. Hofstra. Deze heeft me meer kunnen vertellen over het bedrijf, omdat zij er al vanaf het begin werkzaam is.
Lijbrandt Telecom Nederland B.V. is opgericht door twee personen. Het bedrijf heette voorheen Technotel Installatie B.V. Dit bedrijf verzorgde telecommunicatie installaties bij zorginstellingen door heel Nederland. Omdat er destijds gevraagd werd aan Technotel Installatietechniek B.V. of het bedrijf ook de exploitatie van het aangelegde netwerk wilde verzorgen, is Technotel Exploitatie hieruit voort gekomen.

Omdat de vraag naar exploitatie groeide is het bedrijf doorgegroeid naar een volwaardige aanbieder van telecommunicatiediensten onder de naam Lijbrandt Telecom Nederland B.V. Lijbrandt Telecom Infrastructuren B.V. is een zusterorganisatie van Lijbrandt Telecom Nederland B.V. en zorgt voor de aanleg en het onderhoud van glasvezelnetwerken.
Sinds 1997 is Lijbrandt Telecom Nederland B.V. uitgegroeid van 200 abonnees naar 14.000 abonnees. Waarbij de groei nog steeds doorzet.

2.2 Positie binnen Lijbrandt Telecom

Als afstudeerder heb ik een werkplek toegewezen gekregen binnen de ICT afdeling van Lijbrandt Telecom. Mij rust de taak om een rapport en een product op te leveren ten behoeve van mijn afstudeeropdracht. Ik heb geen verplichtingen opgelegd gekregen van de opdrachtgever over het present zijn binnen het gebouw van Lijbrandt Telecom.

Wel heb ik het zogenaamde 'Standaard Personeelsreglement' gekregen van R. Schäfer. In het reglement staat aangegeven wat van de medewerkers van Lijbrandt Telecom wordt verwacht en wat zij van het bedrijf kunnen verwachten.
Ik heb mij tijdens mijn afstudeerperiode aan het reglement gehouden. Om een indruk te krijgen hoe het is in de praktijk heb ik de werktijden van Lijbrandt Telecom gehanteerd. Indien ik om een reden niet aanwezig kon zijn, dan heb ik dit gemeld aan de directieassistente S. Hofstra.

Op afbeelding 2.2.1 'Toegewezen werkplek' ziet u een foto van de werkplek die ik binnen de afdeling ICT toegewezen heb gekregen.

Afbeelding 2.2.1 'Toegewezen werkplek'

3. Opdrachtomschrijving

Lijbrandt Telecom ervaart de huidige telefonie-infrastructuur op het moment als een grote kostenpost. Middels VoIP wil het bedrijf met name de kosten terugbrengen. Op dit gebied ligt dan ook mijn afstudeeropdracht.
De opdrachtomschrijving heb ik meerdere malen aan moeten passen om specifieker aan te geven wat de probleemstelling en doelstelling is.
Naarmate ik me verdiepte in de probleemstelling waar Lijbrandt Telecom mee kampt, was ik steeds meer in staat om de opdrachtomschrijving specifieker te maken.
Ik heb tot 8 april 2005 de tijd gehad om mijn opdrachtomschrijving te specificeren, en zo een definitieve opdrachtomschrijving op te stellen en hier niet meer van af te wijken.

3.1 Probleemstelling

Lijbrandt Telecom beschikt momenteel over de middelen om VoIP toe te passen in een netwerkomgeving, maar heeft nog geen werkende oplossing/toepassing. Dit wil zeggen dat men over de apparatuur beschikt, maar geen samenwerkende opstelling heeft.
Men wil binnen de VoIP toepassing, verschillende functies kunnen beheren zoals kostenregistratie met behulp van software en een voicemail-systeem.

3.2 Doelstelling

De doelstelling van de opdracht is om een VoIP testomgeving op te stellen.
Er zal software moeten worden gemaakt ten behoeve van de ‘kostenregistratie’ en ‘voicemail’ binnen deze VoIP testomgeving, zodat men dit kan beheren.

3.3 Op te leveren producten

Ik zal de volgende producten opleveren aan de opdrachtgever:

1. VoIP testopstelling
2. Software om de gegevens ten behoeve van ‘Kostenregistratie’ te beheren en deze op te slaan in een gegevensopslagstructuur
3. Software om voicemail te beheren

4. Plan van aanpak

In dit hoofdstuk licht ik toe welke richtlijnen ik heb gehanteerd voor het realiseren van dit afstudeerproject. Om duidelijkheid te verschaffen aan de verschillende partijen die betrokken zijn bij mijn afstudeeropdracht beschrijf ik in dit hoofdstuk onder meer de gehanteerde methoden en de ontwikkelingstechnieken.

4.1 Projectorganisatie

Hier volgt een overzicht van betrokken personen en bedrijfsinformatie die deelnemen binnen dit project:

Student
Naam: Yong Bing Hu
Studentnummer: 20024721
Emailadres: y.b.hu@student.hhs.nl
Telefoonnummer: 06-24258881

Opdrachtgever
Naam: Ing. M.J. Schreuder
Emailadres: j.schreuder@lijbrandt-telecom.nl
Telefoonnummer: 023-8911005

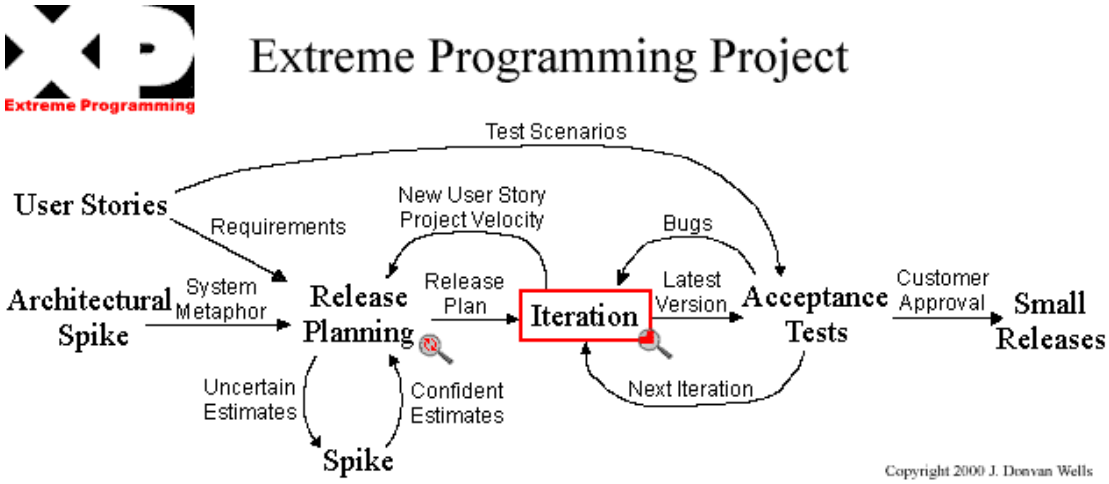
Bedrijf
Naam: Lijbrandt Telecom
Adres: Satellietbaan 20
Postcode: 2181 MH
Plaats: Hillegom
Telefoonnummer: 023-8910000
Fax: 023-8911000

School
Naam: Haagse Hogeschool
Adres: Johanna Westerdijkplein 75
Postcode: 2521 EN
Plaats: Den Haag
Telefoonnummer: 070-4458888
Faxnummer: 070-4458825

De rol van de opdrachtgever binnen dit project is het ondersteunen van de student met betrekking tot het verstrekken van faciliteiten en benodigde informatie.

4.2 Methode

Voor dit softwareontwikkelingtraject zal ik de methode 'Extreme Programming' hanteren. De reden dat ik deze methode heb gekozen is omdat deze zeer geschikt is voor softwareontwikkeling. Dit komt omdat het een iteratieve werkwijze hanteert en zich concentreert op het realiseren van iteraties(softwareonderdelen). Tevens is men bij 'Extreme Programming' niet gebonden aan een specifiek ontwerp dat in het begin van dit traject is opgesteld en niet meer mag afwijken zoals bij de waterval techniek van de methode SDM.
Door deze methode te hanteren is men in staat om fouten, zoals ontwerpfouten, te herstellen om meer aan de klanteisen te kunnen voldoen. De 'Extreme Programming' methode schrijft ook voor om veelvuldig terugkoppeling te vinden met de gebruikers en opdrachtgever om misverstanden op het gebied van communicatie te voorkomen.
Op afbeelding 4.1 'Extreme Programming traject'* vindt u een overzicht van het verloop van het softwareontwikkelingtraject met de methode 'Extreme Programming':



Afbeelding 4.1. 'Extreme Programming' traject

Met behulp van user stories worden 'requirements' vastgesteld, zoals weergegeven in afbeelding 4.1 'Extreme Programming' traject. Met andere woorden: de user stories zijn korte beschrijvingen waarmee de klanteisen worden vastgesteld.
Met behulp van de klanteisen is het mogelijk om een releaseplanning op te stellen. Dit is een planning die het traject verder definieert. Om de releaseplanning aan te vullen kan men spikes/architectural spikes gebruiken. Dit zijn korte experimentele programma's om zo de potentiële oplossingen te achterhalen. De meeste van deze spikes worden veelal niet meer gebruikt in de iteratie zelf, omdat het slechts kleine programma's zijn om te zien of onderdelen van de iteraties realiseerbaar zijn. Het komt erop neer dat het gaat om eerst te testen of het werkt voordat het daadwerkelijk geprogrammeerd wordt voor de iteratie.

Binnen de iteratiefase ga ik aan de slag met het ontwikkelen van de software. Deze iteraties worden getoetst op de werking en functionaliteit door acceptatietests. 'Extreme Programming' biedt de gelegenheid om na het uitvoeren van deze acceptatietests, de bugs binnen de iteratie weg te werken.
Deze iteraties worden uitgebracht als small releases nadat de acceptatietesten zijn goedgekeurd door de opdrachtgever. Een small release is het product van de 'Extreme Programming' traject.

*Afbeelding 4.1 'Extreme Progamming traject' – bronvermelding: http://www.extremeprogramming.org

Tijdens het maken van het plan van aanpak heb ik problemen ondervonden met het opstellen van een planning en het definiëren van de opdrachtomschrijving.
Volgens 'Extreme Programming' moet er in grote lijnen een planning zijn opgesteld en de anticipatie moet minimaal zijn, om zo onnodig werk te voorkomen. Werkzaamheden waarvan niet zeker is of deze überhaupt zullen voorkomen binnen het traject.
Mijn opdrachtgever wilde me inzicht geven in de werkzaamheden van het project met behulp van een workbreakdown structure plan. Dit is een planning dat volledig alle werkzaamheden in kaart brengt. Omdat 'Extreme Programming' adviseert om niet te uitgewerkt te anticiperen, heb ik hem geprobeerd te vertellen dat het tegen de principes van 'Extreme Programming' in gaat. Ik heb mijn planning volgens een work-breakdown structure plan opgesteld en heb wel veel meer inzicht kunnen krijgen op het gebied van de werkzaamheden die mij te wachten stond.
Na verloop van tijd heb ik samen met mijn opdrachtgever afspraken gemaakt met betrekking tot mijn werkwijze en de daar bijhorende verwachtingen. Beide partijen hebben hierin moeten toegeven.
In het plan van aanpak vind u een momentopname van de planning die ik op had gesteld tijdens het maken van het plan van aanpak.

4.3 Technieken

Voor het modelleren van objectmodellen zal ik gebruik maken van de techniek Unified Modeling Language(UML). De reden dat ik de UML techniek hanteer is dat ik hierin ben onderwezen en praktische ervaringen heb in het maken van softwareontwerpen met UML, wat het ontwerpproces zal versnellen.
Door UML te hanteren wordt een duidelijk beeld weergegeven over de uitgangsituatie en de gewenste situatie. UML doet dit met behulp van bijvoorbeeld use cases, use case diagrammen, beschrijvingen, toestandsdiagrammen, etc.
Ik zal tijdens dit project niet alle onderdelen van UML gaan gebruiken, omdat ik hoogstwaarschijnlijk in een procedurele taal ga programmeren en niet objectgeoriënteerd. Het verschil tussen procedureel en objectgeoriënteerd is dat mijn software een begin en een eind zal hebben, in tegenstelling tot objectgeoriënteerd talen waarbij het mogelijk is om bijvoorbeeld te overerven van andere objecten.
Het zal voor mij een nieuwe ervaring zijn om in een procedurele taal te programmeren, alhoewel ik wel eens eerder in aanraking ben gekomen met de procedurele taal 'Perl' maar slechts beperkte kennis hierover heb opgedaan.

4.4 Risicofactoren

Mijn geringe voorkennis over telefonie kan wellicht problemen opleveren op het gebied van tijdsbesteding aan onderzoek. Op dit moment beschik ik slechts heel weinig kennis over de infrastructuur en functionaliteiten van telefooncentrales, dus zal ik waarschijnlijk veel tijd moeten besteden aan de activiteiten die zich voordoen in de definitiefase van een project.
De kans bestaat dat ik niet in staat ben om alle deelproducten op tijd op te leveren vanwege de diepgang en omvang van dit project. Indien hier sprake van is, zal ik overleg plegen met de opdrachtgever Ing. M.J. Schreuder en mijn begeleidend docent om tot overeenstemming te komen wat de 'op te leveren producten' zullen zijn.

Ik zal met behulp van het MoSCoW principe prioriteiten stellen aan de eisen, die de opdrachtgever stelt. Zo worden de functionaliteiten die de opdrachtgever relevant vindt het eerst verwerkt binnen de applicatie(s).
Zo worden de functionaliteiten binnen de software gerangschikt op prioriteit. Door dit te hanteren, probeer ik zo veel mogelijk te voldoen aan de klanteisen en wensen.
Dit principe zult u verder in dit document terugvinden in hoofdstuk 10 (Releaseplan).

4.5 Faciliteiten

Om dit project te realiseren zal ik diverse faciliteiten nodig hebben. Deze faciliteiten heb ik opgesomd en opgedeeld in drie verschillende onderdelen. Dit zijn:

Benodigde software

- Microsoft Word
- Microsoft Excel
- Microsoft Project
- Microsoft Visio
- Fedora Core 3 Linux
- Perl 5 of PHP
- Homesite 5.5
- Topstyle Lite 3.10

Benodigde hardware

- 1 Switch
- 1 Asterisk PBX
- 1 Digium Wildcard T110P - ISDN 30(E1) kaart
- 1 Digium IAXy S100L
- 1 Werkstation
- 1 Server
- 1 Analoge telefoon

Beschikbare rapporten

- Gepubliceerde artikelen op het internet
- Handleidingen van hardware

Tevens wil ik vermelden dat ik van alle faciliteiten gebruik heb kunnen maken binnen Lijbrandt Telecom. Er werd mij duidelijk gemaakt dat ik het slechts hoef te vragen indien ik iets nodig had. Alles wat ik nodig had, werd verzorgd op het gebied van software, hardware en het gebruik van faciliteiten zoals een beveiligingspas en het gebruik van het kopieerapparaat.

4.6 Kwaliteitseisen

Voor dit afstudeerproject zal ik de SMART kwaliteitseisen hanteren. SMART staat voor: Specifiek, Meetbaar, Acceptabel, Realistisch en Tijdsgebonden. Middels deze techniek wil ik de kwaliteit van het project waarborgen. Hieronder licht ik toe waar deze kwaliteitseisen voor staan:

1. Specifiek – het project moet specifiek beschreven zijn.
2. Meetbaar - men moet duidelijke meetbare doelen stellen.
3. Acceptabel – Deze doelen moeten voor de deelnemers haalbaar zijn en door de deelnemers gedraagbaar zijn (leuk en nuttig gevonden worden).
4. Realistisch – het project moet haalbaar zijn.
5. Tijdgebonden – er moet een duidelijke tijdsplanning afgesproken worden om te voorkomen dat het project te lang doorloopt of niet wordt afgesloten

4.7 Versiebeheer

Een ander belangrijk punt binnen de werkwijze van mijn afstudeeropdracht is het hanteren van versiebeheer. Tijdens mijn opleiding heb ik kennisgemaakt met CVS(concurrent version system) en merkte dat dit een bijzonder gemakkelijke manier was om versiebeheer toe te passen op projecten zoals deze.
Hier bij Lijbrandt Telecom maakt men gebruik van Subversion. Dit is een open source versiebeheeroplossing dat men kan vergelijken met CVS. Omdat ik nog geen ervaring met deze versiebeheeroplossing, heb ik wat aanwijzingen gekregen op het gebied van de heer J. Vermeer die hoofd van afdeling ICT is.

Subversion werkt ook met een client server systeem zoals CVS, maar biedt ook de mogelijkheid om lokaal versies te beheren. Dit is uitermate geschikt voor kleine projecten waar men alleen aan werkt, zoals in mijn geval.
Omdat ik nog nooit met TurtoiseSVN had gewerkt, heb ik een uitvoerige beschrijving gemaakt over het gebruik ervan. Deze beschrijving kunt u terugvinden in de bijlage van dit document.

4.8 Belangenkwestie

De opdrachtomschrijving was initieel te ongespecificeerd om een beeld te vormen van wat ik uiteindelijk zou gaan moeten doen. De opdrachtgever dacht zelf aan dienstontwikkeling van VoIP binnen Lijbrandt Telecom, wat meer de architectuur/beheer kant op zou gaan.
Omdat ik begon te twijfelen over wat de opdrachtgever van mij verwachtte, heb ik advies gevraagd aan mijn begeleidend docent J.P. van Dinter en een goede kennis die voor een andere telecom bedrijf heeft gewerkt en veel van het infrastructuur binnen een telecombedrijf af wist. De reden dat ik vroegtijdig aan de bel had getrokken, was om situaties te voorkomen waarin conflicten zouden kunnen ontstaan tussen de wensen van de opdrachtgever en mijn school.
Zij hadden mij geadviseerd om mijn opdrachtgever een keuze te laten maken tussen een dienstontwikkelingsproject (waarbij het opzetten van een VoIP toepassing centraal staat) en een softwareontwikkelingproject (waarbij het ontwikkelen van software ten behoeve van de kostenregistratie en voicemail centraal staan) zodat mijn opdracht beoordeelbaar is voor school.

Mijn interpretatie van de opdracht was dat ik mij voornamelijk zou richten op het ontwikkelen van software ten behoeve van de kostenregistratie en voicemail binnen VoIP.
Om onduidelijkheden weg te nemen bij de opdrachtgever, heb ik een herziende versie gemaakt van het plan van aanpak. In deze herziende versie van het plan van aanpak heb ik de opdrachtomschrijving specifieker beschreven om zo verwarring bij de opdrachtgever te voorkomen.
De opdrachtgever heeft ingestemd met het uitvoeren van een softwareontwikkelingsproject en tot uiterlijk 28 februari 2005 bezig te zijn met het opstellen van een VoIP opstelling.

4.9 Globale planning

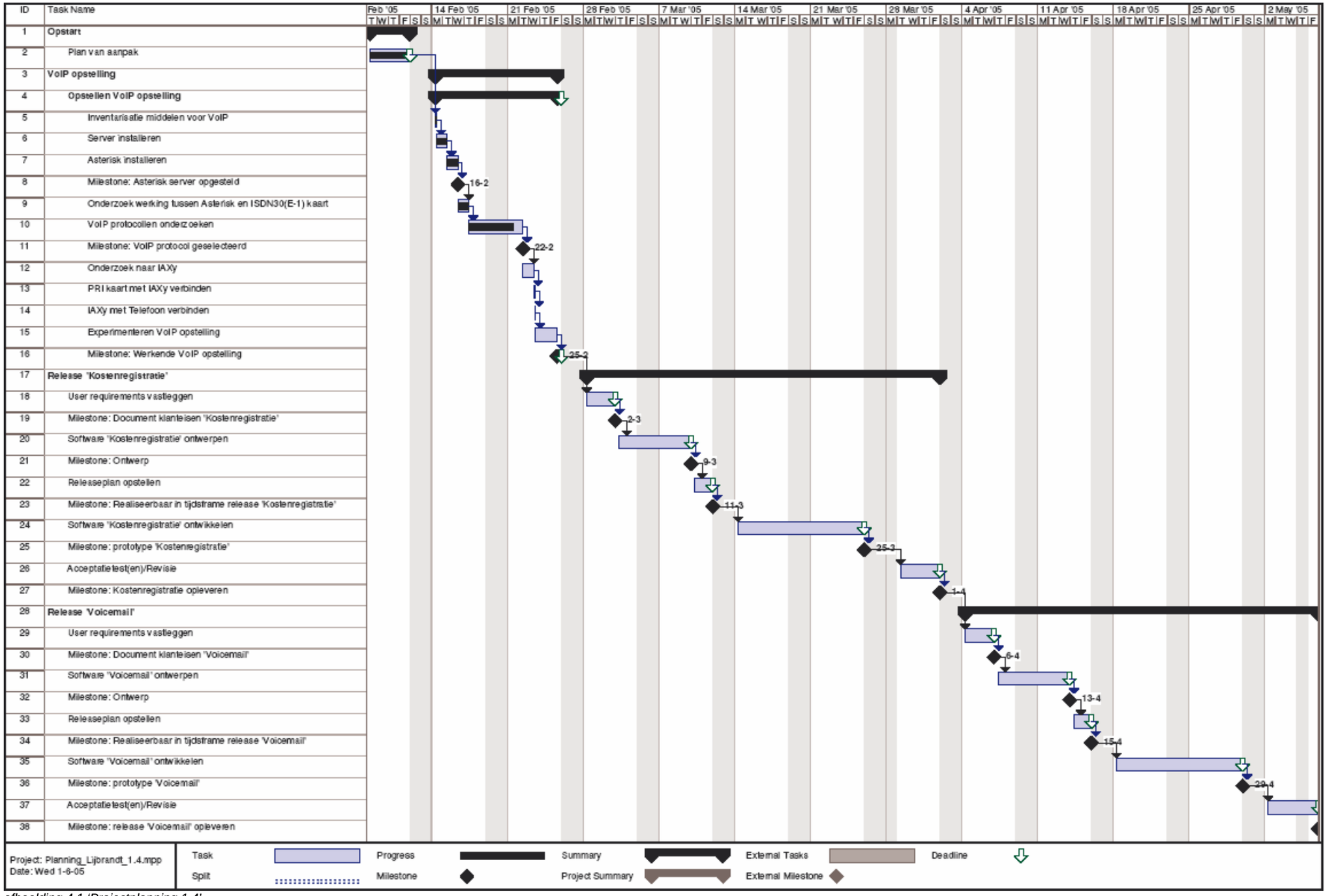
De globale planning voor dit project zal ik maken in de vorm van een gantt chart. Deze planning is gebaseerd op doorlooptijd. Dit wil zeggen dat de duratie van het proces verspreid is over een opgegeven tijd. Deze globale planning heb ik opgesteld om een indruk te geven waarmee ik bezig zal zijn en een ruwe schatting van hoeveel tijd ik in ieder proces ga steken.
De gantt chart heb ik opgesteld in het programma ‘Microsoft Project 2003’. De reden dat ik het in deze applicatie heb gemaakt is omdat ik al meerdere projectplanningen in vorm van gantt charts heb gemaakt en hierdoor enige ervaring heb met het programma.
Zoals ik in paragraaf 4.6 Belangenkwestie heb beschreven is het duidelijk geworden dat er een belangenkwestie plaatsvond tijdens dit project. Ik heb daarom ook meerdere projectplanningen moeten opstellen voordat er overeenstemming was tussen de opdrachtgever en mijzelf die de richtlijnen van ‘Extreme Programming’ probeerde te hanteren.

Ik heb in versie 1.4 van mijn planning geprobeerd om te voldoen aan de eisen die de opdrachtgever stelde, over hoe een projectplanning opgesteld moet worden.
Deze was in detail uitgewerkt, in tegenstelling tot wat ’Extreme Programming’ voorschrijft.
Op afbeelding 4.1 ‘Projectplanning 1.4’ vindt u de planning die de opdrachtgever als basis wilde gebruiken.
In de vorige paragraaf heb ik duidelijk gemaakt dat er een belangenkwestie speelde binnen dit project. Om deze reden ben ik na overleg met een goede kennis; Ing. A. van der Molen en mijn begeleidend docent J.P. van Dinter begonnen met het wijzigen van mijn planning.

In planning 2.2 heb ik mijn werkzaamheden op hoger niveau beschreven zodat ik in grotere lijnen aanduidt waarmee ik me bezig ga houden en niet teveel anticipeer op de werkzaamheden.
Op afbeelding 4.2 ‘Projectplanning 2.2’ vindt u mijn herziende versie waarin ik de werkzaamheden op een hoger niveau beschrijf.

Tot slot, heb ik een derde versie van een projectplanning opgesteld. Ik heb de planning aangepast omdat het steeds duidelijker werd in hoeverre dit project realiseerbaar was binnen de toegewezen tijd.
Omdat het meer werk bevatte dan gedacht, zijn de opdrachtgever en ik overeengekomen om de iteratie ‘Voicemail’ te doen vervallen. Op afbeelding 4.3 ‘Projectplanning 3.1’ vind u laatste versie van de planning.

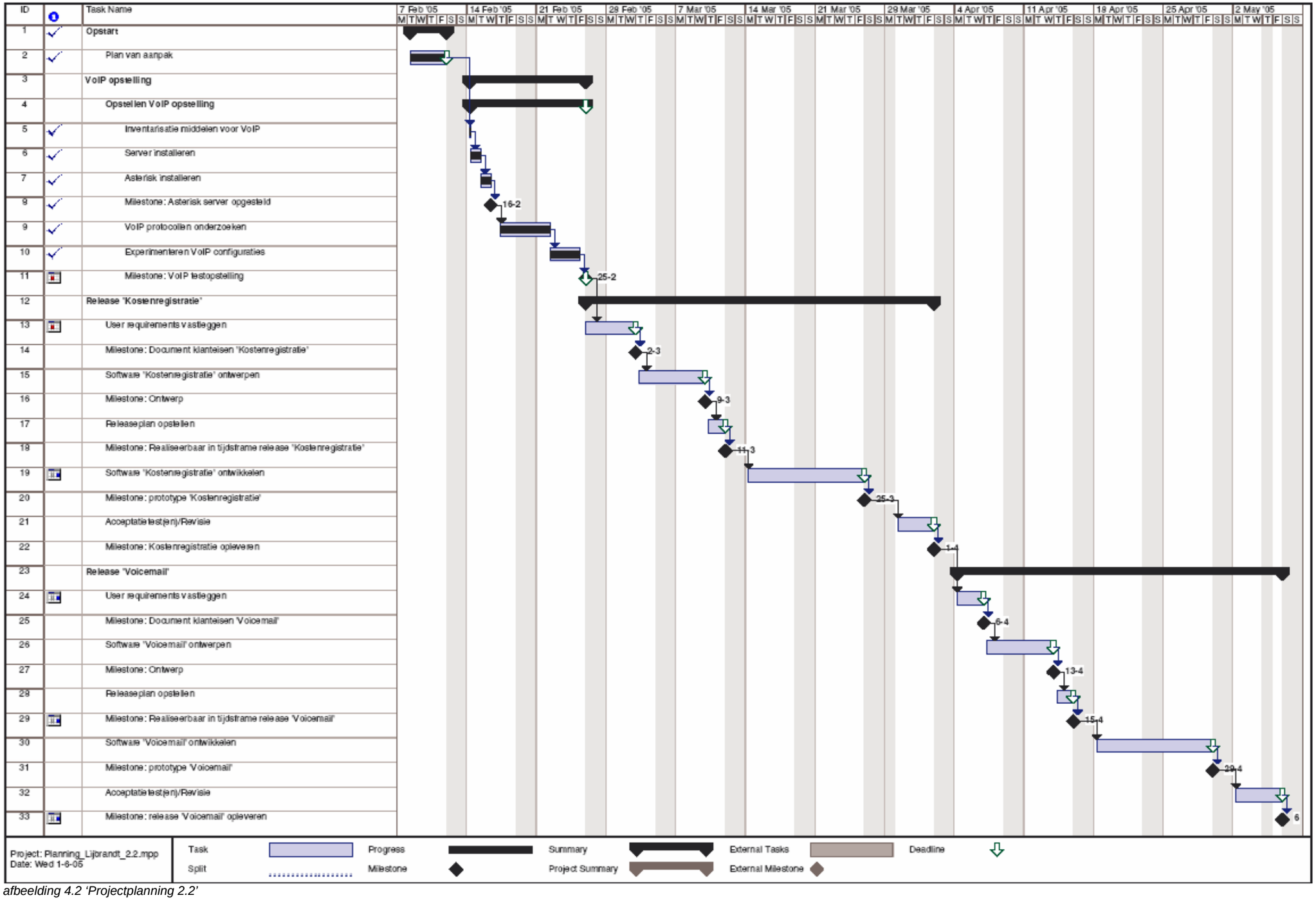
De planning hieronder is een momentopname van planning 1.4 waarin de werkzaamheden in detail zijn uitgewerkt volgens een workbreakdown structure plan.



afbeelding 4.1 'Projectplanning 1.4'

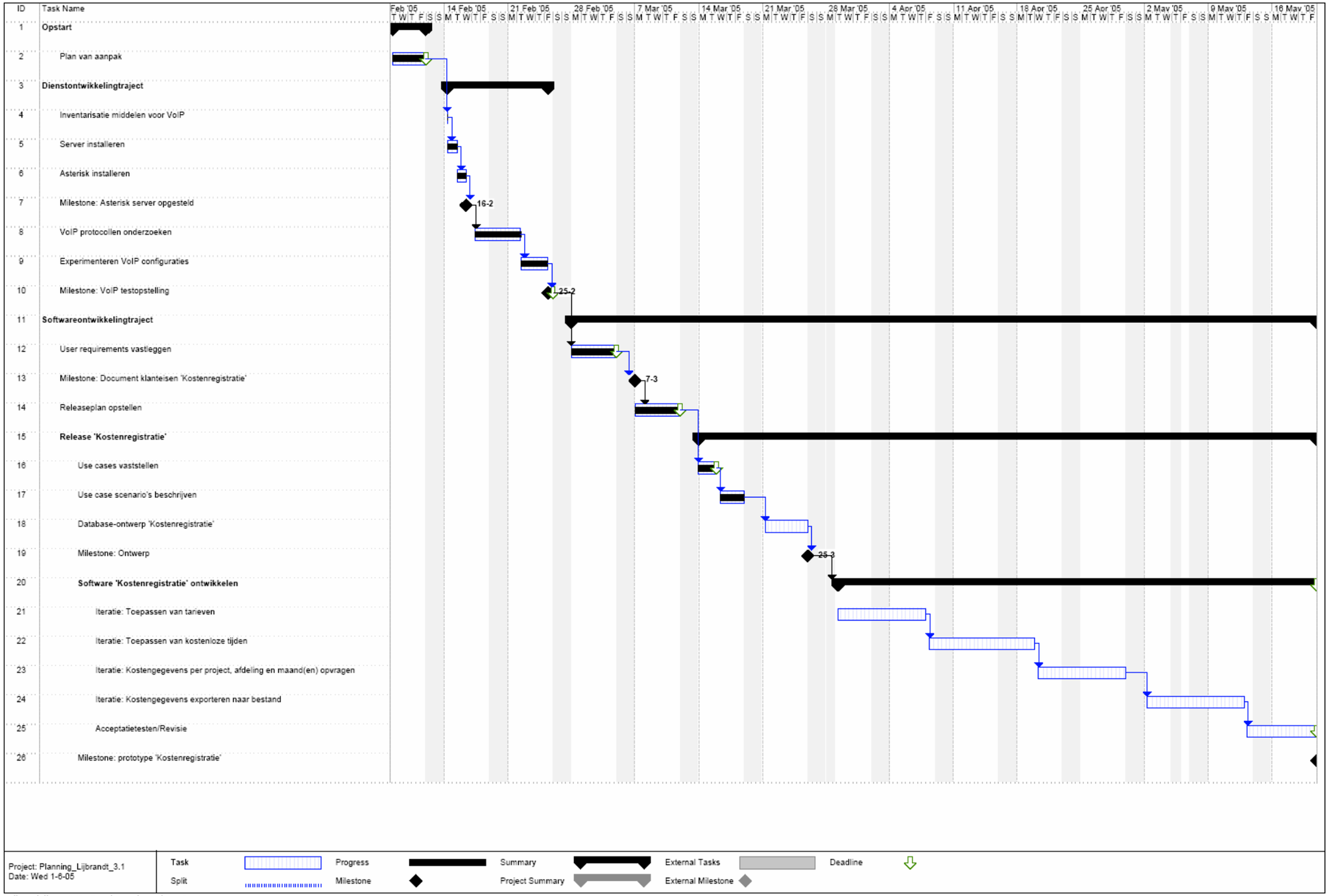
De planning hieronder is een momentopname van planning 1.4 waarin de werkzaamheden in op hoger niveau zijn uitgewerkt volgens de methode ‘Extreme Programming’.

ID	Task Name
1	Opstart
2	Plan van aanpak
3	VoIP opstelling
4	Opstellen VoIP opstelling
5	Inventarisatie middelen voor VoIP
6	Server installeren
7	Asterisk installeren
9	Milestone: Asterisk server opgesteld
9	VoIP protocollen onderzoeken
10	Experimenteren VoIP configuraties
11	Milestone: VoIP testopstelling
12	Release "Kostenregistratie"
13	User requirements vastleggen
14	Milestone: Document klanteisen "Kostenregistratie"
15	Software "Kostenregistratie" ontwerpen
16	Milestone: Ontwerp
17	Releaseplan opstellen
18	Milestone: Realiseerbaar in tijdsframe release "Kostenregistratie"
19	Software "Kostenregistratie" ontwikkelen
20	Milestone: prototype "Kostenregistratie"
21	Acceptatietest(en)/Revisie
22	Milestone: Kostenregistratie opleveren
23	Release "Voicemail"
24	User requirements vastleggen
25	Milestone: Document klanteisen "Voicemail"
26	Software "Voicemail" ontwerpen
27	Milestone: Ontwerp
28	Releaseplan opstellen
29	Milestone: Realiseerbaar in tijdsframe release "Voicemail"
30	Software "Voicemail" ontwikkelen
31	Milestone: prototype "Voicemail"
32	Acceptatietest(en)/Revisie
33	Milestone: release "Voicemail" opleveren



afbeelding 4.2 ‘Projectplanning 2.2’

De planning hieronder is een momentopname van planning 3.1 waarin de iteratie ‘voicemail’ komt ter vervallen.



afbeelding 4.3 ‘Projectplanning 3.1’

5. Uitgangssituatie

In dit hoofdstuk wordt de uitgangssituatie van Lijbrandt Telecom schematisch in kaart gebracht. Middels schema's worden de volgende processen in chronologische volgorde weergegeven: 'Telefoneren binnen Lijbrandt Telecom' en 'Het registeren en verrekenen van gespreksgegevens'.
Omdat de genoemde processen afhankelijk zijn van de infrastructuur binnen Lijbrandt Telecom, is deze ook opgenomen in dit hoofdstuk. Door VoIP toe te passen zullen er in de infrastructuur immers veranderingen plaatsvinden.
Het in kaart brengen van de infrastructuur en de processtromen binnen Lijbrandt Telecom is essentieel voor het ontwerpen en ontwikkelen van de software.

Het concept van telefonie binnen Lijbrandt Telecom is diensten aanbieden binnen het netwerk en via Priority Telecom deel uitmaken van PSTN[1]. Dit bedrijf zorgt ervoor dat het mogelijk is om gesprekken te voeren buiten het netwerk van Lijbrandt Telecom. De gesprekken die worden gevoerd naar individuen buiten het netwerk van Lijbrandt Telecom worden verrekent door Priority Telecom. Dit gaat met behulp van een ISDN 30 (E1) kaart in de Tenovis PABX telefooncentrale.
Lijbrandt Telecom verrekent geen kosten voor het bellen in het weekend voor gesprekken met andere Lijbrandt Telecom abonnees die zich in dezelfde regio als de beller bevinden. De bron van inkomsten voor Lijbrandt Telecom ligt dan ook op het verrekenen van de maandelijkse abonnementskosten en niet op het verrekenen van gesprekskosten.

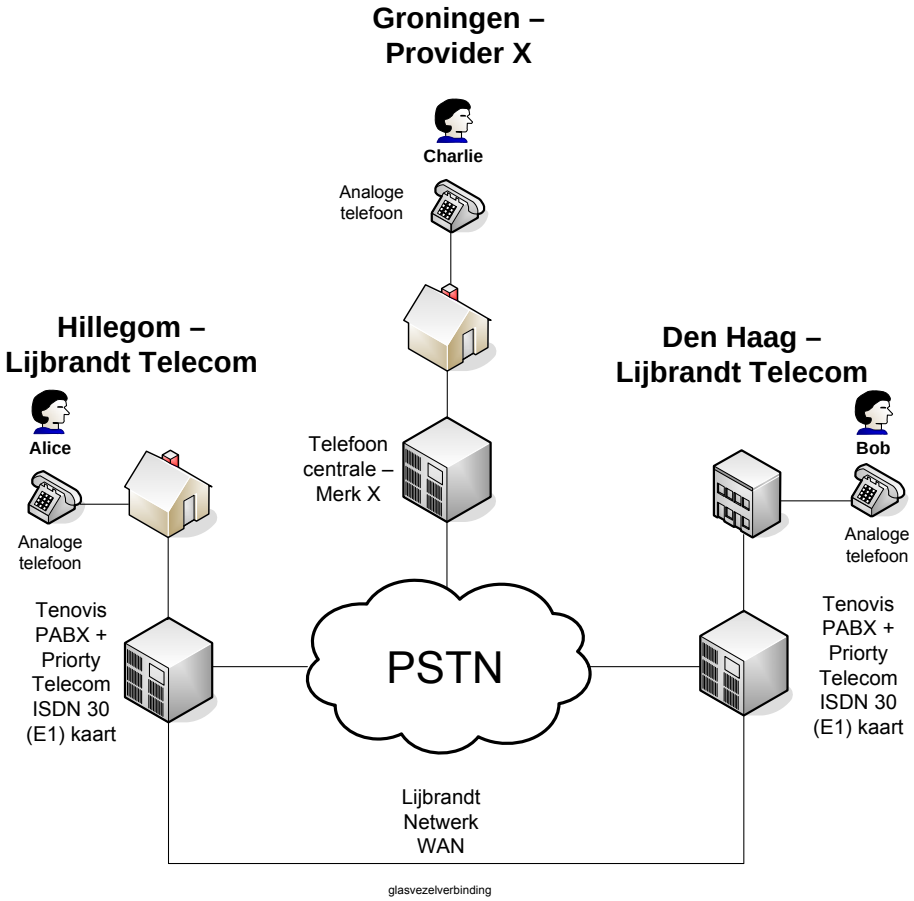
5.1 Infrastructuur

De infrastructuur van Lijbrandt Telecom bestaat momenteel uit een POTS-opstelling en heeft (nog) geen vorm van VoIP geïmplementeerd. Dit houdt in dat het netwerk van Lijbrandt Telecom momenteel alleen telefonie aanbiedt door middel van analoge telefonie via koperdraden. Er worden geen diensten aangeboden zoals ISDN
Lijbrandt Telecom beschikt momenteel over 25 telefooncentrales die verspreid staan op verschillende locaties waar zich een groot aantal abonnees bevindt.
Iedere locatie wordt aangeduid als een 'project' binnen de terminologie van Lijbrandt Telecom. In ieder project staat een Tenovis PABX[2] telefooncentrale, die het mogelijk maakt om te bellen binnen het netwerk van Lijbrandt Telecom en naar de 'buitenwereld' PSTN, in dit geval met behulp van de provider Priority Telecom.
Tussen de verschillende locaties binnen het netwerk van Lijbrandt Telecom worden glasvezelverbindingen gebruikt.

De aanschafkosten van de Tenovis PABX telefooncentrales zijn hoog (€100.000 per stuk). Omdat er 25 centrales zijn, liggen de kosten van deze telefooncentrales gezamenlijk op circa €2.500.000. Lijbrandt Telecom zal verschillende projecten starten dit jaar dat als gevolg heeft dat het aantal abonnees groeit.
Om kosten te drukken van het aanschaf van Tenovis PABX telefooncentrales, is Lijbrandt Telecom momenteel op zoek naar een alternatieve oplossing.

[1] PSTN – Public Switched Telephone Network, internationaal telefoon systeem gebaseerd op koperdraden die de techniek ISDN ondersteund.
[2] PABX – Private Automatic Branch Exchange, een geautomatiseerde telefooncentrale binnen een besloten netwerk

Op afbeelding 5.1 'Proces - telefoneren binnen Lijbrandt Telecom' schets ik een situatie in de vorm van een schema. In deze schets zal ik uitleggen wat er gebeurt als Alice, abonnee van Lijbrandt Telecom, naar Bob belt.



Afbeelding 5.1 'Proces - telefoneren binnen Lijbrandt Telecom'

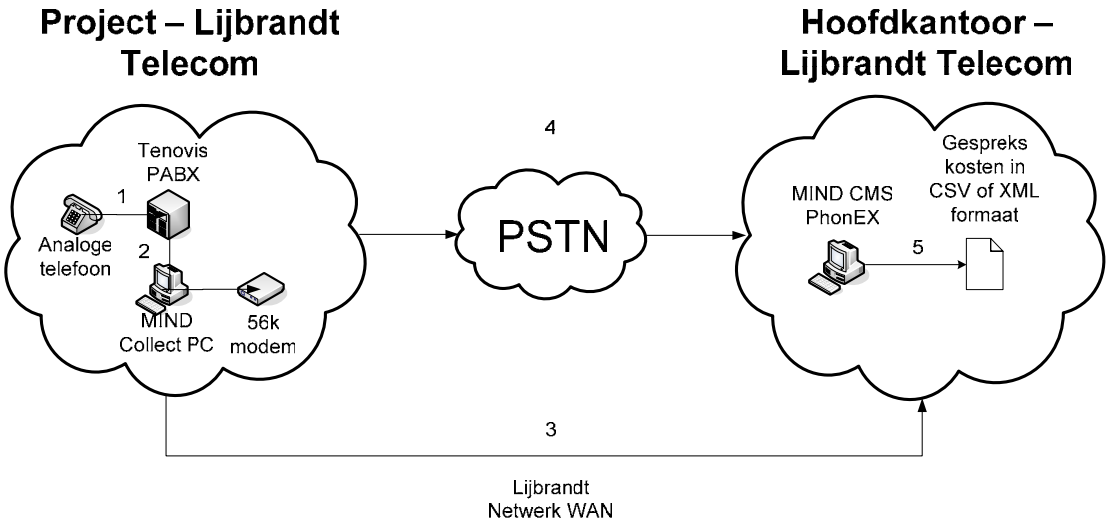
Onderstaand schets ik twee scenario's waarin de processen worden verduidelijkt van [1] telefoneren binnen het netwerk van Lijbrandt Telecom met POTS en [2] telefoneren vanuit het netwerk van Lijbrandt Telecom naar buiten met POTS.

Scenario 1 – Alice belt naar Bob
Alice (wonend te Hillegom en abonnee van Lijbrandt Telecom) belt naar haar vriend Bob (wonend te Den Haag en tevens abonnee van Lijbrandt Telecom). Het gesprek verloopt middels de Tenovis PABX telefooncentrale van haar project(locatie) die via het netwerk van Lijbrandt Telecom verbindt naar de Tenovis PABX telefooncentrale van het project waar Bobs verbinding onder valt.

Scenario 2 – Alice belt naar Charlie
Alice (wonend te Hillegom en abonnee van Lijbrandt Telecom) belt naar haar broer Charlie (wonend te Groningen en geen abonnee van Lijbrandt Telecom). Het gesprek verloopt middels de Priority ISDN 30 (E1) kaart binnen de Tenovis PABX telefooncentrale van haar project, die via de PSTN aankomt bij een PABX telefooncentrale van de provider waar Charlie abonnee van is.

5.2 Het registreren en verrekenen van gesprekgegevens

In deze paragraaf zal ik toelichten hoe het proces verloopt om gespreksgegevens te registreren en vervolgens de gesprekskosten te verrekenen. De term die binnen Lijbrandt Telecom wordt gebruikt voor het verrekenen van gesprekskosten is 'Kostenregistratie'. Voor het registreren en verrekenen van de gesprekskosten wordt gebruik gemaakt van MIND CMS PhonEx. Dit is een softwarepakket dat is ontwikkeld door het bedrijf MAF unlimited te Amsterdam die gespecialiseerd is in het ontwikkelen van billing-systems (systemen die met behulp van gegevens kosten verrekenen). MIND CMS PhonEx maakt gebruik van licenties in de vorm van dongles[3] om toegang te verlenen voor het gebruik van de software. Per MIND Collect PC en MIND CMS PhonEx moet er een licentie worden aangeschaft. De licenties van de software dienen eenmalig te worden aangeschaft en de kosten hiervan zijn €3.000 per licentie.
De MIND software wordt gebruikt voor het registreren en verrekenen van de gesprekskosten. Per Tenovis PABX telefooncentrale dient men een MIND Collect PC aan te koppelen zodat deze de CDR's[4] (Call Detail Records) registreert.
Op afbeelding 5.2 'Schema - uitgangssituatie' ziet u een schema van de levenscyclus van het registreren van CDR's tot het verrekenen van de gesprekskosten:



Afbeelding 5.2 'Schema – uitgangssituatie'

Hieronder leg ik het proces uit van wat er in chronologische volgorde gebeurt:

1. Een Lijbrandt Telecom abonnee heeft een telefoongesprek gevoerd vanuit een project en de Tenovis PABX verwerkt de gespreksgegevens hiervan
2. De MIND Collect PC luistert en verzamelt de CDR's die de Tenovis centrale verwerkt
3. De MIND Collect PC verstuurd periodiek de CDR gegevens naar het MIND CMS PhonEX systeem, gelegen op het hoofdkantoor, via het netwerk van Lijbrandt Telecom.
4. Wanneer het niet mogelijk is om de CDR gegevens via het netwerk van Lijbrandt Telecom te versturen, dan zal de MIND Collect PC de CDR gegevens via een 56k analoge modem over het PSTN versturen

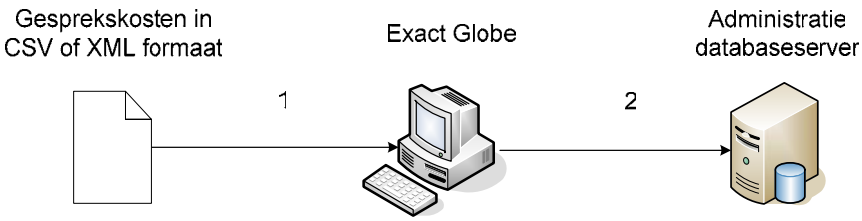
[3] Dongles – Softwarelicentie in fysieke vorm, deze zijn vereist voordat de software pas functioneert
[4] CDR – Call Detail Record, gegevens van het telefoongesprek

5. De MIND CMS PhonEX berekent de gesprekskosten met behulp van de verbruiksgegevens en genereert bestanden in het bestandsformaat CSV[5] of XML[6]

5.3 Facturering

Voor het factureren van de gesprekskosten wordt het softwarepakket Exact Globe gebruikt. Dit softwarepakket biedt oplossingen op het gebied van: Financiële administratie, logistieke administratie, Project Management en Personeelsmanagement & Salarisadministratie. Binnen Lijbrandt Telecom wordt het softwarepakket gebruikt voor financiële administratie.
De abonnementskosten van ieder project worden maandelijks verrekend op de eerste dag van de maand. De abonnees die zich aanmelden na de 15e van een maand worden niet verrekend op de abonnementskosten van de maand waarin ze zich hebben aangemeld. De abonnees die zich tussen de 1e en de 16e van de maand hebben aangemeld worden wel verrekend met de maandelijkse abonnementskosten. Dit gebeurt binnen Exact Globe. Men voert de periode in waar men een factuur van wilt uitdraaien. Vervolgens worden de facturen van het geselecteerde project en uitgedraaid.

Momenteel worden CSV en XML bestanden met daarin de gesprekskosten van abonnees gebruikt voor het factureren. Deze worden ingelezen door Exact Globe en vervolgens verwerkt binnen de administratiedatabase.
Op afbeelding 5.3.1 'Flowchart – facturering uitgangsituatie' ziet u in een schema weergegeven wat er precies gebeurt nadat de gesprekskosten zijn verrekend door MIND CMS PhonEx en in CSV of XML bestandsformaat zijn gegenereerd.



Afbeelding 5.3.1 'Schema – factureren uitgangsituatie'

Hieronder zal ik de flowchart nader toelichten aan de hand van korte beschrijvingen.

1. CSV of XML bestand wordt ingelezen met Exact Globe
2. Exact Globe verwerkt deze gegevens in de Administratie database-server

5.4 Tarieven

Momenteel gelden er verschillende tarieven voor verschillende dagdelen. De tarieven hebben bijvoorbeeld een nachttarief van 00:00:00 tot 06:59:59, een daltarief van 07:00:00 tot 15:59:59, een piektarief van 16:00:00 tot 21:59:59 en tot slot een avondtarief van 22:00:00 tot 23:59:59. Al deze tarieven zijn in te stellen met MIND CMS PhonEx.

[5] CSV – Comma Seperated Value, formaat die veelal wordt gebruikt voor databases en spreadsheets
[6] XML – eXtensible Markup Language, gegevensopslagstructuur dat wereldwijd wordt gebruikt in verschillende toepassingen

6. Klanteisen

Voor het achterhalen van de klanteisen heb ik volgens de methode ‘Extreme Programming’ gewerkt en de opdrachtgever een reeks van user stories laten opstellen.

User stories zijn korte beschrijvingen van functionaliteiten van wat het systeem moet gaan doen. Deze beschrijvingen zijn geschreven in klantjargon.
De software die ik zal gaan maken wordt ontworpen met de UML technieken; Use case diagram en Use case scenario’s.
Omdat mijn opdrachtgever zelf al technisch is aangelegd, bevatten sommige user stories termen die mij nog niet bekend waren. Het was voor mij om deze reden dan ook moeilijk om zoveel mogelijk technische termen te ontwijken omdat mijn opdrachtgever deze user stories in klantjargon heeft opgesteld.

Voor de software ‘Kostenregistratie binnen VoIP’, hebben we de volgende user stories kunnen achterhalen:

- Het software systeem moet gespreksgegevens kunnen berekenen.
- Men moet aan kunnen geven wanneer men wel en wanneer men geen kosten verrekend.
- Het systeem moet verschillende tarieven kunnen hanteren voor regionaal, buiten de regio, internationaal, mobiele en 0900 informatie- nummers.
- De gegevens moeten met de applicatie Exact Globe ingelezen kunnen worden.
- De tarieven moeten per project en per maand ingesteld worden.
- De tarieven moeten verwijderd kunnen worden.
- Het instellen van tarieven moet toegankelijk zijn vanaf iedere locatie.

In het verloop van het traject merkte ik op dat er geen specifieke gegevens duidelijk zijn geworden over de metingen. De oorzaak van dit probleem is dat de omvang en diepgang van het softwareontwikkelingstraject naar mijn mening groot zijn. Dit leidde tot het gevolg dat ik maar geleidelijk informatie heb weten te vergaren en pas tegen problemen aan ben gelopen op het moment dat ik informatie miste.
Een voorbeeld hiervan is de duur van het proces om de gespreksgegevens te berekenen en weer te geven. Nadat dit duidelijk was geworden heb ik, in overleg met de opdrachtgever, de klanteisen aangevuld.
Hieronder de gewijzigde/additionele klanteisen die pas in later stadium aan bod kwamen:

- De software systeem moet sneller de gesprekskosten kunnen berekenen dan met MIND CMS PhonEx.

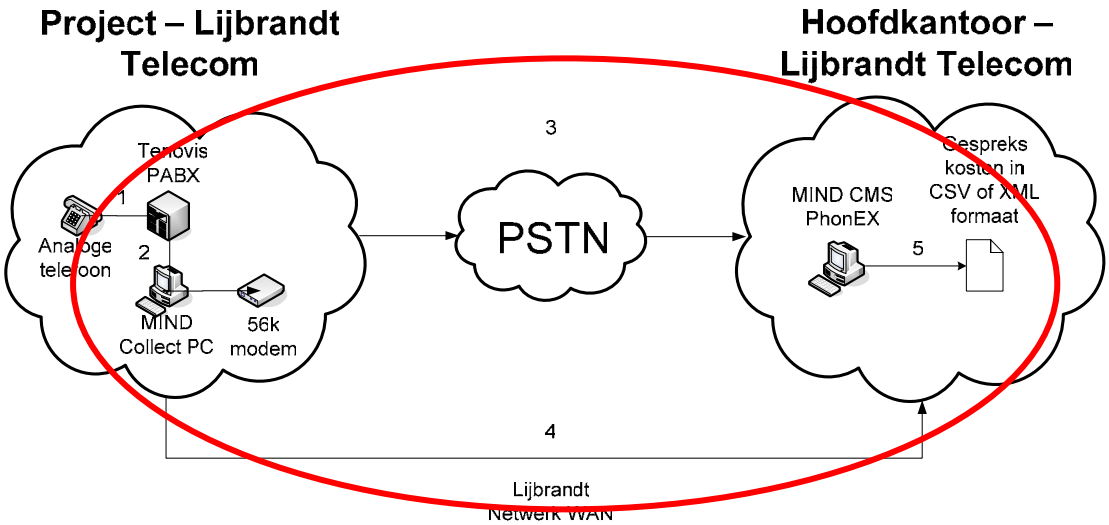
Deze wijziging komt later aan bod in dit document en ook zal het duidelijk worden waarom ik deze wijziging heb moeten invoeren.

7 Gewenste situatie

Na het vaststellen van de user stories is het duidelijk geworden wat de opdrachtgever wilt. De gewenste situatie van infrastructuur en processtroom zal ik, net als de uitgangssituatie, in kaart brengen met behulp van tekeningen.
In mijn afstudeeropdracht zal ik me concentreren op een alternatieve oplossing vinden voor het gebruik van de Tenovis PABX telefooncentrales.

Een alternatieve een oplossing zal ik aanbieden met Asterisk PBX software. Deze software heeft in open-source broncode en is geheel gratis. Door over te gaan op deze oplossing is Lijbrandt Telecom niet meer genoodzaakt deze Tenovis PABX telefooncentrales aan te schaffen, maar slechts een server voor de Asterisk PBX software beschikbaar moet stellen. De kosten van een Asterisk PBX server liggen rond de €1.000. Voor een verbinding buiten het netwerk van Lijbrandt Telecom is voor deze Asterisk PBX server een ISDN 30 E1 kaart beschikbaar. De kosten van deze ISDN 30 kaart is circa $500, omgerekend €407.70. Indien Lijbrandt Telecom voor deze alternatieve oplossing kiest, dan kan dit (maar) liefst €98.592.30 schelen in vergelijking met het aanschaffen van de Tenovis PABX telefooncentrale. Het prijsverschil tussen de aanschafkosten is de reden om over te stappen naar deze alternatieve oplossing.

Ik zal mij voornamelijk concentreren op het schrijven van software ten behoeve van het verrekenen van de gesprekskosten. Dit moet de processen gaan veranderen binnen het verrekenen van de gesprekskosten en het verwerken als een outputbestand in CSV of XML vorm.
Op afbeelding 7.1 ‘Scope van het project’ geef ik met een rode cirkel weer waarop ik me zal concentreren tijdens dit project.



Afbeelding 7.1 ’Scope van het project’

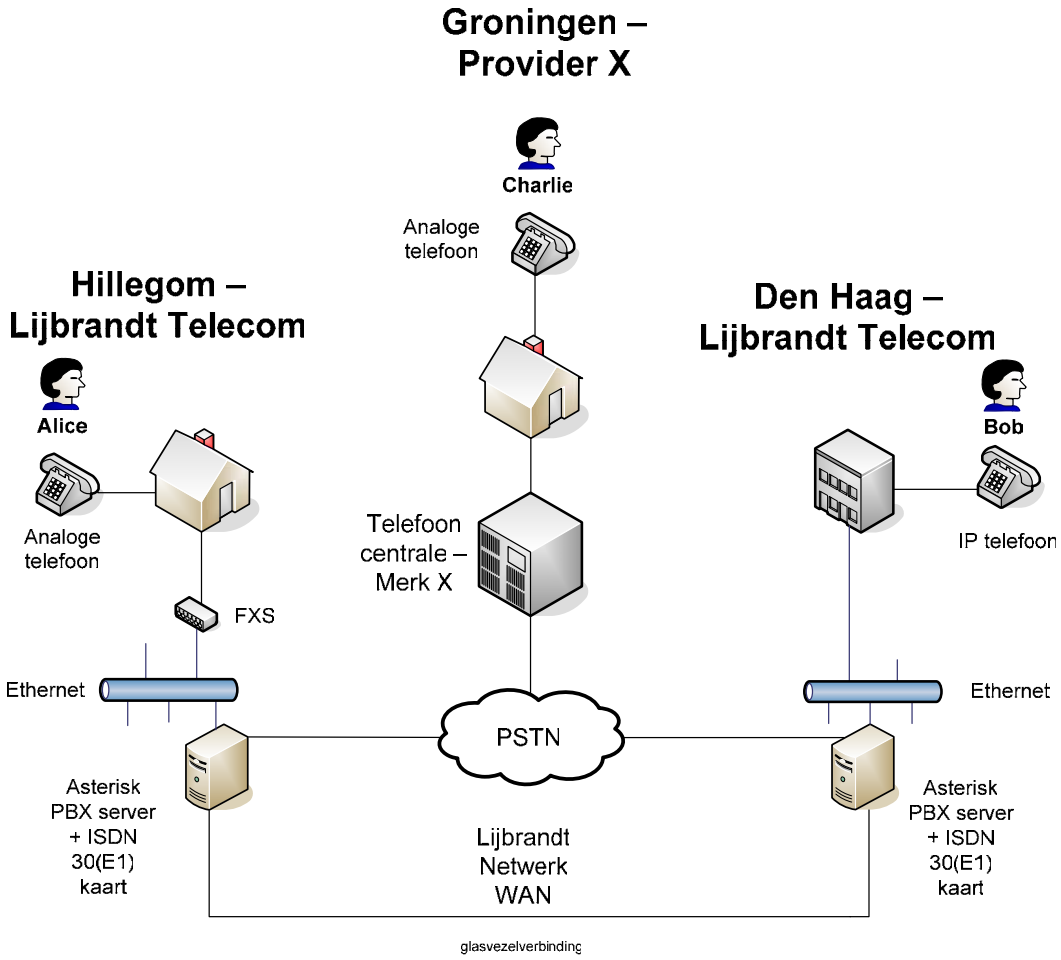
Het gaat voornamelijk om het gebruik van de Tenovis PABX, MIND CMS PhonEX, het versturen/ophalen van de gegevens vanuit het hoofdkantoor en het verrekenen van de gesprekskosten en deze in het bestandsformaat CSV of XML aan te bieden.

Om de software te kunnen ontwikkelen heb ik een VoIP testomgeving nodig om acceptatietesten uit te voeren. Daarom zal ik naast het ontwikkelen van de software ook een VoIP testomgeving opstellen.

Tot slot zal ik me concentreren op het ontwikkelen van software ten behoeve van een voicemail-systeem binnen VoIP.

7.1 Infrastructuur

Zoals eerder vermeld, wordt de infrastructuur gewijzigd door VoIP toe te passen. In de gewenste situatie worden alle Tenovis PABX centrales vervangen door Asterisk PBX servers. Deze zijn verbonden met het netwerk van Lijbrandt Telecom(WAN) en heeft Foreign Exchange Stations(FXS's[7]) in zijn wijkcentrales, (ook wel beter bekend) als distributiepunt voor analoge telefoontoestellen. Met deze FXS is het mogelijk om calls naar de analoge toestellen te sturen en te ontvangen, zodat men kan bellen en gebeld kan worden met analoge toestellen via het VoIP netwerk van Lijbrandt Telecom.
Afbeelding 7.1.1 'VoIP Infrastructuur' toont een alternatieve oplossing om de kosten te drukken van de Tenovis PABX telefooncentrale:



Afbeelding 7.1.1 'VoIP Infrastructuur'

[7] FXS – Foreign Exchange Station, apparaat die signalen stuurt naar analoge toestellen om deze over te laten gaan

Onderstaand schets ik twee scenario's waarin de processen worden verduidelijkt van [1] telefoneren binnen het netwerk van Lijbrandt Telecom met VoIP en [2] telefoneren vanuit het netwerk van Lijbrandt Telecom naar buiten met VoIP.

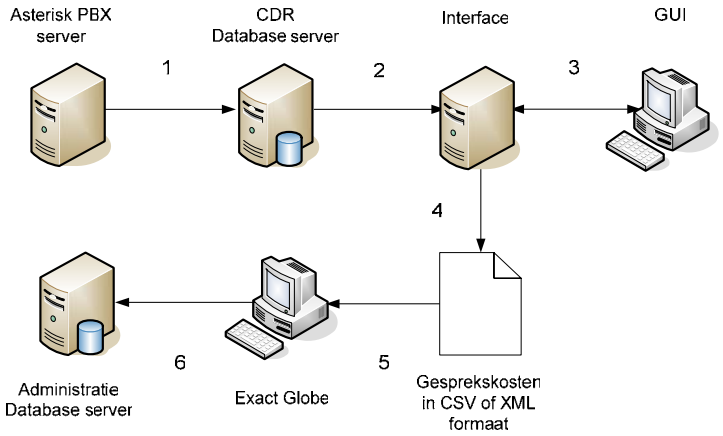
Scenario 1 – Alice belt naar Bob
Alice belt naar haar vriend Bob in Den Haag. De FXS zorgt ervoor dat het gesprek met analoge telefoon via ethernet verzonden kan worden. De Asterisk PBX server te Hillegom verbindt met de Asterisk PBX server te Den Haag via een glasvezelverbinding van het Lijbrandt Netwerk. Omdat Bob een IP toestel gebruikt en aan is gesloten op ethernet, is het niet meer nodig om het gesprek om te laten zetten door een FXS en gaat de telefoon over bij Bob.

Scenario 2 – Bob belt naar Charlie
Bob belt naar zijn schoonbroer Charlie die in Groningen woont. Via ethernet zal de IP telefoon van Bob verbinding leggen met de Asterisk PBX server en gebruikt de Asterisk PBX server een ISDN 30 kaart als een gateway naar de PSTN. De telefooncentrale van Provider X zorgt er vervolgens voor dat de telefoon van Charlie overgaat.

Als men overstapt naar een VoIP infrastructuur, dan blijft het mogelijk om van en naar analoge toestellen te bellen als er gebruik wordt gemaakt van FXS's.

7.2 Verloop van processen

Verschillende processen die voorheen nodig waren om de kosten te registreren worden vervangen door de software die ik ga ontwikkelen. Het gaat hier om de processen 'Collecteren van CDR's met een Collect PC' en 'Berekenen van kosten en het toegankelijk te maken voor Exact Globe'.
Asterisk PBX server biedt de mogelijkheid om zijn CDR's weg te schrijven naar verschillende DBMS'en zoals bijvoorbeeld MySQL en PostgreSQL. Dankzij deze functionaliteit is het niet meer nodig om een Collect PC toe te wijden aan het opvangen en versturen van CDR's.
Op afbeelding 7.2 'Schema – gewenste situatie' ziet u een schets van de processen die er gaan komen.



Afbeelding 7.2 'Schema - gewenste situatie'

Hieronder zal ik stapsgewijs het schema toelichten:

1. De Asterisk PBX verwerkt de gesprekgegevens en schrijft deze weg naar een CDR database server.
2. De interface haalt de gegevens op uit de CDR database van de CDR database server. Vervolgens berekent de applicatie de kosten van iedere CDR.
3. Een GUI kan gesprekskosten van een specifieke maand opvragen van ieder project weergeven en tevens de opdracht geven om de verrekende gesprekskosten in bestandformaat op te slaan. De locatie van de GUI is transparant.
4. De interface genereert de gesprekskosten en zet deze om in CSV of XML formaat.
5. Exact Globe leest dit bestand in
6. Exact Globe slaat de gegevens op ten behoeve van het factureren in de Administratie database die op de Administratiedatabase server staat.

Omdat Lijbrandt Telecom geen gesprekskosten verrekent in het weekend, zal de software verschillende tarieven moeten kunnen hanteren. Een voorbeeld hiervan is het weektarief[8] en het weekendtarief[9]. Hoewel Lijbrandt Telecom momenteel gebruik maakt van deze tarieven, dient men de mogelijkheid te krijgen om zelf in te stellen wanneer er kosten worden verrekend en wanneer niet.

[8] Weektarief – Tarief dat momenteel geldt van maandag 0:00:00 tot en met vrijdag 23:59:59
[9] Weekend – Tarief dat momenteel geldt van zaterdag 0:00:00 tot en met zondag 23:59:59

8. Ontwerp van de software

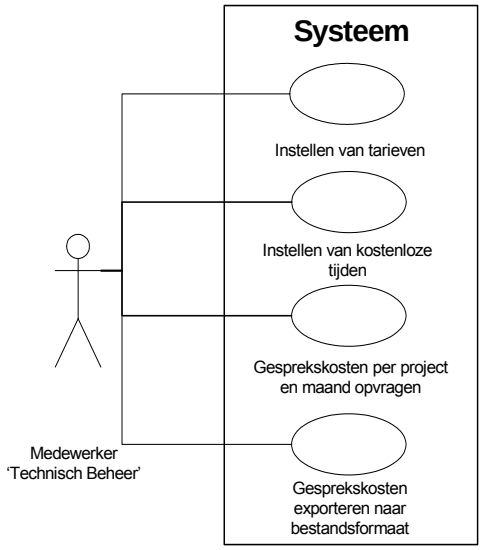
Voor het ontwerpen van het systeem heb ik de in hoofdstuk 6 genoemde user stories verder uitgewerkt in use cases en use case scenario's.
De keuze om het ontwerp te modelleren in UML is, omdat ik heb ervaren dat het uitermate geschikt is voor het ontwikkelen van software. Een andere reden is dat ik meerdere ontwerpfasen van softwaretrajecten heb doorlopen met Unified Modelling Language en zo risicofactoren wil vermijden zoals tijdgebrek. Zo bespaar ik tijd die ik kan steken in het concentreren op het ontwerpen en ontwikkelen van de software, geschreven is in een procedurele taal. Omdat ik nog nooit de modelleringtechniek UML heb gebruikt in combinatie met een procedurele taal, ben ik niet in staat om alle technieken toe te passen zoals bijvoorbeeld: Klassendiagrammen en Sequentiediagrammen.

8.1 Use cases

De UML techniek hanteert use cases om de functionaliteiten van het systeem in kaart te brengen. Dit gebeurt met behulp van functionaliteiten die een actor[10] uit kan voeren op het systeem. Per functionaliteit wordt een use case opgesteld, om de basis te vormen voor het ontwerp van het systeem.
Met behulp van de user stories ben ik tot de conclusie gekomen dat het systeem aan verschillende functionaliteiten moet voldoen, namelijk:

1. Instellen van tarieven
2. Instellen van kostenloze tijden
3. Gesprekskosten van een project per maand opvragen
4. Gesprekskosten exporteren naar bestandsformaat

Hoe ik de user stories heb kunnen verwerken naar deze use cases leest u in paragraaf 8.2 Use case scenario's bij de beschrijving van de use cases.
Op de afbeelding 8.1 'Use Cases' worden de functionaliteiten in kaart gebracht, wat het systeem moet bevatten en wie de functionaliteiten uit gaat voeren.



Afbeelding 8.1 'Use Cases'

[10] Actor - een persoon of object case die de use case activeert

8.2 Use case scenario's

Om een beter beeld te geven hoe de use cases zullen functioneren en wat de aannames zijn om deze functionaliteiten uit te kunnen voeren, heb ik de use cases uitgewerkt tot use case scenario beschrijvingen.
In een use case scenario beschrijving komen de volgende aspecten voor; pre conditie, beschrijving en post conditie. In de pre conditie beschrijf ik wat de aanname is om de use case uit te kunnen voeren. In de beschrijving wordt in detail beschreven hoe de use case itereert met de actor en wat er gebeurt.
Tot slot beschrijf ik in de post conditie wat het eindresultaat is van de use case.
Hieronder de use cases uitgewerkt in use case scenario's:

Instellen van tarieven

Pre-conditie
Er moet een verbinding tot stand zijn gebracht met de database waar de tarieven in staan of komen te staan.

Beschrijving
De gebruiker opent met behulp van een web-browser de GUI[11]. In de GUI is het zichtbaar wat de huidige tarieven zijn voor gesprekken binnen dit project.
Na het instellen van de waarde van een van deze categorieën, drukt men op een submit/bevestigen knop en zal het systeem de tarieven opslaan in de database.
Voor het instellen van 0900 informatienummers, dient de gebruiker van het systeem op een link te klikken. Deze link verwijst de gebruiker door naar een nieuw scherm met een overzicht van alle 0900 nummers en bijbehorende tarieven. Deze tarieven kunnen worden ingesteld door de gebruiker en opgeslagen voor toekomstige kostenberekeningen.

Post-conditie
Tarieven zijn ingesteld.

Instellen van kostenloze tijden

Pre-conditie
Geen.

Beschrijving
De gebruiker opent met behulp van een web-browser de GUI. In de GUI is het zichtbaar wat de huidige kostenloze tijden zijn.
De gebruiker van het systeem krijgt de mogelijkheid om de dag[12] en tijd in te stellen wanneer de 'kostenloze tijd' start en wanneer de 'kostenloze tijd' eindigt.
Tevens is het ook mogelijk om een datum en tijd in te voeren om een dag dan wel dagen kosteloos te maken.
Indien de gebruiker van het systeem de kostenloze tijden heeft ingesteld, dan drukt de gebruiker op de submit/bevestigen knop om deze wijzigingen door te voeren en op te slaan in de database.

[11] GUI – Grafische User Interface, grafische koppeling tussen mens en machine
[12] Dag – Dag in de tekstuele vorm zoals; maandag, dinsdag, woensdag, donderdag, vrijdag, zaterdag of zondag

Post-conditie
Kostenloze tijden zijn ingesteld.
Compatibiliteit facturering

Gesprekskosten van een project per maand opvragen

Pre-conditie
De tarieven dienen ingevoerd te zijn.

Beschrijving
De gebruiker van het systeem opent met behulp van een web-browser de GUI. In de GUI kan men een maand selecteren met behulp van een drop down list en voert men een jaar in.
Tot slot dient men op de knop submit/bevestigen te drukken om de gesprekskosten in beeld te krijgen. De gesprekskosten worden opgehaald uit de database die voldoen aan de criteria die de gebruiker heeft ingesteld en met deze gegevens worden de kosten berekend.

Post-conditie
De gegevens van de kosten zijn per project, per afdeling en per maand(en) opgevraagd en afgebeeld. Ook zijn de kosten verrekend.

Gesprekskosten exporteren naar bestandsformaat

Pre-conditie
De maandelijkse gesprekskosten van een project dienen opgevraagd zijn.

Beschrijving
Nadat de gesprekskosten zijn opgehaald uit de database en deze zijn weergegeven op de GUI, heeft de gebruiker de mogelijkheid om deze gegevens te exporteren naar een bestand. De gebruiker drukt op de knop ‘export/exporteren’ en het systeem zal de gegevens omzetten naar bestandsformaat.

Post-conditie
De gesprekskosten zijn geëxporteerd naar bestandformaat en gereed om ingelezen te worden door Exact Globe.

8.3 Toestandsdiagrammen

Een toestandsdiagram is een modelleringwijze binnen UML die kan worden toegepast ter verduidelijking van de toestanden waarin de verschillende objecten, in dit geval de software, zich kunnen bevinden. In het toestandsdiagram worden de toestanden en transities opgenomen. Deze transities zorgen voor veranderingen van de toestand binnen het systeem.
Ik heb op dit project toestandsdiagrammen toegepast om zo meer duidelijkheid te verschaffen op het gebied van welke toestanden de applicatie zich bevindt.
Tijdens het maken van de toestandsdiagrammen had ik moeite met het in kaart brengen van het hele proces. Dit omdat ik ga programmeren in een procedurele taal en dat het onmogelijk is objectgeoriënteerde modellen voor te ontwerpen.
Om de toestandsdiagrammen te realiseren heb ik gebruik gemaakt van het programma 'Microsoft Visio'. Dit programma heb ik gebruikt omdat ik van de trial versie van IBM Rational[13], slechts gelimiteerd gebruik kan maken.

Bij het modelleren van verschillende toestandsdiagrammen heb ik gebruik moeten maken van beslissingen en join[14]-tekens. Beslissingen worden aangeduid met behulp van een ruitvorm zoals weergegeven in afbeelding 8.3.1 'Visio – Beslissingteken'.

Afbeelding 8.3.1 'Beslissingteken'

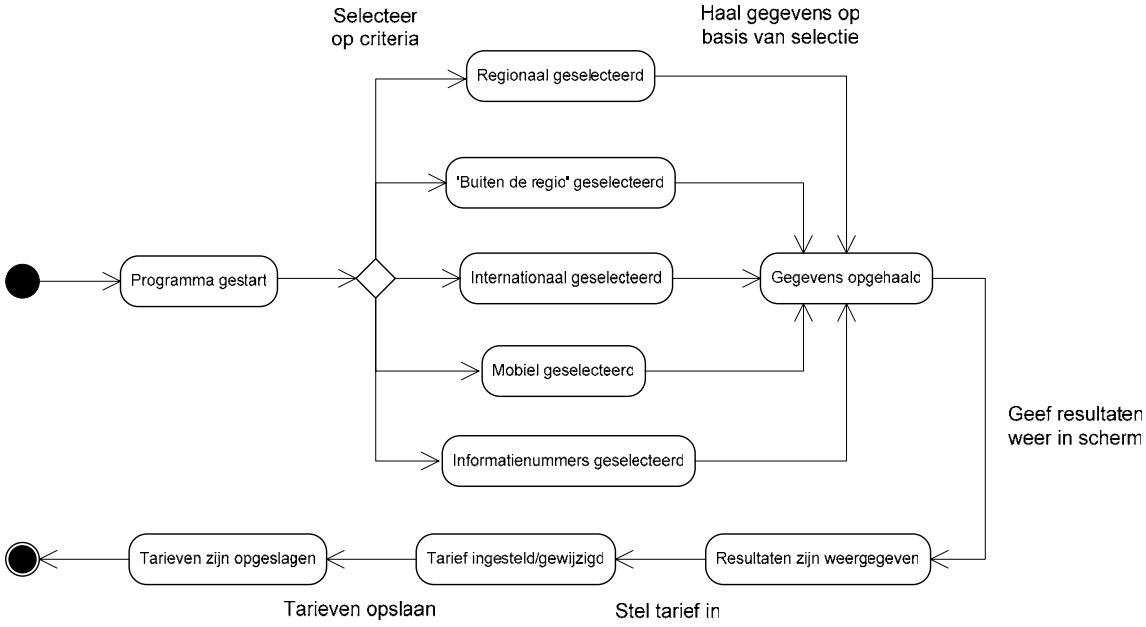
Het afbeelden van joins doe ik door gebruik te maken van een samengang van een pijl naar een rechthoekig zwart blok en een opsplitsing van verschillende pijlen die de transities moeten voorstellen. Op afbeelding 8.3.2 'Visio – Join teken' ziet u de teken die ik hiervoor heb gebruikt.

Afbeelding 8.3.2 'Join-teken'

[13] IBM Rational – Softwareontwikkeling-software van IBM
[14] Join - samenkomende transities voor het bereiken van een toestand

Op de volgende afbeelding 8.3.3 ‘Toestandsdiagram – Instellen tarieven’ kunt u zien hoe ik een beslissing heb toegepast binnen een toestandsdiagram.
De pijlen die bijeenkomen bij de toestand ‘Gegevens opgehaald’ zijn op basis van OR. Dit houdt in dat de toestand pas van kracht is indien een transitie uit één van de toestanden afkomstig is. De verschillende toestanden zijn; ‘Regionaal’, ‘Buiten de regio’, ‘Internationaal’, ‘Mobiel’ of ‘Informatienummer’.
Hieronder de toestandsdiagram ‘Instellen tarieven’:

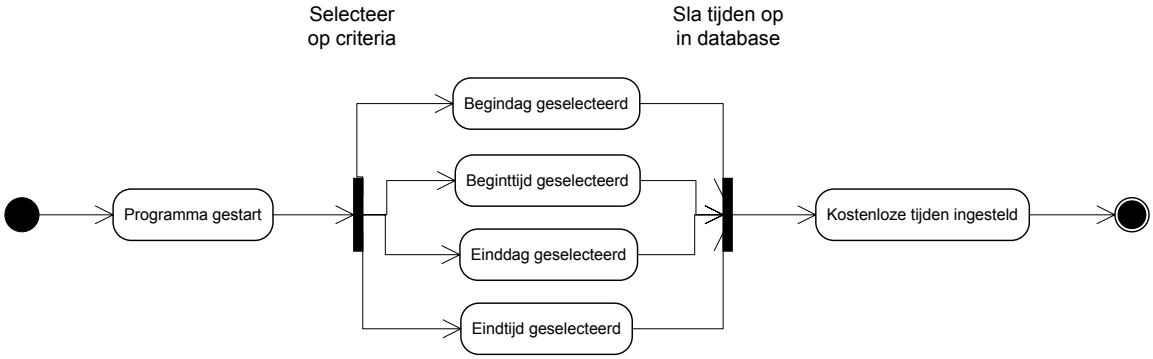
Instellen van tarieven



Afbeelding 8.3.3 ‘Toestandsdiagram – Instellen tarieven’

Op afbeelding 8.3.4 ‘Toestandsdiagram – Instellen kostenloze tijden’ ziet u hoe ik de join heb toegepast binnen een toestandsdiagram. De toestand ‘Kostenloze tijden ingesteld’ is pas van kracht indien alle transities uit de toestanden bijeengekomen zijn. Dit betekend dat de gebruiker van het systeem eerst de ‘Begindag’, ‘Begintijd’, ‘Einddag’ en ‘Eindtijd’ heeft moeten selecteren zodat het mogelijk is om deze gegevens op te slaan in een database. Hieronder het toestandsdiagram ‘Instellen van kostenloze tijden’:

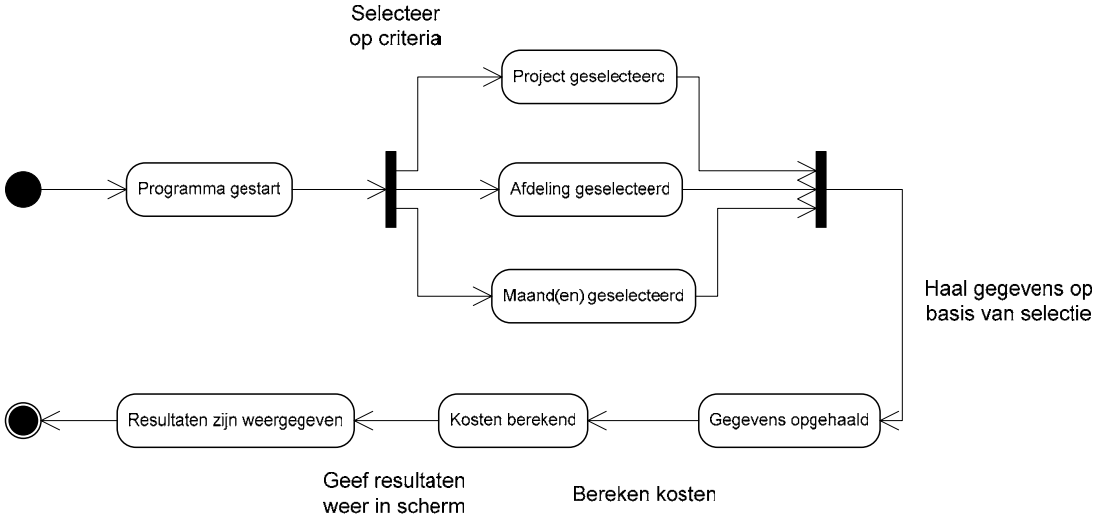
Instellen van kostenloze tijden



Afbeelding 8.3.4 ‘Toestandsdiagram – Instellen kostenloze tijden’

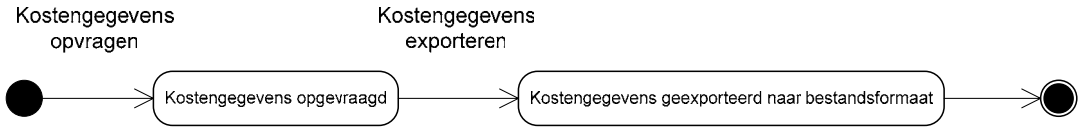
Gesprekskosten van een project per maand opvragen

De use case ‘Gesprekskosten van een project per maand opvragen’ kent het volgende toestandsdiagram:



Gesprekskosten exporteren naar bestandsformaat

De use case ‘Gesprekskosten exporteren naar bestand’ kent het volgende toestandsdiagram:



8.4 Databaseontwerp

Voor het ontwerpen van de database heb ik eerst uitgezocht welke informatie de Asterisk PBX server aanbiedt. Vervolgens heb ik uitgezocht welke gegevens er zijn op het gebied van tarieven.
De asterisk server kan in combinatie met verschillende DBMS'en werken en kan de waarden wegschrijven naar een database. Volgens de Asterisk documentatie wordt het volgende gegevensmodel gehanteerd voor CDR:

```
calldate
clid
src
dst
dcontext
channel
dstchannel
lastapp
lastdata
duration
billsec
disposition
amaflags
accountcode
uniqueid
userfield
```

Met deze informatie alleen is het niet mogelijk om de gesprekskosten te berekenen, men moet ook weten welke tarieven er worden gehanteerd voor het verrekenen van de verbruikskosten. Om deze reden ben ik genoodzaakt om naast deze tabel, andere tabellen op te nemen in mijn database die deze parameters bevatten.

Voor het ontwerpen van een database moet ik eerst weten voor welk type DBMS ik de database ga ontwerpen. Binnen Lijbrandt Telecom wordt gebruik gemaakt van PostgreSQL DBMS. Dit DBMS is gratis voor commercieel en non-commercieel gebruik. PostgreSQL is een ORDBMS[15] die tevens relationele databases ondersteunt. De keuze is aan mij om een databaseontwerp te maken voor een ORDBMS of een RDBMS[16]. Ik heb gekozen om mijn ontwerp te maken voor een relationele database. Dit vanwege de omvang van mijn afstudeerproject. Mijn kennis en ervaring van ORDBMSen is beperkt. Om deze reden zal ik mij hier nog in moeten verdiepen. Een gevolg kan zijn dat mijn afstudeerproject gevaar zou kunnen lopen in verband met tijdgebrek. In tegenstelling tot ORDBMSen , is mijn kennis en ervaring van RDBMSen aanzienlijk hoger.

Het relationeel databaseontwerp zal ik aan de hand van het proces normaliseren[17] ontwerpen. Dit doe ik met behulp van 4 zogenaamde normaalvormen. Dit zijn: 0^{e} normaalvorm, 1^{e} normaalvorm, 2^{e} normaalvorm en tot slot de 3^{e} normaalvorm.

[15] ORDBMS – Object Relational Database Management System
[16] RDBMS - Relational Database Management System
[17] Normaliseren – een proces om inconsistentie en redundantie te voorkomen binnen een database

8.4.1 0e Normaalvorm

In de 0e normaalvorm ga ik alle gegevens inventariseren. Voor ieder gegeven geef ik een beschrijving om zo te achterhalen welke gegevens relevant zijn voor de use cases die ik heb opgesteld:

```
CDR
calldate – Datum en tijd van het gesprek
clid – CallerID in tekstvorm
src – CallerID in number
dst – Bestemming extensie
dcontext – Bestemming context
channel – Channel dat gebruikt wordt
dstchannel – Channel van bestemming
lastapp – Laatste applicatie
lastdata – Laatste applicatie data
duration – Lengte van het gesprek
billsec – Totale lengte van seconden die verrekent worden
disposition – Wat er met het gesprek gebeurt is
amaflags – Te gebruiken 'flags'
accountcode – Welke accountnummer er gebruikt moet worden
uniqueid – Unique Channel Identifier
userfield – Veld waar gebruiker wat mee kan geven

Tarieven
prefix – Netnummer
tarief – Tarief voor netnummer
begintijd – Tijd waar vanaf het tarief op geldt
eindtijd – Tijd waar het tarief op eindigt
starttarief – Tarief voor de startkosten van een gesprek

Kostenloze tijden
startdate – Datum en tijd waarop de kostenloze tijden
            tarief start
expiredate – Datum en tijd waarop de kostenloze tijden
             tarief eindigt
```

8.4.2 1e Normaalvorm

In de 1e normaalvorm worden alle multi-valued attributen uit het model gehaald en deze dient men op te nemen in een nieuw datamodel.
De attributen die ik tijdens de 0e normaalvorm heb geïnventariseerd bevatten geen multi-valued attributen dus staan de datamodellen al in de 1e normaalvorm.

8.4.3 2e Normaalvorm

De datamodellen staan pas in de 2e normaalvorm indien alle functioneel afhankelijke attributen een determinant hebben. Dit betekend dat gegevens van een attribuut opgezocht kunnen worden aan de hand van een determinant(primaire sleutel). Een primaire sleutel mag slechts één keer voorkomen binnen een tabel en nooit veranderen. Een primaire sleutel wordt aangeduid door het onderstrepen van het gegeven.

We hebben de volgende datamodellen: CDR, Tarieven en Kostenloze tijden.

De gegevens van de CDR's zijn niet allemaal nodig voor de use cases die ik heb opgesteld. Om deze reden zal ik niet alle gegevens gebruiken binnen mijn datamodel die ik geïnventariseerd had. Slechts 4 gegevens zijn relevant binnen het datamodel van de tabel 'cdr':

```
src
dst
calldate
billsec
```

In het datamodel 'cdr' zijn alleen functioneel afhankelijke attributen aanwezig en geen determinanten. Hierdoor is het niet mogelijk om een attribuut binnen het datamodel een primaire sleutel toe te kennen.
Een oplossing hiervoor is om een samengestelde primaire sleutel aan te stellen aan: src en calldate. Een andere oplossing is het opnemen van een veld en deze toekennen als primaire sleutel.
Ik heb gekozen voor het opnemen van een nieuw veld en deze toe te kennen als primaire sleutel omdat een primaire sleutel zo kort mogelijk moet zijn. Hieronder het vernieuwde datamodel van 'cdr':

```
call_id
src
dst
calldate
billsec
```

Voor het datamodel 'tarieven' geldt hetzelfde probleem als het datamodel van 'cdr'. Er is geen determinant waar de functioneel afhankelijk attributen opgezocht kunnen worden. Hieronder het datamodel 'tarieven':

```
prefix
tarief
begintijd
eindtijd
starttarief
```

Na het toevoegen van een extra veld en deze toe te wijzen als primaire sleutel heb ik het volgende datamodel voor de tabel 'tarieven':

```
tarieven_id
prefix
tarief
begintijd
eindtijd
starttarief
```

Tot slot het laatste datamodel 'kostenloze tijden'. Om de use case 'Toepassen van kostenloze tijden' uit te kunnen voeren, is het noodzakelijk om deze kostenloze tijden in te stellen en op te vragen.

Hieronder het datamodel van de tabel 'kostenloze tijden':

```
startdate
expiredate
```

Ook hierbij neem ik een nieuw veld op en stel ik deze aan als primaire sleutel. Dit veld heb ik tijd_id genoemd. Hieronder het resultaat van het normaliseren naar de 2e normaalvorm:

```
tijd_id
startdate
expiredate
```

8.4.4 3e Normaalvorm

De 3e normaalvorm is bereikt indien de niet-sleutel attributen niet afhankelijk zijn van andere niet-sleutel attributen. In mijn datamodellen zijn geen attributen afhankelijk van andere niet-sleutel attributen, dus staan ze in de 3e normaalvorm.

9. Releaseplan

9.1 MoSCoW-principe

De methode 'Extreme Programming' stelt prioriteiten aan iteraties door aan te duiden welke iteratie als eerst ontwikkeld moet worden en welke daarna moet volgen. Normaal gesproken houdt men workshops waar alle betrokken personen(gebruiker, opdrachtgever en andere) een mening kunnen uitspreken over de prioriteit van iedere iteratie.
De opdrachtgever en ik hebben het gelaten bij het doornemen van de use cases omdat het proces te veel tijd in beslag neemt.

Om meer duidelijkheid te verschaffen en de verwachtingen tussen mij en de opdrachtgever vast te stellen, heb ik gebruik gemaakt van het MoSCoW-principe op het gebied van welke use cases prioriteit hebben en dus als eerst gerealiseerd moeten worden.
Door het MoSCoW-principe te hanteren worden de use cases van het systeem opgedeeld in 4 verschillende prioriteiten; 'Must have', 'Should have', 'Could have' en 'Want to have'.
Hieronder een opsomming met daarbij een toelichting per prioriteit:

- Must have, *de use cases die gerealiseerd moeten worden, deze vormen de basis van de software*
- Should have, *de use cases die het systeem zou moeten bevatten, maar waarvoor eventueel een work-around* voor beschikbaar is*
- Could have, *de use cases die het systeem zou kunnen bevatten indien er genoeg tijd beschikbaar is en eventueel kan komen te vervallen zonder dat het conflicteert met de toepasbaarheid van het systeem*
- Want to have, *de use cases die het systeem moet bevatten maar die niet tot de doelstelling behoren*

Door het bovenstaande principe te hanteren wil ik bewerkstelligen dat de opdrachtgever een duidelijk beeld heeft over de prioriteiten die ik vaststel en wat iedere prioriteit bevat op het gebied van het realiseren van het systeem.
De use cases die ik heb opgesteld ken ik de volgende prioriteiten toe:

Kostenregistratie	Must have	Should have	Could have	Want to have
Toepassen van tarieven	X			
Toepassen van kostenloze tijden			X	
Gesprekskosten van een project, per maand opvragen		X		
Gesprekskosten exporteren naar bestandformaat		X		

Tabel 9.1 MoSCoW-principe 'Kostenregistratie binnen VoIP'

9.2 Project Velocity

Tijdens het definiëren van de 'project velocity' wordt vastgesteld in hoeverre het project kan worden afgerond binnen de opgegeven tijd. Dit kan op verschillende manieren worden gedaan. Zo kan het releaseplan 'time-based' of 'scope-based' worden gedefinieerd. Bij een 'time-based' releaseplanning worden de iteraties uitgestrekt over de gegeven tijd die men voor het project heeft. Indien het project 'scope-based' is gepland, dan wordt er gepland hoe lang iedere iteratie zal gaan duren.

Voor het plannen van de iteraties voor het softwareontwikkelingstraject, heb ik gekozen om middels 'time-based' het project te plannen. De reden dat ik het project op 'time-based' basis heb ingedeeld, is omdat er een gegeven tijd is toegekend om het project te kunnen realiseren.
Bij de planning van de ontwikkeling van de software heb ik rekening gehouden met de acceptatietesten en het reviseren van de iteratie. Zo voorkom ik tijdsdruk door onverwachte werkzaamheden. In het plan van aanpak (zie hoofdstuk 4) is een globale planning opgesteld, die echter niet kan worden toegepast na de bevindingen over de grootte van de werkzaamheden.
Er zijn een aantal werkzaamheden gecombineerd, dit om te kunnen achterhalen of het mogelijk is om de software te ontwikkelen binnen de toegekende tijd. Dit zijn de werkzaamheden; User requirements vaststellen en het opstellen van een releaseplan.

Op afbeelding 9.2.1 ziet u de nieuwe planning voor het softwareontwikkelingstraject. Het softwareontwikkelingtraject bestaat uit de software 'Kostenregistratie binnen VoIP'.

		Task Name	Duration	Start	Finish
1		**Softwareontwikkelingtraject**	**58 days**	**Mon 28-2-05**	**Fri 20-5-05**
2	✓	User requirements vastleggen	5 days	Mon 28-2-05	Fri 4-3-05
3		Milestone: Document klanteisen 'Kostenregistratie'	0 days	Fri 4-3-05	Fri 4-3-05
4	✓	Releaseplan opstellen	5 days	Mon 7-3-05	Fri 11-3-05
5		**Release 'Kostenregistratie'**	**48 days**	**Mon 14-3-05**	**Fri 20-5-05**
6	✓	Use cases vaststellen	2 days	Mon 14-3-05	Tue 15-3-05
7	✓	Use case scenario's beschrijven	3 days	Wed 16-3-05	Fri 18-3-05
8		Database-ontwerp 'Kostenregistratie'	5 days	Mon 21-3-05	Fri 25-3-05
9		Milestone: Ontwerp	0 days	Fri 25-3-05	Fri 25-3-05
10		**Software 'Kostenregistratie' ontwikkelen**	**32 days**	**Tue 29-3-05**	**Thu 12-5-05**
11		Iteratie: Toepassen van tarieven	8 days	Tue 29-3-05	Thu 7-4-05
12		Iteratie: Toepassen van kostenloze tijden	8 days	Fri 8-4-05	Tue 19-4-05
13		Iteratie: Kostengegevens per project, afdeling en mac	8 days	Wed 20-4-05	Fri 29-4-05
14		Iteratie: Kostengegevens exporteren naar bestand	8 days	Mon 2-5-05	Thu 12-5-05
15		Milestone: prototype 'Kostenregistratie'	0 days	Fri 13-5-05	Fri 13-5-05
16		Acceptatietesten/Revisie	6 days	Fri 13-5-05	Fri 20-5-05
17		Milestone: Kostenregistratie opleveren	0 days	Fri 20-5-05	Fri 20-5-05

Afbeelding 9.2.1 'Softwareontwikkelingtraject 'Kostenregistratie binnen VoIP'

10. VoIP testopstelling opzetten

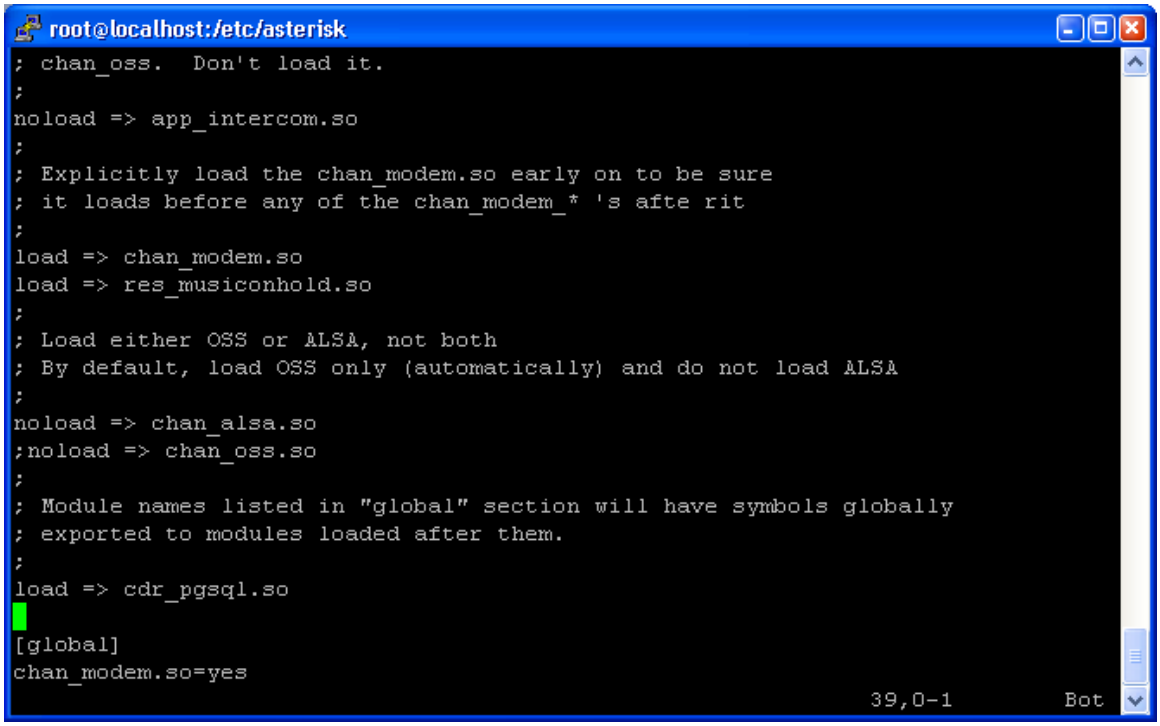
Na het opzetten van het plan van aanpak ben ik aan de slag gegaan met het maken van VoIP testopstelling. Dit houdt in dat er een omgeving is opgesteld waar ik mee kan experimenteren. Een omgeving die ik tot mijn beschikking heb om mee te werken zonder dat er consequenties volgen voor het functioneren van diensten binnen Lijbrandt Telecom.
Voor het opzetten had ik tot uiterlijk 27 februari 2005 ingepland, om zo te voorkomen dat ik me voornamelijk bezig zou houden met het ontwerpen van de netwerkarchitectuur.
Indien de VoIP testopstelling nog niet klaar zou zijn op 27 februari, dan zou ik beginnen aan het softwareontwikkelingtraject met de testopstelling die op dat moment is opgesteld.
In voorgaande planningen had ik werkzaamheden opgenomen om een onderzoeksrapport over VoIP protocollen op te leveren. Vanaf planning versie 2.1 heb ik dit achterwege gelaten om minder deelproducten op te leveren op het gebied van netwerkarchitectuur en zo meer tijd vrij te maken voor het softwareontwikkelingtraject.

10.1 Gegevensverwerking van Asterisk

Om de mogelijkheden van Asterisk PBX server op het gebied van gegevensverwerking te ontdekken heb ik informatie gevonden via diverse websites. Met deze informatie heb ik kunnen concluderen dat Asterisk de mogelijkheid biedt om op verschillende manieren Call Detail Records(CDR) te verwerken. Standaard staat Asterisk ingesteld om de CDR's op te slaan in CSV bestanden.
Tevens biedt Asterisk de mogelijkheid om de CDR's op te slaan in een database management systeem zoals MySQL en PostgreSQL.
Hiervoor moet men Asterisk instellen binnen de configuratiebestanden om het daadwerkelijk weg te laten schrijven naar een database op één van deze DBMSen.
Eerst moeten de modules worden geïnstalleerd met behulp van het `make` commando. Ik heb de verschillende modules geïnstalleerd die bij de Asterisk softwarepakket zaten.

Vervolgens dienen we de postgreSQL module te laden, zodat de CDR gegevens worden weggeschreven naar een postgreSQL database. Dit doe ik door een configuratiebestand aan te passen. Hiervoor heb ik de tekst-editor 'VI' gebruikt dat standaard op de Fedora Core linux distributie te vinden is. Het gaat hier om het configuratiebestand '`modules.conf`'.
Dit bestand kan men vinden onder de padnaam `/etc/asterisk/`.

Op afbeelding 10.2.1 ‘remote modules.conf’ vindt u een deel van wat in het configuratiebestand ‘`modules.conf`’ te zien is.



Afbeelding 10.2.1 ‘remote modules.conf’

Door de volgende regel aan het script toe te voegen zorgt men ervoor dat de Asterisk PBX server, CDR gegevens wegschrijft naar een postgreSQL database:

```
load => cdr_pgsql.so
```

Nu de Asterisk PBX server de module laad om CDR gegevens weg te kunnen schrijven naar een postgreSQL database, moet men ook nog opgeven waar deze database zich bevindt. Dan pas is het mogelijk om toegang te verkrijgen tot de gegevens van de database.
Met Asterisk PBX is het mogelijk om de CDR gegevens weg te schrijven naar databases op een toegewezen systeem. In mijn opstelling bevindt de database zich op hetzelfde systeem als waar de Asterisk PBX server is geïnstalleerd omdat het hier slechts om een testopstelling gaat. Wellicht is er behoefte om de Asterisk PBX server van de database te onderscheiden tijdens grootschalige implementatie. Er bestaat een mogelijkheid dat de Asterisk PBX server’s performance onder druk komt te staan indien de database wordt geraadpleegd door de software.

Op afbeelding 10.2.2 ‘Instellingen pgsql’ ziet u de instellingen voor het configureren van een database.

Afbeelding 10.2.2 ‘Instellingen pgsql’

De Asterisk PBX server zal CDR gegevens wegschrijven naar de database ‘asterisk’ met als gebruikersnaam ‘root’.
Na dit ingesteld te hebben is het nu mogelijk om de gegevens op te vragen over de CDR’s.

11 Architectural Spike

Voordat ik begin met het ontwikkelen van de software ga ik eerst een prototype ontwikkelen ten behoeve van de use case 'Kosten opvragen'. Dit doe ik om in te kunnen schatten of de implementatiewijze mogelijk is. Zo voorkom ik situaties waarin ik vast kom te zitten tijdens het ontwikkelen van de software.
Voordat het prototype wordt ontwikkeld moet er eerst een klein programma geschreven worden dat de database vult met willekeurige CDR's.

11.1 Programma om database te vullen

Om het prototype te kunnen testen op de werking ervan heb ik een programma geschreven om de database te vullen met CDR's die willekeurige data bevatten.
Om een ruwe schatting te kunnen maken over hoe de software zich zal gedragen op het gebied van performance, ga ik simulaties uitvoeren om te meten hoe lang het zou duren voordat de gegevens uit de database zijn gehaald.
Op dat moment besefte ik dat dit niet aan bod was gekomen tijdens het opstellen van de use cases, dus heb ik dit moeten achterhalen bij de opdrachtgever. Daaruit was de volgende klanteis voortgekomen:

- Het software systeem moet sneller de gesprekskosten kunnen berekenen dan met MIND CMS PhonEx.

Om een indicatie te krijgen hoe lang MIND CMS PhonEx bezig met het verrekenen van de gesprekskosten, heb ik informatie vergaard van de medewerkers van afdeling Technisch Beheer. Zij hebben de gesprekskosten van een project laten berekenen met behulp van MIND CMS PhonEx. Dit project bevatte circa 75.000 uitgaande gesprekken die waren gevoerd binnen één maand. Het duurde ongeveer 5 minuten om de gesprekskosten te berekenen van dit project. Deze gegevens zal ik dit als een meetindicatie gebruiken voor het testen van het prototype op het gebied van performance.

Vervolgens heb ik het programma geschreven in de programmeertaal Perl voor het vullen van willekeurige waardes:

```
use DBI;
use POSIX qw/strftime/; #Roep de module strftime aan om
tijdsformaten aan te geven

$db = DBI->connect('dbi:Pg:dbname=asterisk', 'root', undef,
        { RaiseError => 1, AutoCommit => 1});

$st = $db->prepare("INSERT INTO cdr (calldate, src, dst,
billsec) VALUES (?, ?, ?, ?);");

#Maak subroutine voor het toekennen van een numerieke waarde
sub random_number {
  my ($n) = @_; #Defineer de meegegeven parameter
  my $str = "";
  #Voer de loop een X aantal malen uit
  for(1..$n) {
    $str .= int(rand(10));
  }
  return $str; #Geef de waarde terug van de 10 getallen die aan
elkaar zijn geplakt
}
```

```
#Voer dit 75000 maal uit
for(1..75000) {
  $calldate = strftime "%Y-%m-%d %H:%M:%S", gmtime(rand(time));
  $src = random_number(10);
  $dst = random_number(10);
  $billsec = int(rand(1000));
  print "$src\n";
  $st->execute($calldate, $src, $dst, $billsec); #Schrijf dit
weg naar de database
}
```

Omdat het datamodel van de tabel ‘cdr’ een datumveld bevat, is het alleen mogelijk om hiervoor datum en/of tijden voor in te voeren.
Binnen het script maak ik gebruik van de `strftime` functie uit de POSIX module, die een aangepast formaat kan weergeven van datum/tijd. De reden dat ik gebruik maak van de `strftime` functie is dat het mij de mogelijkheid biedt om de datum en tijd in aangepaste vorm weer te geven.
Om deze functie te kunnen benutten, ben ik genoodzaakt om de module POSIX en de functie `strftime` bekend te maken binnen de software. Dit staat ook bekend als ‘Information Expert’ in de theorie van Craig Larman, uit het boek ‘Creating patterns’.
Hieronder de syntax voor het gebruik van de `strftime` functie:

```
use POSIX qw/strftime/;
```

Na het verbinden met de database, heb ik een subroutine geschreven ‘random_nummer’. Deze subroutine zorgt ervoor dat er een willekeurig tiencijferig nummer wordt toegekend aan een variabel. Hieronder het deel van het script dat ervoor zorgt dat er een willekeurige tiencijferig nummer wordt gegenereerd:

```
sub random_number {
  my ($n) = @_;
  my $str = "";
  for(1..$n) {
    $str .= int(rand(10));
  }
  return $str;
}
```

Binnen deze functie definieer ik variabelen met behulp van ‘`my`’. Dit betekent dat deze variabelen alleen geldig zijn binnen de subroutine, ook wel bekend als scope.
Allereerst definieer ik variabele `$n` als de waarde van het argument welke met de `random_number` functie wordt meegegeven. Vervolgens zorg ik ervoor dat de variabele `$str` een lege string is, want deze ga ik gebruiken voor het opbouwen van het willekeurig tiencijferige nummer.
De ‘`for`’-loop zal ik een aantal keer doorlopen, in dit geval het getal dat wordt meegegeven aan de subroutine. Iedere keer als deze ‘`for`’-loop doorlopen wordt, wordt er een willekeurig nummer tussen 0 en 9 toegevoegd aan de variabele `$str`.
Uiteindelijk wordt het opgebouwde nummer dat gedefinieerd is als de variabele `$str` teruggegeven door middel van een `return`.

Om willekeurig gegenereerde waarden weg te schrijven naar de database, gebruik ik een ‘`for`’-loop die 75.000 tuples[18] in de tabel invoert.
De variabelen `$src` en `$dst` krijgen beiden een willekeurig tiencijferig nummer toegekend en de variabele `$billsec` een willekeurig nummer tussen 0 en 1000.
Tot slot schrijf ik de gegevens weg naar de database met de willekeurige nummers door de SQL query uit te voeren op de database.

[18] Tuple – een rij uit een tabel

11.2 PL/pgSQL

Momenteel worden de werkstations belast met het opvragen van de gespreksinformatie en het berekenen van de verbruikskosten over de uitgaande gesprekken. Om dit soort scenario's te voorkomen zal ik gebruik maken van PL/pgSQL. Dit staat voor Procedural Language/ postrgeSQL en is een procedurele taal die binnen PostgreSQL gebruikt kan worden om beperkingen/toevoegingen en andere bewerkingen uit te voeren op een database. De reden dat ik gebruik maak van PL/pgSQL is omdat ik de werkzaamheden niet aan de werkstations wil toekennen, maar aan de server die hier beter geschikt voor is.

Om een aantal acties uit te laten voeren binnen PL/pgSQL, heb ik mijzelf eerst in deze procedure taal moeten verdiepen. Het ging voornamelijk om de syntax, omdat ik de logica erachter wel begreep en dat sommige dingen herkenbaar waren uit andere programmeertalen.
Na een korte zelfstudie heb ik een voorbeeld weten op te zetten om de kosten te kunnen berekenen.
Hieronder op script 11.2.2 'PL/pgSQL' ziet u het script die ik heb gebruikt om de kosten te berekenen op basis van de gebruiksgegevens die beschikbaar zijn via de database.
Let op dat op delen van het script commentaar is gegeven door eerst '--' ervoor te zetten.

```
CREATE OR REPLACE FUNCTION get_call_cost(oid) RETURNS
numeric LANGUAGE plpgsql VOLATILE STRICT AS '
DECLARE

-- Definieer variabelen
  call_oid ALIAS FOR $1; -- Ken de parameter een ALIAS
naam toe genaamd call_oid
  call_billsec int;
  call_netnummer VARCHAR;
  call_tarief NUMERIC;
  call_cost NUMERIC;
  call_startcode INT;
  call_starttarief FLOAT;

BEGIN
-- Ken de gegevens uit de database toe aan de variabelen
binnen de functie
  SELECT INTO call_billsec billsec
    FROM cdr
    WHERE oid = call_oid;
  IF call_billsec IS NULL THEN
    RAISE NOTICE ''call_billsec is null'';
    RETURN NULL;
  END IF;
  SELECT INTO call_netnummer SUBSTR(dst, 1, 3)
    FROM cdr
    WHERE oid = call_oid;
  SELECT INTO call_tarief tarief
    FROM tarieven
    WHERE call_netnummer = prefix;
  IF call_tarief IS NULL THEN
    RAISE NOTICE ''Geen tarief gevonden voor: %.'',
call_netnummer;
    RETURN NULL;
  END IF;
```

```
  SELECT INTO call_startcode startcode
    FROM tarieven;
  SELECT INTO call_starttarief tarief
    FROM starttarieven
    WHERE call_startcode = startcode;

-- Bereken de kosten van het gesprek
  call_cost := call_billsec * call_tarief +
call_starttarief;
  RETURN call_cost;
END;';
```

Script 11.2.2 'Pl/pgSQL'

Dit script dient men te laden binnen postgreSQL door dit in te voeren binnen de sql prompt.
Voor het achterhalen van de tarieven ga ik eerst het tarief ophalen voor het desbetreffende telefoonnummer waar naar gebeld wordt. Dit realiseer ik door de eerste drie nummers te definiëren als een variabele en deze te vergelijken met de netnummers(prefix) binnen de 'tarieven' tabel. Dit doe ik met het volgende queries:

```
SELECT INTO call_netnummer SUBSTR(dst, 1, 3)
  FROM cdr
  WHERE oid = call_oid;
SELECT INTO call_tarief tarief
  FROM tarieven
  WHERE call_netnummer = prefix;
```

Met de functie SUBSTR() geef ik het deel aan van de waarde die ik wil zien. Aan deze functie geef ik de parameters mee van de posities van een string, waar die moet beginnen en waar die moet eindigen. In dit geval gaat het om het eerste tot het derde nummer in de reeks. Een voorbeeld hiervan is wanneer ik een gesprek heb dat als bestemmingsnummer '0105762293' heeft. Het script zal dan de eerste drie nummers; '010' als variabele gebruiken.

Bij het gedeelte waar ik de gegevens uit de database haal en toeken aan de variabelen binnen de functie, worden meerdere queries uitgevoerd op de database. Hieronder een query die ik uitvoer op de database.

```
SELECT INTO call_duration duration
FROM cdr
WHERE oid = call_oid;
```

In het gedeelte waarin ik de functie declareer, definieer ik eerst de variabelen die ik binnen de functie wil gebruiken. Ik ken de waarde van het veld in de kolom 'duration' toe aan de variabele `call_duration`. Met deze variabelen ben ik in staat om berekeningen uit te laten voeren en zo de kosten te achterhalen na ieder gesprek mits er verschillende factoren bekend zijn: Tarief, duratie van het gesprek en de startkosten. In het script ken ik dan de volgende waarde toe:

```
-- Bereken de kosten van het gesprek
  call_cost := call_duration * call_tarief +
call_startcode;
```

De duur van het gesprek wordt vermenigvuldigd met het tarief en daarnaast wordt het starttarief eraan toegevoegd. Zo kom ik uiteindelijk uit op de kosten van het gesprek. Tot slot geef ik de waarde van `call_cost` terug als returnwaarde. Dit betekent dat deze waarde het eindresultaat is van het script.

11.3 Resultaten

Tijdens het uitvoeren van de tabel ‘cdr’ met het vulprogramma die ik in Perl had geschreven, kwam ik erachter dat sommige telefoonnummers niet volledig waren en niet 10 nummers bevatten. De reden dat dit gebeurde was dat de datatypes van src en dst kolommen integers waren. Omdat ik deze twee kollommen als integer datatype had geïmplementeerd, werden de waarden niet als tiencijferig nummer weergegeven. Er waren een aantal willekeurige nummers gegenereerd die met een 0, 00, etc begonnen. Omdat integers deze nullen niet weergeeft en ook het getal verandert naar een getal zonder nullen in het begin, was ik genoodzaakt om het op een andere manier op te lossen.
Om dit probleem op te lossen heb ik de datatypes veranderd in de tabel ‘cdr’. Ik heb hier ‘varchar’ type van gemaakt, die dit als een string verwerkt binnen de tabel. Hieronder het nieuwe datamodel:

```
CREATE TABLE cdr (
  calldate TIMESTAMP NOT NULL,
  src VARCHAR(25) NOT NULL,
  dst VARCHAR(25) NOT NULL,
  billsec INT NOT NULL
);
```

Ik heb het maximum ingesteld op 25 karakters, want het is mogelijk dat men naar het buitenland belt. Het getal 25 heeft geen specifieke reden, maar is naar mijn mening een ruime marge. Om naar het buitenland te bellen vanuit Nederland is het vereist om twee maal een 0 te draaien, vervolgens de landcode, dan het netnummer en tot slot het telefoonnummer. Een landcode bestaat maximaal uit 5 nummers. Dit baseer ik op de landcode nummers die zijn opgegeven in de prijslijst van Priority Telecom. Voor de prijslijst van Priority Telecom verwijs ik u naar de bijlage van dit document. Het is niet duidelijk hoeveel nummers het uit het buitenland bevat, dus reserveer ik 18 nummers voor de netnummer en de telefoonnummer. Als men bijvoorbeeld naar een land als China of India kijkt met een inwoneraantal van 1.3 en 1 miljard, lijkt het mij verstandig om een ruime marge te nemen. Mocht het ooit voorkomen dat alle inwoners van het land China een telefoonnummer zouden hebben, dan zullen er ook geen problemen komen met de software.

Voor het uitvoeren van het vulprogramma is slechts 1 minuut en 15 seconde nodig om 75.000 tuples willekeurige getallen te genereren en weg te schrijven naar de database.

Na het vullen van de database met het vulprogramma die ik heb gemaakt en een aantal tarieven toe te passen binnen de tarieven en starttarieven tabellen, kan ik nu de performance meten.
Ik roep de PL/pgSQL script aan door deze binnen SQL in te voeren en vervolgens voer ik het volgende selectie query uit:

```
SELECT
  calldate,
  src,
  dst,
  billsec,
  get_call_cost(oid)
FROM cdr;
```

Binnen de query wordt de functie aangeroepen die in PL/pgSQL is geschreven met behulp van de volgende regel:

```
get_call_cost(oid)
```

Het resultaat van deze query met PL/pgSQL functie kostte 46 minuten om te voltooien op een tabel met 75.000 tuples. Dit overschrijdt de tijd voor het verrekenen van de gesprekskosten per maand en project met MIND CMS PhonEx. Hierdoor ben ik genoodzaakt om een alternatieve oplossing te vinden voor het verrekenen van de gesprekskosten.

11.4 Workaround

Middels deze architectural spike ben ik erachter gekomen dat het verrekenen van de gesprekskosten op een andere manier moet worden opgelost dan met behulp van PL/pgSQL. De toepassing dient anders gerealiseerd te worden ter verbetering van de performance.
De reden dat dit 46 minuten duurt is omdat de PL/pgSQL script een reeks van queries uitvoert om de kosten te berekenen. Voor ieder tuple, worden 5 queries op de tabellen cdr, tarieven en starttarieven uitgevoerd.

Hieronder heb ik de queries die worden uitgevoerd voorzien van commentaar met behulp van de # teken en een getal:

```
BEGIN
-- Ken de gegevens uit de database toe aan de variabelen
binnen de functie
#1SELECT INTO call_billsec billsec
    FROM cdr
    WHERE oid = call_oid;
  IF call_billsec IS NULL THEN
    RAISE NOTICE ''call_duration is null'';
    RETURN NULL;
  END IF;
#2SELECT INTO call_netnummer SUBSTR(dst, 1, 3)
    FROM cdr
    WHERE oid = call_oid;
#3SELECT INTO call_tarief tarief
    FROM tarieven
    WHERE prefix = call_netnummer;
  IF call_tarief IS NULL THEN
    RAISE NOTICE ''Geen tarief gevonden voor: %.'',
call_netnummer;
    RETURN NULL;
  END IF;
#4SELECT INTO call_startcode startcode
    FROM tarieven;
#5SELECT INTO call_starttarief tarief
    FROM starttarieven
    WHERE call_startcode = startcode;
```

Men kan er dus vanuit gaan dat er 5 * 75.000 queries uitgevoerd moeten worden om de gemiddelde gesprekskosten te kunnen berekenen per maand en project.
Om een andere oplossing te vinden voor het berekenen van de gesprekskosten die sneller verloopt dan MIND CMS PhonEx, heb ik met behulp van zoekmachines op het internet getracht te zoeken. Zonder resultaat, want ik belandde in een web van

commerciële sites die complete kostenregistratiesystemen verkochten i.p.v. technologieën die hiervoor gebruikt werden.
Omdat ik raad nodig had heb ik het hoofd van afdeling ICT 'J. Vermeer' gevraagd of hij implementaties kende op het gebied van berekeningen uitvoeren. Het advies dat hij gaf was dat ik het kon proberen op te lossen met behulp van SQL queries.
Om deze reden ben ik aan de slag gegaan om een SQL query te schrijven die de gesprekskosten berekent.

Om de gesprekskosten te kunnen berekenen heb ik de volgende gegevens nodig: bestemming, de datum waarop het telefoongesprek heeft plaatsgevonden, het aantal seconden waar kosten over wordt verrekend, tarief geldend voor dat nummer en het starttarief.
Deze gegevens zal ik uit meerdere tabellen moeten halen. In dit geval heb ik de gegevens nodig uit de 'cdr' tabel en 'tarieven' tabel. Om vergelijkingen uit te voeren zal ik genoodzaakt zijn om een cartesisch[19] product te creëren bestaande uit de tabel 'cdr' en de tabel 'tarieven'. Dit bereik ik met de volgende regel:

```
cdr LEFT JOIN tarieven
```

Ik heb hier gebruik gemaakt van een 'LEFT JOIN'. De reden dat ik hier een LEFT JOIN gebruik is dat ik alle gegevens uit de tabel 'cdr' wil opnemen in het resultaat, ongeacht of een bijpassende tarief is gevonden.
Nu ik het cartesisch product hebt aangemaakt, wil ik deze tarieven met de gesprekken combineren. Dit los ik op door een voorwaarde te stellen aan het tarief dat geldt voor het gesprek.

```
ON (tarieven_id = tarieven_id)
```

Met de `ON` functie stel ik deze voorwaarden en kan ik de primaire sleutel vergelijken met het desbetreffende tarief die voor het gesprek telt.
Door gebruik te maken van een subselect kan ik de vergelijking uitvoeren.

```
SELECT tarieven.tarieven_id
FROM tarieven
```

Nu heb ik de mogelijkheid om een conditie te stellen ter controle van het netnummer in het bestemmingsnummer. Dit doe ik met behulp van de funtie `POSITION`. Deze functie bepaalt de plaats van de ene string binnen de andere string. Als de positie van tarieven.prefix(netnummer) in cdr.dst(bestemmingnummer) 1 is, betekent het dat bestemmingsnummer met het netnummer begint. Deze conditie heb ik als volgt opgesteld:

```
WHERE POSITION(tarieven.prefix IN cdr.dst) = 1
```

Naast deze conditie moet ik de tarieven controleren op de verschillende tarieven die voor een netnummer kunnen gelden. Zoals in de uitgangsituatie beschreven, worden er verschillende tarieven voor dagdelen gehanteerd. Het kan het zijn dat er een nacht-, dal-, piek- en avondtarief geldt. Ter uitbreiding van de conditie stel ik het volgende:

```
AND (begintijd IS NULL
  OR (CAST(cdr.calldate AS time) > tarieven.begintijd
    AND CAST(cdr.calldate AS time) < tarieven.eindtijd))
```

[19] Cartesisch product – het product van een join (samengevoegde tabel).

Eerst kijk ik of er een begintijd is ingevuld, zo niet: hanteer de tarief voor alle gesprekken naar dit nummer waar geen tijden voor zijn ingevuld. Het belangrijkste is dat er wordt gekeken of het tijdstip van het telefoongesprek binnen de begin en eindtijd valt. Dit doe ik door de conditie te stellen dat het tijdstip een grotere waarde heeft dan de begintijd van een tarief en een kleinere waarde heeft dan de eindtijd van een tarief.

Om het tijdstip van het gesprek te gebruiken, in plaats van de hele datum, dien ik de datum om te zetten naar een tijdstip. Dit doe ik met de functie `CAST`. Door de datum mee te geven als parameter aan deze functie en het gewenste formaat (`TIME`) aan te geven, krijgen we als resultaat een tijdstip, bijvoorbeeld 00:00:00.
Het probleem waar ik vervolgens mee kampte was de handeling van de DBMS die onjuiste tarief opvraagt. De reden hiervoor is dat er verschillende tarieven kunnen zijn die beide beginnen met hetzelfde nummer. Om een zo'n specifiek mogelijk tarief hebben, zal ik de resultaten ordenen met behulp van de `ORDER BY` clausule. Ik orden het volgende:

```
ORDER BY
   prefix DESC,
   begintijd IS NULL ASC,
   (begintijd - eindtijd) DESC,
   begintijd ASC
LIMIT 1;
```

Door het netnummer die het meest specifiek is(meeste nummers heeft) te ordenen op een aflopende wijze, krijg ik de netnummer met het hoogste waarde. Dit gebeurt met de commando `DESC`, wat voor descending(aflopend) staat.
Tevens wil ik proberen tarieven op te halen met dat een begintijd gedefinieerd heeft gekregen. Door een oplopende ordening, ASC wat voor ascending staat, zorg ik ervoor dat de begintijd(en) die geen waarde bevatten, onderaan de resultaten terecht komen.
Om de meest specifieke tijden te hanteren zorg ik ervoor dat ik begintijd van eindtijd aftrek en deze aflopend weergeef. Een voorbeeld hiervan:
Dagdeel 1

```
Begintijd 12:00:00
Eindtijd  14:00:00
```

Dagdeel 2

```
Begintijd 08:00:00
Eindtijd  20:00:00
```

Zoals men kan zien heeft dagdeel 1 specifiekere tijden ingesteld gekregen dan dagdeel 2. Middels dit aflopend te ordenen krijg ik het tarief met de meest specifieke ingestelde tijd. Dit gaf nog steeds niet correcte tuples met daarin de juiste waarden terug, dus heb ik nog moeten aangeven dat een tarief dat eerder begint, voorrang heeft op een tarief dat later begint.
Nadat het meest specifieke tarief en de tarieftijden zijn gerangschikt, wil ik deze gebruiken als tarief voor het gesprek. Door het '`LIMIT 1`' commando op te geven, wordt er slechts één rij teruggegeven die ik ga gebruiken als subselect waarde om mee te vergelijken.

De gegevens die ik wil laten weergeven zijn Bron, Bestemming, Datum, aantal seconden waar gesprekskosten over mag worden verrekend en de gesprekskosten

van het gesprek. Deze gegevens verkrijg ik door de volgende gegevens op te vragen:

```
SELECT
  cdr.src,
  cdr.dst,
  cdr.calldate,
  cdr.billsec,
  tarieven.starttarief + tarieven.tarief * billsec AS kosten
```

Uiteindelijk heb ik dit in één query kunnen verwerken om zo de gesprekskosten te berekenen. Hieronder de query die de gesprekskosten berekent en het meest specifieke tarief hanteert:

```
SELECT
  cdr.src,
  cdr.dst,
  cdr.calldate,
  cdr.billsec,
  tarieven.starttarief + tarieven.tarief * billsec AS kosten
FROM
  cdr LEFT JOIN tarieven
    ON (tarieven.tarieven_id =
      (SELECT tarieven.tarieven_id
       FROM tarieven
       WHERE POSITION(tarieven.prefix IN cdr.dst) = 1
         AND (begintijd IS NULL
           OR (CAST(cdr.calldate AS time) > tarieven.begintijd
             AND CAST(cdr.calldate AS time) <
tarieven.eindtijd))
       ORDER BY
         prefix DESC,
         begintijd IS NULL ASC,
         (begintijd - eindtijd) DESC,
         begintijd ASC
       LIMIT 1));
```

Deze SQL query heb ik op de tabel CDR uitgevoerd die 75.000 records bevatte. De duur van het berekenen en weergeven van de gesprekskosten waren slechts enkele seconden.

12. Implementatie

In dit hoofdstuk beschrijf ik de activiteiten die hebben plaatsgevonden tijdens het implementeren van de database en de software. Ook zal duidelijk worden waar ik tegen problemen op ben gelopen en welke stappen ik heb genomen om dit op te lossen.

12.1 Database realisatie

Na het ontwerpen van de database ga ik de database realiseren door verschillende tabellen aan te maken.
Ik ga de volgende tabellen gebruiken; cdr, tarieven en kostenloze_tijden.
Hieronder de realisatie van de databaseopdrachten om deze tabellen aan te maken:
Afgeleid van het databaseontwerp volgt hier de realisatie van de tabel ‘cdr’:

```
CREATE TABLE cdr (
  call_id SERIAL PRIMARY KEY,
  calldate TIMESTAMP NOT NULL,
  src VARCHAR(20) NOT NULL,
  dst VARCHAR(20) NOT NULL,
  billsec INT NOT NULL
);
```

Voor het realiseren van de tarieven tabel heb ik een aantal condities gesteld aan de gegevens die in de velden komen. De CHECK functie wordt toegepast op de velden prefix en begin- eindtijd.

```
CREATE TABLE tarieven (
  tarieven_id SERIAL PRIMARY KEY,
  prefix varchar(20) NOT NULL
    CHECK(prefix ~ '^[0-9]+$'),
  begintijd TIME,
  eindtijd TIME,
  tarief FLOAT NOT NULL DEFAULT '0.0',
  starttarief FLOAT NOT NULL DEFAULT '0.0',
  CHECK(begintijd IS NULL = eindtijd IS NULL),
  CHECK(begintijd < eindtijd)
);
```

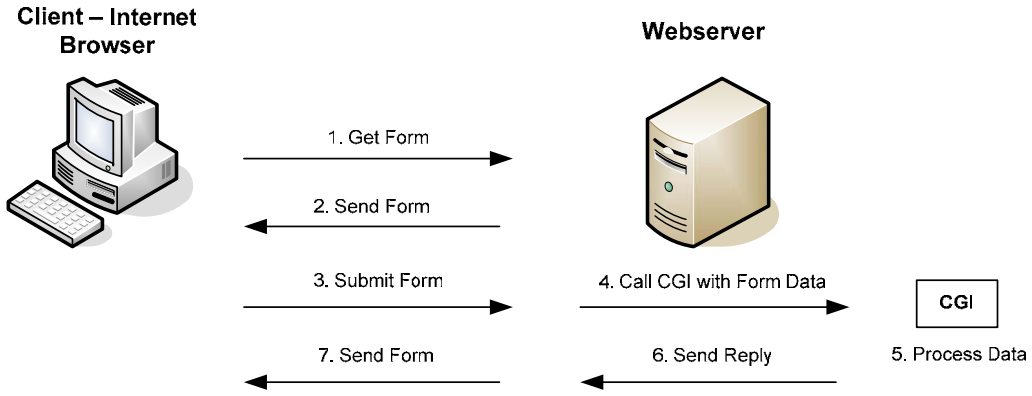
Tot slot de tabel ‘kostenloze_tijden’, waarmee de use case ‘kostenloze tijden instellen’ wordt gerealiseerd. In dit datamodel heb ik gebruik gemaakt van ‘timestamp’ velden. Dit betekent dat het alleen mogelijk is om een datum en/of tijd in te voeren, maar geen andere waarde. Zo wil ik voorkomen dat er ongeldige waarden ingevoerd kunnen worden en dat de software de ongeldige waardes niet kan verwerken. In de onderstaande broncode toon ik de wijze waarop ik de tabel ‘kostenloze_tijden’ implementeer:

```
CREATE TABLE kostenloze_tijden (
  tijd_id SERIAL PRIMARY KEY,
  startdate TIMESTAMP,
  einddate TIMESTAMP,
PRIMARY KEY(naam)
);
```

De reden om '`naam`' als primaire sleutel op te nemen is dat er verschillende tijden kunnen zijn waar verschillende tarieven voor gelden. Een voorbeeld hiervan zijn feestdagen. Op feestdagen wilt men geen kosten verrekenen voor alle uitgaande gesprekken binnen de regio en binnen het netwerk van Lijbrandt Telecom.
Om men de keuze te geven om meerdere kostenloze tijden toe te kennen binnen de software, voldoe ik aan de beschrijvingen van de use case 'kostenloze tijden instellen'.

12.2 Common Gateway Interface

Zoals ik eerder in het document had vermeldt, zal ik gebruik maken van een interface. Voor deze interface wil ik gebruik maken van CGI. Dit staat voor Common Gateway Interface en zorgt voor het leveren van gegevens van een HTML form.
Op afbeelding 12.1 'Schema CGI' heb ik een illustratie gebruikt uit het boek 'Teach Yourself Perl in 21 Days' en ziet u hoe het proces verloopt tussen een webbrowser en een webserver met behulp van CGI.

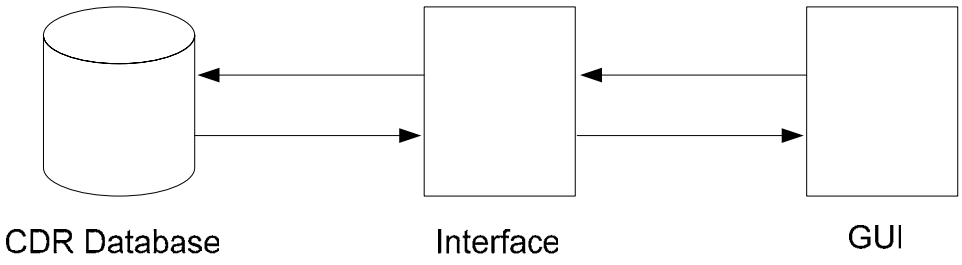


Afbeelding 12.1 'Schema CGI'

Vanaf een computer met een internet-browser is het mogelijk om gegevens op te halen van een webserver en gegevens te versturen naar een webserver. Op deze wijze benadert de webserver de database waarop de gespreksgegegevens zich bevinden. Vervolgens stuurt de webserver de nodige gegevens naar de cliënt.
Met een interface wil ik de database kunnen benaderen en acties uit laten voeren om zo gegevens terug te krijgen van de database. Een interface kan men dus ook zien als een aanspreekpunt voor een GUI.

De interface zal ik in de programmeertaal Perl schrijven. Om gebruik te kunnen maken van CGI, dient men deze module te laden binnen de Perl-script. Perl is een procedurele taal die veel gebruikt wordt voor serveroplossingen.
In mijn vooropleiding heb ik een kennismaking gehad met deze programmeertaal en heb ik basiselementen geleerd. Dit was slechts in beperkte mate, zodat ik genoodzaakt was om mijzelf het programmeren in Perl aan te leren tijdens dit softwareontwikkelingtraject. Ik heb daarom ook rekening gehouden met mijn planning om wat meer tijd toe te kennen aan het programmeren van de use cases.

Hieronder in overzicht 12.2.1 ‘Interface functie’ ziet u hoe de interface gaat functioneren tussen de database en de GUI:



Afbeelding 12.2.1 ‘Interface functie’

Voor het verbinden met de database heb ik gebruik gemaakt van het boek ‘Perl Cookbook’ van O’Reilly. In dit boek staan voorbeelden van onderdelen beschreven, waaronder het verbinden met een database.
Middels de module DBI, dat met databases kan verbinden. De reden dat ik gekozen heb om met de DBI module een verbinding met de database vast te stellen is omdat de DBI module transparant met verschillende DBMS’en kan werken. Dit houdt in dat de DBI module weet hoe men om moet gaan met verschillende DBMS’en door aan te geven welke DBMS men gebruikt.
Bijvoorbeeld, de handle(opdracht) om te kunnen verbinden met een database luidt: `DBI->connect`. Zo vereist deze handle de volgende gegevens: Driver(MySQL, Oracle, PostgreSQL, etc), Databasenaam, gebruiker en wachtwoord.
In mijn context moest ik gebruik maken van het volgende:

```
DBI->connect(‘dbi:pg:asterisk’, ‘asterisk’, undef,
      { RaiseError => 1, AutoCommit => 1});
```

Zo gebruik ik `‘pg’` om aan te duiden dat ik gebruik maak van een PostgreSQL database en `‘asterisk’` dat de database asterisk heet. Verder geef ik de gebruikersnaam en wachtwoord op om toegang te krijgen tot de database.
Tot slot gebruik ik `RaiseError` om foutmeldingen te zien indien er fouten voorkomen. Met `AutoCommit` laat ik iedere SQL actie automatisch committen.
Om te achterhalen of het programma daadwerkelijk verbonden is met de database heb ik het volgende stuk geschreven:

```
use DBI;
$dbh = DBI->connect('dbi:pg:asterisk', 'root', undef,
        { RaiseError => 1, AutoCommit => 1});
print "Connected to database!”;
$dbh->disconnect;
```

Met behulp van de databaseverbinding is het nu mogelijk om verschillende acties uit te voeren op de database.

12.2.1 Tarieven instellen

Voor de iteratie ‘Tarieven instellen’ ben ik begonnen met het programmeren in combinatie met het maken van de templates. Dit omdat er knoppen binnen de GUI zijn waar de interface op reageert. Een voorbeeld hiervan is de broncode die ik heb ontwikkeld indien op de knop ‘toevoegen’ wordt gedrukt. Met een `IF` conditie controleer ik of hier sprake van is. Indien dit het geval is, definieer ik de parameters prefix, begintijd, eindtijd, tarief en starttarief aan mijn variabelen binnen mijn script.

```
if(defined param("toevoegen")){
  my $prefix = param("prefix");
  my $begintijd = param("begintijd");
  my $eindtijd = param("eindtijd");
  my $tarief = param("tarief");
  my $starttarief = param("starttarief");
```

Omdat ik de parameter van begin- en eindtijd toeken aan mijn variabelen, betekent dit dat ze een lege string bevatten en geen null waarde hebben. Dit leverde fouten op bij het wegschrijven van de tariefgegevens naar de database, omdat ik in het datamodel ‘tarieven’ heb opgenomen waar de attributen `begintijd` en `eindtijd` alleen geldige tijdsnotering mogen hebben. Om dit probleem te vermijden doe ik het volgende:

```
undef $begintijd if $begintijd eq "";
undef $eindtijd if $eindtijd eq "";
```

Hierin ondefinieer ik de variabalen `$begintijd` en `$eindtijd`. Dit heft de toekenning van een lege string op.
Vervolgens geef ik de waardes mee aan een query die wordt uitgevoerd op de database. Het betreft hier een `INSERT` query, die de gegevens wegschrijft naar de database. De parameters worden binnen de query opgegeven als vraagtekens. De waarden van deze vraagtekens geeft men later in de constructor van de functie ‘do’ op. In mijn geval zijn dit de variabelen die ik als parameter heb meegekregen vanuit de webpagina. Tevens leid ik de gebruiker terug naar de ‘Tarieven instellen’-pagina.

```
$db->do("
  INSERT INTO tarieven (prefix, begintijd, eindtijd,
tarief, starttarief)
    VALUES (?, ?, ?, ?, ?);", {},
    $prefix, $begintijd, $eindtijd, $tarief,
$starttarief);
  print redirect("tarieven_instellen.pl");
}
```

De klanteis om tarieven te kunnen verwijderen is ook opgenomen in deze iteratie. Dit gebeurt middels de template die voor ieder gevonden tarief een ‘verwijder’ knop afbeeld. Wanneer er op deze knop wordt gedrukt, dan zal de interface deze tarieven_id verwijderen uit de database en vervolgens de gevonden tarieven weer afbeelden.
Ik realiseer dit door een loop te creëren die controleert welke knop er is ingedrukt. In de template geef ik ieder result een knop, zodat deze verwijderd kan worden. De parameters die zijn meegegeven vanuit de pagina, worden doorlopen in de onderstaande broncode:

```
my $verwijder_id;
for my $param (param()) {
  if(index($param, "verwijder_") == 0) {
```

```
        $verwijder_id = substr($param, length("verwijder_"));
    }
}
```

Binnen de broncode definieer ik de tarieven_id aan de variabele `$verwijder_id`. De variabele `$verwijder_id` gebruik ik vervolgens weer binnen de query. Hieronder de actie indien er op de verwijderknop is gedrukt ($verwijder_id is gedefinieerd).

```
elsif ($verwijder_id) {
  $db->do("DELETE FROM tarieven WHERE tarieven_id = ?",
{}, $verwijder_id);
  print redirect("tarieven_instellen.pl");
}
```

Bij het openen van deze webpagina dienen de tarieven op het scherm worden afgebeeld. Ik maak een array `$data{'tarieven'}` aan. Deze array wordt met de waarden gevuld die ik opvraag met behulp van de functie `select_arrayref`. Deze gegevens worden meegegeven aan het template, die de array aanspreekt om de gegevens op het scherm af te beelden. Hieronder de broncode waarmee de array 'tarieven' aanmaakt en deze vult met gegevens uit de database:

```
$data{'tarieven'} = $db->selectall_arrayref("
  SELECT tarieven_id, prefix, begintijd, eindtijd,
    tarief, starttarief
  FROM tarieven
  ORDER BY prefix DESC, begintijd IS NULL, (begintijd -
    eindtijd) DESC, begintijd", {Slice => {}});
```

Op de laatste regel wordt het resultaat opgesplitst per attribuut die is opvraagd uit de database.

12.2.2 Gesprekskosten van een project per maand opvragen

Voor deze iteratie stel ik de conditie dat men een maand en een jaartal moet hebben geselecteerd. Pas na het tegemoetkomen aan deze criteria, dan zal de interface communiceren met de database.

De usecase-beschrijving van de use case 'Gesprekskosten van een project per maand opvragen' wordt nageleefd, doordat ik binnen de broncode een conditie stel. Voordat de gesprekskosten van een project per maand wordt berekend is men genoodzaakt om een maand te selecteren uit de dropdown-list en een jaartal in het invoerveld op te geven.
Indien men op de knop 'opvragen' drukt, dan geef ik de parameters mee van het selectveld 'maand' en invoerveld 'jaar' die zijn opgegeven en ken ik deze toe binnen het script. Hieronder de variabelen die de parameters krijgen toegekend:

```
my $maand = param("maand");
my $jaar = param("jaar");
```

Pas wanneer beide zijn ingevuld, kan de interface de database benaderen om de gegevens op te vragen. Deze conditie ken ik toe aan de variabele $data{'query_opgegeven'}, omdat ik deze gegevens ook wil gaan gebruiken binnen de webpagina's. Ik zal hier meer over vertellen in hoofdstuk 11.3.4 Template Toolkit. Hieronder controleer ik of de variabelen `$maand` en `$jaar` zijn gedefinieerd, indien dit het geval is dan ken ik de waarde 1 toe aan deze variabele.

```
    $data{'query_opgegeven'} = 1 if defined $maand && defined
    $jaar;
```

De gesprekskosten kunnen opgevraagd worden met behulp van een SQL query die ik in hoofdstuk 10.3 'Work around' heb beschreven. Middels deze workaround zal ik de gesprekskosten opvragen binnen in de iteratie. Met de volgende broncode haal ik de gegevens op van de gesprekskosten:

```
    if($data{'query_opgegeven'}){
      $data{'cdr'} = $db->selectall_arrayref("
        SELECT src, dst, billsec, calldate, billsec,
          tarieven.starttarief + tarieven.tarief * billsec AS
    kosten
        FROM
          cdr LEFT JOIN tarieven ON (tarieven.tarieven_id =
            ( SELECT tarieven.tarieven_id
              FROM tarieven
              WHERE POSITION(tarieven.prefix IN cdr.dst) = 1
                AND (begintijd IS NULL
                  OR (CAST(cdr.calldate AS time) >
    tarieven.begintijd
                    AND CAST(cdr.calldate AS time) <
    tarieven.eindtijd))
              ORDER BY prefix DESC, begintijd IS NULL,
    (begintijd - eindtijd) DESC, begintijd
              LIMIT 1))
```

De query moet nog een extra conditie stellen om alleen de gesprekskosten op te vragen van de opgegeven jaar en maand.
Hieronder de conditie binnen SQL:

```
    WHERE EXTRACT(YEAR FROM calldate) = ?
      AND EXTRACT(MONTH FROM calldate) = ?", {Slice => {}},
    $jaar, $maand);
}
```

12.2.3 Gesprekskosten exporteren naar CSV of XML bestandformaat

Wegens tijdgebrek ben ik hier slechts deels aan toe gekomen.

12.2.4 Kostenloze tijden instellen

Wegens tijdgebrek ben ik niet aan deze iteratie toegekomen.

12.3 Grafische User Interface

Voor het implementeren van de GUI heb ik gebruik gemaakt van diverse technieken zoals bijvoorbeeld XHTML 1.1, template toolkit en CSS. In de volgende paragrafen vindt U meer informatie over deze desbetreffende onderwerpen.

12.3.1 Apache 2.0 webserver

Om webpagina's te kunnen weergeven op internet-browsers van verschillende locaties(niet lokaal) heb ik eerst een webserver moeten opzetten. Er zijn

verschillende softwarepakketten die dit kunnen. De reden dat ik Apache http webserver 2.0 gebruik is omdat deze software gebruik is omdat het standaard met de Fedora Core Linux distributie geleverd wordt.
De locatie van de computer waarop ik deze software heb geïnstalleerd is de server die ik gebruik voor de Asterisk PBX server en het wegschrijven van de verbruiksgegevens.

Eerst dien ik de HTTP server service te starten. Dit doe ik met behulp van `httpd` in de map `/etc/init.d/`. De correcte syntax om deze service te starten is:

```
#/etc/init.d/httpd start
```

Na het starten van de service is het mogelijk om de webserver te benaderen. Om dit te controleren heb ik een HTML bestand gemaakt en deze geplaatst in de directory `/var/www/html/`. Dit is de zogenaamde DocumentRoot, het beginpunt, waar de webserver op zal draaien. Door nu het IP adres van de webserver in een internet browser in te voeren controleer ik of de webserver werkt. De HTML pagina die ik in de root had geplaatst, was te zien in mijn internet browser.

12.3.2 XHTML

De GUI zal ik maken met behulp van XHTML 1.1(eXtensible Hyper Text Markup Language). Dit is een standaard waar internetpagina's worden weergegeven. XHTML is de opvolger van de serie HTML 4. Door gebruik te maken van XHTML in plaats van HTML voor de GUI, is de kans kleiner dat er problemen zijn met het weergeven van de internetpagina's in verschillende browsers zoals bijvoorbeeld Internet Explorer, Netscape Navigate, Mozilla Firefox, etc. Ook is XHTML ontworpen om met XML-gebaseerde talen, applicaties en protocollen te communiceren.
Om aan internet-browsers bekend te maken dat mijn pagina's volgens de XHTML 1.1 standaard is opgesteld dien ik de volgende regels op te geven binnen iedere internetpagina:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="nl">
```

Een aantal kenmerken van XHTML is dat het bijvoorbeeld case sensitive is, in tegenstelling tot HTML 4. Om het begrijpbaar te laten maken voor internet-browsers heb ik de internetpagina's in 'lower case' moeten typen.

12.3.3 CSS

De websites van Lijbrandt Telecom zijn gemaakt met behulp van CSS. Dit staat voor Cascading Style Sheet en zorgt voor de vormgeving van HTML pagina's.
De internetadministratie van Lijbrandt Telecom waar de afdeling ICT en Technisch beheer toegang tot hebben, maakt gebruik van een CSS.
Omdat ik het uiterlijk van de GUI een herkenbare omgeving wil geven, ga ik CSS toepassen op mijn HTML pagina's. Dit wil ik bereiken door gebruik te maken van het huidige CSS bestand die voor de internetadministratie website wordt gebruikt. Deze heb ik kunnen ophalen van de webserver waarop de internetadministratie gehost wordt.
Ik heb het bestand hernoemt en op mijn webserver geplaatst. Om nu gebruik te kunnen maken van de CSS bestand dient men in de HTML pagina's een verwijzing op te geven binnen het `head` gedeelte. In het `head` gedeelte is het mogelijk om de

content-type, de titel van de pagina, het padnaam van een css bestand, java scripts en dergelijke op te geven. Hieronder de regel voor het opgeven van de verwijzing naar het css-bestand:

```
<link rel="stylesheet" href="css/kosten_generator.css"
type="text/css" />
```

In het CSS bestand worden de achtergrondkleur, kleuren van invoervelden, knoppen, tekstgrootte en tabelformaten gedefinieerd. Ik heb het css-bestand dermate aangepast om alleen de opmaak te definiëren van de componenten die ik ga gebruiken.
Hieronder de broncode van de css-bestand die ik zelf ook zal gebruiken voor de GUI van mijn software:

```
body, table, input {
  background-color: #d4d0c8;
  font-family: Verdana, Arial, Helvetica, sans-serif;
  font-size: 8.25pt;
}

input {
  background-color: #ffffff;
}

.button {
  background-color: #d4d0c8;
}

h1, h2 {
  font-size: 9.25pt;
}

.table th {
  font-weight: normal;
  text-align: left;
  border: 1px solid #808080;
  background-color: #85aee0;
}

.table {
  clear: both;
  border-collapse: collapse;
  border: 1px solid #000000;
  width: 100%;
}

td {
  background-color: inherit;
  font-size: inherit;
}
```

12.3.4 Template toolkit

Een ander hulpmiddel die ik zal gebruiken voor het schrijven van mijn software is Template Toolkit. Dit is software die geschreven is in de programmeertaal Perl voor het hanteren van templates. In de Template Toolkit wordt gebruikt gemaakt van verschillende condities zoals de IF, ELSE, ELSIF en FOR conditie. Met behulp van deze voorwaarden is het mogelijk om een dynamische website te creëren en redundantie te voorkomen.

Een voorbeeld van redundantie is het verwerken van broncode binnen XHTML 1.1 die altijd voorkomt en slechts afwijken van elkaar op enkele zinnen/regels.

Met behulp van de Template Toolkit heb ik een zogenaamde 'header' en een 'footer' gemaakt. Dit zijn blauwdrukken die ik in verschillende pagina's blijf gebruiken. Hieronder de broncode van het bestand "`header.html.templ`":

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="nl">
<head>
  <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8"/>
  <title>Kostenregistratie - [% title %]</title>
     <link rel="stylesheet"
href="css/kosten_generator.css" type="text/css" />
</head>
<body>
<h1>[% title %]</h1>
```

In de broncode hierboven ziet u twee maal `[% title %]` verwerkt. Dit is een onderdeel van de Template Toolkit, dat staat voor een meegegeven parameter, in dit geval genaamd 'title'. Men kan met behulp van deze parameters de internetpagina's dynamisch maken en voor meerdere internetpagina's gebruiken.
De 'footer' bevat slechts afsluitende tags om een geldige XHTML pagina te vormen. Hieronder de broncode van `footer.html.templ`:

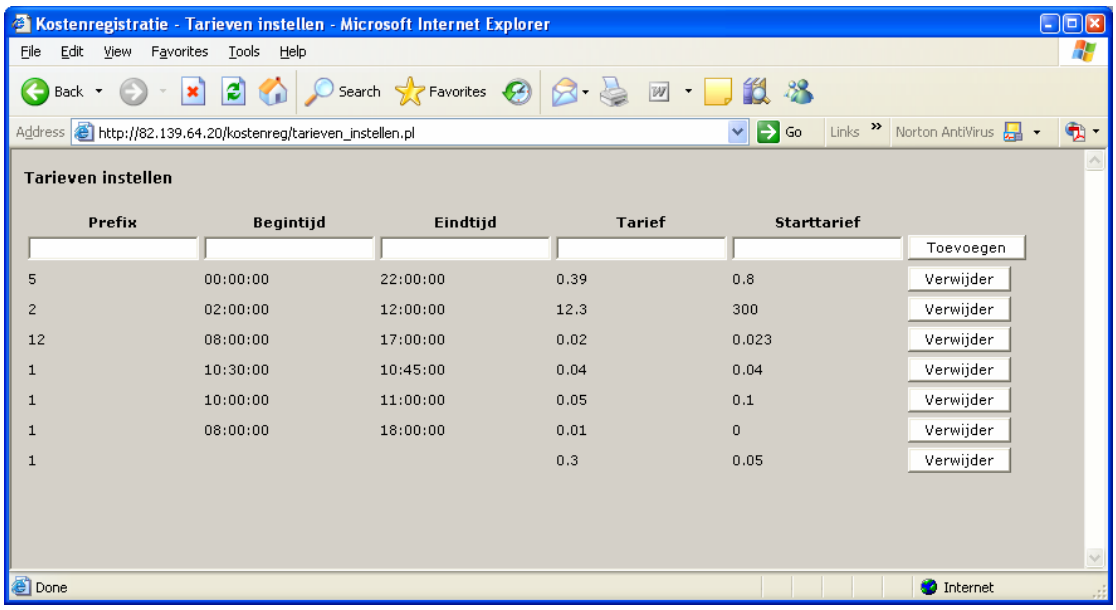
```
  </body>
</html>
```

Ik heb deze pagina's als bestandsnaam `.templ` opgegeven om zo aan te geven dat het hier om een XHTML bestand gaat, waarin gebruik wordt gemaakt van de functionaliteiten van de template toolkit. Voor de broncode van de template verwijs ik u naar de bijlage van dit document.

Het is mogelijk om zelf variabel te definiëren, om zo specifieke waardes mee te geven. Ik ken aan de variabele 'title' de waarde toe: Tarieven instellen. Dit doe ik omdat ik deze variabele aanroep binnen de header en zo deze waarde kan gebruiken voor het weergeven van de titel van de pagina.
Doormiddel van de `[% INCLUDE %]` functie kan ik internetpagina's toevoegen binnen deze pagina. Dit wil zeggen dat ik gebruik kan maken van de `header.html.templ` en de `footer.html.templ` om zo te voorkomen dat ik dit voor iedere pagina moet typen.
Tevens maak ik gebruik van de [% FOREACH %] functie om meerdere resultaten weer te geven. Met behulp van deze functie is het mogelijk om een loop[20] te doorlopen indien er wordt voldaan aan een specifieke conditie.
Het moet mogelijk zijn om nieuwe tarieven te kunnen instellen, dus is het ook essentieel om te weten of het desbetreffende tarief voor het nummer al is ingesteld.

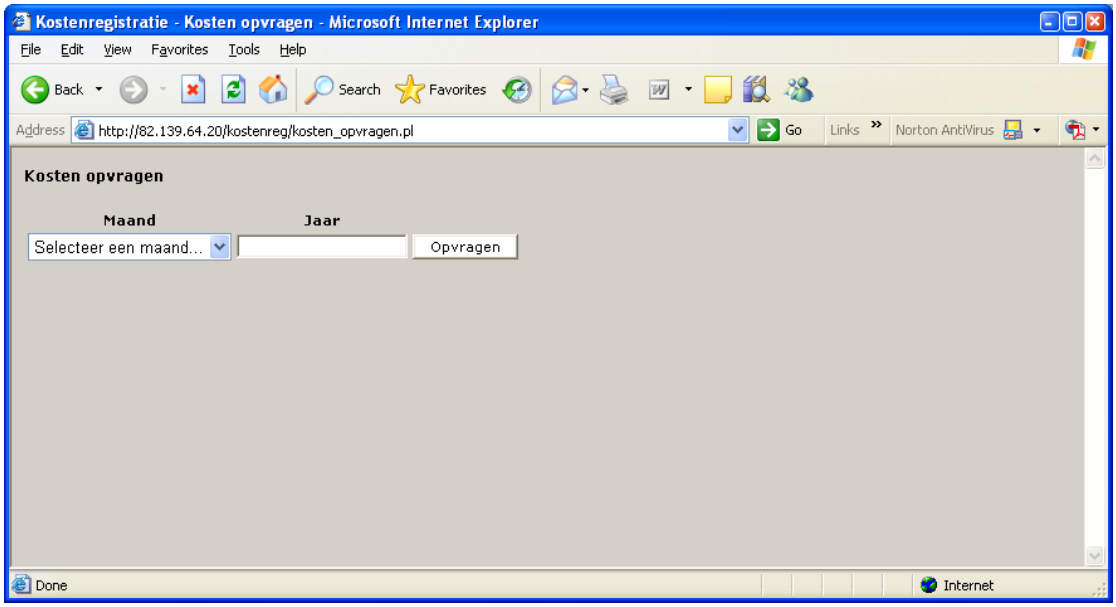
[20] Loop - Iteratie die wordt doorlopen

Om hierachter te komen geef ik de resultaten weer van de tarieven die zijn ingesteld, in de vorm van een lijst. Door een aantal tarieven in te stellen heb ik het volgende scherm gerealiseerd:



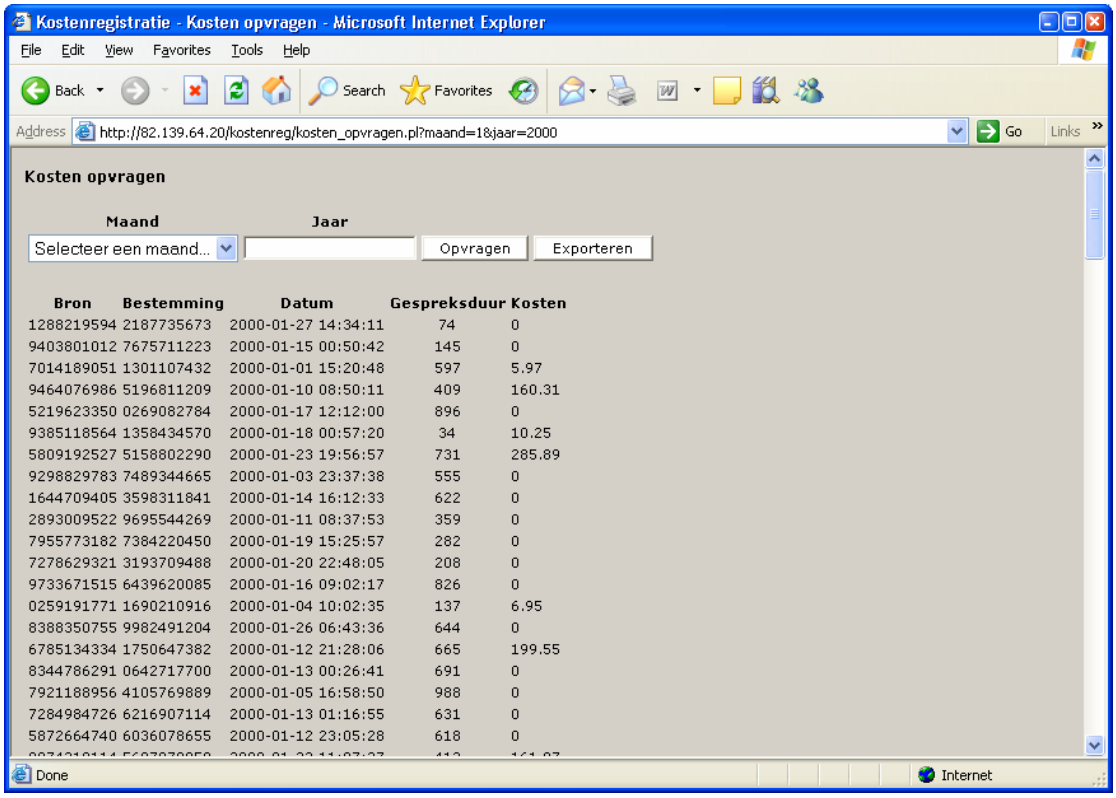
Afbeelding 12.3.4.1 'Momentopname – Tarieven instellen'

Op het scherm van 'Kosten opvragen' heb ik een keuze item met daarin alle maanden. Iedere keuze ken ik een waarde toe om deze als parameter mee te geven aan de interface. Ik heb bewust de waarden 1 tot en met 12 opgegeven, dit omdat de interface dan simpelweg het nummer van de maand kan verwerken binnen zijn query.
Hieronder het scherm waar de iteratie 'Gesprekskosten van een project per maand opvragen':



Afbeelding 12.3.4.2 'Momentopname – Kosten opvragen'

Pas nadat men een maand en jaar heeft ingevoerd, krijgt men een overzicht te zien van de gesprekskosten van dit project. Op afbeelding 12.3.4.3 ‘Momentopname – Kosten berekend’ ziet u een momentopname van de software ‘Kosten opvragen’.



Afbeelding 12.3.4.1 ‘Momentopname – Tarieven instellen’

13. Acceptatietesten

Binnen Extreme Programming worden de acceptatietesten uitgevoerd als blackbox-testen. Dit houdt in dat de iteraties worden getest op functionaliteit. Een user story is pas compleet indien deze de acceptatietesten hebben doorstaan.
De klant is verantwoordelijk voor het keuren van de acceptatietesten en dient de prioriteit aan te geven aan de iteraties die niet voldoen aan de acceptatietesten.

13.1 Tarieven instellen

Na het realiseren van de user story ‘Tarieven instellen’, hebben de opdrachtgever en ik samen de acceptatietest doorlopen. Na het invoeren van een netnummer, begintijd, eindtijd, tarief en starttarief hebben we een nieuw tarief aangemaakt door deze toe te voegen. De webpagina werd herladen en toonde direct het tarief die we zojuist hadden ingevoerd. De webpagina kan via de webserver vanaf iedere locatie benaderd worden.
Hiermee is aan de volgende klanteisen voldaan:

- Het systeem moet verschillende tarieven kunnen hanteren voor regionaal, buiten de regio, internationaal, mobiele en 0900 informatie- nummers.
- Het instellen van tarieven moet toegankelijk zijn vanaf iedere locatie.
- De tarieven moeten verwijderd kunnen worden.

De opdrachtgever heeft deze acceptatietest goedgekeurd.

13.2 Gesprekskosten van een project en per maand opvragen

Tijdens het maken van een prototype ten behoeve van de architectural spike, was ik genoodzaakt om een workaround te ontwikkelen. Dit hield in dat ik een alternatieve oplossing moest bedenken voor het berekenen van de gesprekskosten. Het resultaat was een SQL query die dit kon.
Voor het keuren van de acceptatietest, heb ik de opdrachtgever de software laten gebruiken. Zoals ik in de pre-conditie van de use case beschrijving heb opgegeven, dient er eerst een tarief ingesteld te zijn. De reden hiervoor is dat de gesprekskosten niet berekend kunnen worden zonder deze gegevens.
Na het selecteren van een maand en het invoeren van een jaar wordt er met behulp van de ‘opvragen’ knop de gesprekskosten verrekend en op beeld getoond.
Hiermee is aan de volgende klanteis voldaan:

- De software systeem moet sneller de gesprekskosten kunnen berekenen dan met MIND CMS PhonEx.

Zo is er een klanteis die niet met deze use case wordt opgelost. Er wordt aan de klanteis voldaan indien er per project een aparte databaseserver zou zijn aangesteld. Het is dan een kwestie van het opgeven van de locatie van de database om per project de gesprekskosten op te kunnen vragen.

- De tarieven moeten per project en per maand ingesteld worden.

Omdat ik momenteel alleen binnen de testomgeving werk heeft de opdrachtgever deze acceptatietest toch goedgekeurd.

13.3 Gesprekskosten exporteren naar CSV of XML bestandsformaat

Wegens tijdgebrek ben ik niet aan deze iteratie toegekomen. Deze iteratie zou de volgende klanteis vervullen

- De gegevens moeten met de applicatie Exact Globe ingelezen kunnen worden.

13.4 Kostenloze tijden instellen

Wegens tijdgebrek ben ik niet aan deze iteratie toegekomen.

- Men moet aan kunnen geven wanneer men wel en wanneer men geen kosten verrekent.

14. Projectevaluatie

In dit hoofdstuk zal ik de procesgang tijdens de afstudeerperiode en de opgeleverde producten evalueren. Het zal duidelijk worden wat ik heb voldaan op welke punten en welke niet. Dit schrijf ik om te kunnen reflecteren op mijn handelingen binnen dit afstudeerproject en van mijn fouten kan leren door het anders aan te pakken.

14.1 Productevaluatie

De op te leveren producten zijn op de volgende punten bereikt:

VoIP testopstelling	√
Iteratie: Tarieven instellen	√
Iteratie: Kosten opvragen	√
Iteratie: Gesprekskosten exporteren	X
Iteratie: Kostenloze tijden instellen	X

Tabel 14.1.1 'Overzicht productvoortgang'

Tijdens het opstellen van de releaseplan heb ik de volgende prioriteiten toegekend aan de iteraties:

Kostenregistratie	Must have	Should have	Could have	Want to have
Toepassen van tarieven	X			
Toepassen van kostenloze tijden			X	
Gesprekskosten van een project, per maand opvragen		X		
Gesprekskosten exporteren naar bestandformaat		X		

Tabel 14.1.2 MoSCoW-principe 'Kostenregistratie binnen VoIP'

De iteraties 'Tarieven Instellen' en 'Gesprekskosten opvragen' hebben de acceptatietesten doorstaan en zijn uitgebracht als small releases.
De iteratie 'Kostenloze tijden instellen' die onder 'Could have' valt is geen must, maar een pre. Indien er genoeg tijd was, dan had ik hier aan gewerkt maar is komen te vervallen wegens tijdgebrek.
Wat betreft de iteratie 'Gesprekskosten exporteren naar CSV of XML bestandsformaat', was ik niet in staat om deze te realiseren. In dit opzicht kan ik concluderen dat het me niet is gelukt om binnen de gegeven tijd de op te leveren producten, op te leveren.
Wel zou het hoogwaarschijnlijk zijn dat ik de iteratie 'Gesprekskosten exporteren naar CSV of XML bestandsformaat' kan realiseren in de tijd tussen het inleveren van het afstudeerverslag en de zitting.

14.2 Procesevaluatie

In het begin van het traject was het moeilijk om in één lijn te komen met de opdrachtgever. De methode die ik hanteerde botste met wat hij verwachtte en dit veroorzaakte discussies over hoe ik dingen ging aanpakken. ‘Extreme Programming’ schrijft voor om veel te communiceren met de opdrachtgever, meestal op dagelijkse basis. Er waren regelmatig besprekingen tussen mij en de opdrachtgever, maar niet op dagelijkse basis. Het nastreven van de richtlijnen binnen ‘Extreme Programming’ is moeilijk na te leven binnen dit afstudeerproject. Dit komt omdat de opdrachtgever simpelweg niet iedere dag tijd had om besprekingen te houden. Gemiddeld heb ik wekelijkse besprekingen gehad met mijn opdrachtgever.
Tevens speelde er zich een belangenkwestie zich af wat is opgelost door een compromis te sluiten. De opdrachtgever toonde begrip dat mijn afstudeeropdracht een softwareontwikkelingsproject was en gaf in op zijn verwachtingen van een dienstontwikkelingsproject.
Ik heb veel inzicht gekregen in het project dankzij het aandringen van mijn opdrachtgever. Ik zal voortaan meer open staan voor suggesties. Ik vreesde dat mijn project niet beoordeeld zou kunnen worden als ik me niet hanteerde aan één specifiek methode en de richtlijnen van school. Om deze reden heb ik geprobeerd het volledig naar de richtlijnen van school te realiseren. Ik denk dat dit de botsingen veroorzaakte tussen mij en de opdrachtgever. Door aan beide kanten te geven en te nemen heb ik een compromis kunnen sluiten met de opdrachtgever wat het werk een stuk aangenamer maakte.

Omdat ‘Extreme Programming’ user stories hanteert voor het achterhalen van de klanteisen en dat iedere user storie slechts uit een aantal zinnen moeten bestaand, is het een probleem geworden op het gebied van technische specificaties. Vooral op het gebied van de performance van de software, dat een groot vlak aan gespecificeerde informatie miste en vertelde hoe de software zich moet gedragen. Dit komt mede door de user stories die in klantjargon zijn beschreven, waardoor geen specifieke tijdsaanduidingen is opgegeven. Ik zal in het vervolg terugkoppeling moeten vinden met de opdrachtgever op het gebied van technische specificaties. Naar mijn mening is de methode ‘Extreme Programming’ in dit afstudeerproject niet op alle vlakke de ideale methode. In het vervolg zal ik wellicht een andere methode hanteren voor het ontwikkelen voor software, dit is afhankelijk van de situatie waar het project zich in verkeert.

Tijdens het afstudeerproject heb ik moeite gehad met het in kaart brengen van het bedrijf en de processen. Dit kwam volgens mij door de omvang van het project en de verschillende onderwerpen waarin ik mij heb moeten verdiepen om dit project te kunnen realiseren. Geleidelijk heb ik een steeds beter beeld kunnen vormen van de uitgangssituatie en de gewenste situatie.

Voordat ik was begonnen met het ontwikkelen van de software heb ik eerst een architectural spike uitgevoerd. Tijdens het uitvoeren van deze spike heb ik vertraging opgelopen. Dit veroorzaakte dat ik minder tijd had om aan de iteraties te werken. Vanwege de complexiteit van het probleem was ik genoodzaakt om me te verdiepen in een alternatieve oplossing vinden met behulp van SQL. Dit had een vertraging van 2 weken als gevolg. Hoogstwaarschijnlijk heeft dit veroorzaakt dat ik niet alle op te leveren producten heb kunnen realiseren. Ik heb hiervan geleerd dat het heel goed mogelijk is om vast te raken en er te lang mee bezig te zijn geweest. In dit geval is het goed gegaan en heb ik dit kunnen implementeren om aan de klanteis met de hoogste prioriteit te voldoen.

Nadat ik eindelijk een workaround had gevonden, ben ik gelijk aan de slag gegaan met de implementatie. De implementatiefase heeft ook veel tijd vereist, dit vanwege de verschillende technologieën die ik heb gebruikt om dit allemaal te kunnen realiseren.
In deze fase van het project zijn er veel deuren voor me open gegaan op het gebied van implementatievormen. Ik zal hier in de toekomst met behulp van de kennis die ik heb opgedaan, veel baat aan hebben.

In het algemeen ben ik zeer tevreden over de kennis die ik tijdens dit afstudeerproject heb opgedaan en is de samenwerking met de werknemers van Lijbrandt Telecom en de opdrachtgever naar mijn inzien goed verlopen.

Appendix

Software t.b.v. kostenregistratie binnen VoIP

in opdracht van

Appendix I - Bronnenlijst

Voor het vergaren van informatie met betrekking tot de werkzaamheden van dit project heb ik de volgende URL's gebruikt:

http://www.ecolenet.nl/beeld/scenario.htm
http://www.extremeprogramming.org
http://www.dsdm.nl/nl/dsdm/timeboxing.asp
http://www.postgresql.org/docs/8.0/static/index.html
http://www.voip-info.org/wiki-Asterisk+cdr+csv
http://www.voip-info.org
http://www.tldp.org/HOWTO/VoIP-HOWTO.html
http://www.automated.it/guidetoasterisk.htm
http://www.asterisk.org/
http://www.tweakers.net/nieuws/35261
http://www.emerce.nl/nieuws.jsp?id=412628
http://www.ixiacom.com/library/technology_guides/tg_display.php?skey=h323
http://www.coin.nl/
http://www.template-toolkit.org/tpc4/index.html
http://www.exact.nl/
http://www.webopedia.com/TERM/P/POTS.html
http://www.sum-it.nl/cursus/dbdesign/hollands/logis010.php3
http://www.phphulp.nl/php/tutorials/3/150/259/

Tevens heb ik gebruik gemaakt van de volgende boeken:

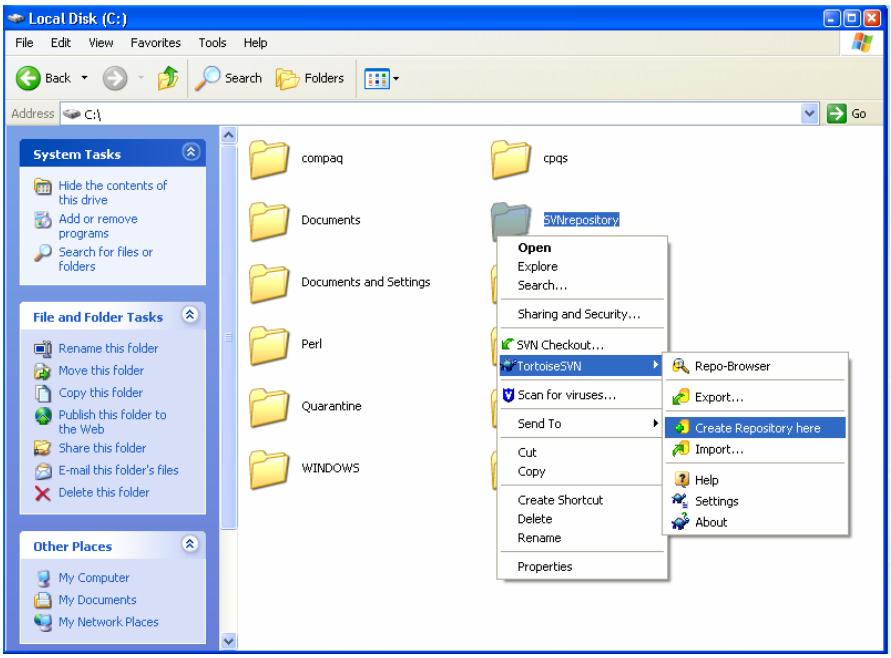
- Perl Cookbook, 2nd edition, Tom Christiansen & Nathan Torkington, O'REILLY, ISBN 0-596-00313-7
- De UML toolkit, Hans-Erik Eriksson & Magnus Penker, Academic Service, ISBN 90-395-1015-6
- Programming Perl, 3th edition, Larry Wall, Tom Christiansen & Jon Orwant, O'REILLY, ISBN 0-596-00027-8
- A Guide to THE SQL STANDARD, fourth edition, C.J. Date & Hugh Darwen, Addison Wesley, ISBN 0-201-96426-0
- PostgreSQL, Korry Douglas & Susan Douglas, Developer's Library, ISBN 0-7357-1257-3
- Teach Yourself Perl, Laura Lemay, SAMS, ISBN 0-672-31305-7
- Applying UML and patterns, second edition, Craig Larman, ISBN 0-13-092569
- Step by Step XML, Michael J. Young, Microsoft, ISBN 0-7356-1020-7
- Moduleboek DB-71, M. Heijne, Sector Informatica

Appendix II – Bijlagen

Om nadere toelichting te geven op het gebied van de broncode en dergelijke, heb ik een onderscheid gemaakt tussen het procesverslag en bijlagen. Zo wil ik voorkomen dat het verslag volledig is gevuld met irrelevante informatie op het gebied van procesvoortgang.
Op de volgende pagina's vindt u de bijlage waar ik naar verwijs in dit document.

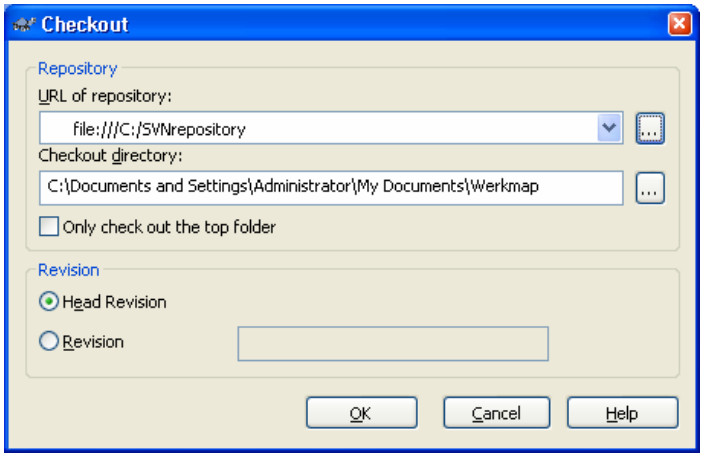
Versiebeheer – Subversion

TurtoiseSVN is een Subversion windows client en kan lokale repositories aanmaken. Een repository is de ruimte waar de versies worden opgeslagen en de meta-gegevens over deze versies. Om deze lokale repository aan te maken is het een kwestie van een nieuwe lege map aanmaken op de lokale schijven van het werkstation. Het is niet mogelijk om repositories aan te maken op andere systemen via het netwerk.
Allereerst heb ik een map gemaakt op mijn `C:\` partitie en heb het “SVNrepository” genoemd.
Als men de map met de rechtermuisknop klikt, krijgt men de kans om deze map toe te wijzen als lokale repository zoals ik heb gedaan op afbeelding 4.5.1 ‘Lokale repository aanmaken’.



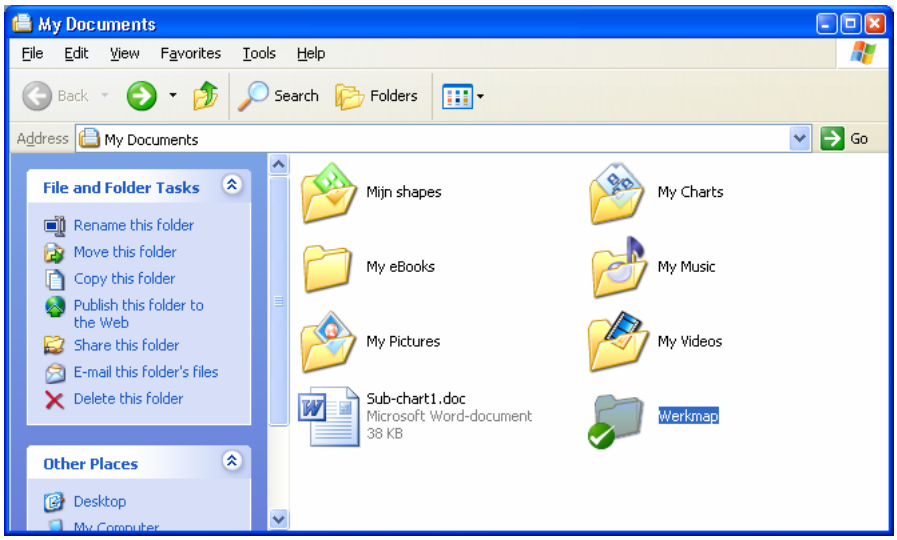
Afbeelding 4.5.1 ‘Lokale repository aanmaken’

Vervolgens geef ik aan in welke map de bestanden staan waarop ik versiebeheer wil toepassen. Men is genoodzaakt om de repository op te geven, in dit geval de map die ik in de voorbeeld heb gebruikt. Op afbeelding 4.5.2 ‘Checkout-map’ ziet u de scherm waarop men de repository moet opgeven.



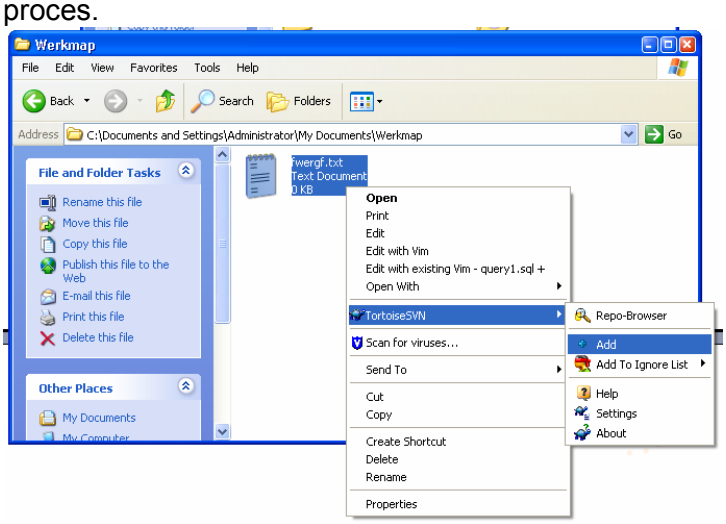
Afbeelding 4.5.2 ‘Checkout-map’

De werkmap waarop ik versiebeheer wil toepassen is nu van icoon veranderd en heeft een groene ‘check’ teken op de map. De map met daarin alle bestanden is nu gereed om versies voor te beheren. Op afbeelding 4.5.3 ‘Gereed voor gebruik’ ziet u de icoon.



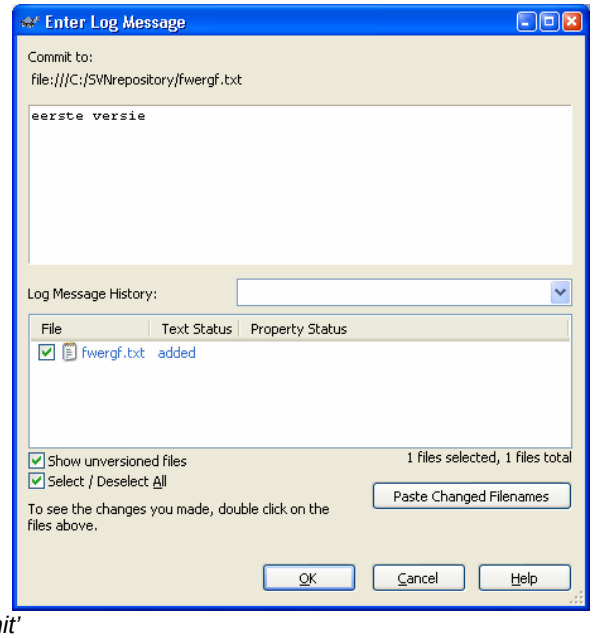
Afbeelding 4.5.3 ‘Gereed voor gebruik’

Ik maak nu een nieuwe bestand aan en door deze toe te voegen met TurtoiseSVN client. Op de volgende afbeelding 4.5.4 ‘Toevoegen’ ziet een momentopname van het toevoegen proces.



Afbeelding 4.5.4 ‘Toevoegen’

Er wordt nu versiebeheer toegepast op dit bestand en men kan doormiddel van het bestand te ‘commiten’ nadat er wijzigingen zijn aangebracht, de versie verhogen. Op afbeelding 4.5.5 ‘Commit’ ziet u een momentopname van de scherm indien er gecommit wordt.



Afbeelding 4.5.5 ‘Commit’

Naar mijn mening heeft de TurtoiseSVN interface een gebruiksvriendelijke omgeving en is het gemakkelijker om versies te beheren dan WinCVS(Windows client voor CVS). Dit omdat TurtoiseSVN client geïntegreerd is met Windows op het gebied van opdrachten uitvoeren. Dit gebeurt allemaal met behulp van de rechtermuisknop.

Priority prijslijst

Internationaal

Steden bestemmingen	Land / Netcode	Per minuut
België, Antwerpen	323	€ 0,044
België, Brussel	322	€ 0,044
Duitsland, Berlijn	4930	€ 0,038
Duitsland, Düsseldorf	49211	€ 0,038
Duitsland, Frankfurt am Main	4969	€ 0,038
Duitsland, München	4989	€ 0,038
Duitsland, Stutgart	49711	€ 0,038
Frankrijk, Parijs	331	€ 0,043
Italië, Milaan	392	€ 0,076
Italië, Rome (incl. Vaticaanstad)	396	€ 0,076
Italië, Turijn	3911	€ 0,076
Spanje, Barcelona	3493	€ 0,078
Spanje, Madrid	3491	€ 0,078
Verenigd Koninkrijk, London	4420	€ 0,032
Verenigde Staten, Boston	1617	€ 0,037
Verenigde Staten, Los Angeles	1213, 1310	€ 0,037
Verenigde Staten, New York	1212	€ 0,037
Verenigde Staten, San Francisco	1415	€ 0,037
Verenigde Staten, Washington D.C.	1202	€ 0,037

Prijzen in Euro's en exclusief 19% BTW December 2004 v5.0

Internationale bestemming	Land / Netcode	Vast per minuut	Mobiel per minuut
Afghanistan	93	€ 1.39	
Albanië	355	€ 0,272	€ 0,425
Algerije	213	€ 0,345	€ 0,515
Am. Maagdeneil.	1	€ 0,145	
Am. Samoa	684	€ 0,66	
Andorra	376	€ 0,17	€ 0,385
Internationale bestemming	**Land / Netcode**	**Vast per minuut**	**Mobiel per minuut**
Angola	244	€ 0,8760	
Anguilla	1264	€ 0,962	
Antigua & Barbuda	1268	€ 0,8760	

Argentinië	54		€ 0,408
Armenië	374		€ 0,585
Aruba	297		€ 0,331
Ascension	247		€ 1,1250
Austr. Antarctica	672		€ 1,18
Australië	61	€ 0,10	€ 0,38
Azerbeidzjan	994		€ 0,436
Bahama eilanden	1242	€ 0,9650	€ 1,07
Bahrein	973		€ 0,703
Bangladesh	880		€ 1,071
Barbados	1246		€ 0,785
België	32	€ 0,048	€ 0,241
Belize	501		€ 0,74
Benin	229		€ 0,74
Bermuda	1441		€ 0,313
Bhutan	975		€ 0,535
Bolivia	591		€ 0,735
Bosnië Herzegowina	387		€ 0,3150
Botswana	267		€ 0,427
Brazilië	55	€ 0,4150	€ 0,665
Britse Maagdeneil.	1284		€ 0,908
Brunei	673		€ 0,6850
Bulgarije	359	€ 0,3650	€ 0,495
Burkina Faso	226		€ 0,781

Broncode

Tarieven_instellen.pl

```
#!/usr/bin/perl -w
# Author: Yong Bing Hu
# Studentnumber: 20024721
# Filename: tarieven_instellen.pl
# Date: 2005-04-21

#Geef de modulen die ik ga gebruiken
use strict;
use CGI qw(:standard);
use CGI::Carp qw/fatalsToBrowser/;
use Template;
use DBI;

#Verbind met de database
my $db = DBI-
>connect('dbi:Pg:dbname=asterisk;host=localhost;', 'root',
undef,
   { RaiseError => 1, AutoCommit => 1});

#defineer een lege hash om gegevens mee te geven aan de
templates
my %data = ();

# Loop eerst de parameter lijst door om te kijken of er een
verwijder knop is
# ingedrukt
my $verwijder_id;
for my $param (param()) {
  # Kijk of de parameter begint met verwijder_
  if(index($param, "verwijder_") == 0) {
    # Haal het id nummer op wat achter verwijder_ staat
    $verwijder_id = substr($param, length("verwijder_"));
  }
}

#Wanneer knop is ingedrukt doe:
if(defined param("toevoegen")) {
  my $prefix = param("prefix");
  my $begintijd = param("begintijd");
  my $eindtijd = param("eindtijd");
  my $tarief = param("tarief");
  my $starttarief = param("starttarief");

  #Undefine de variabelen indien er geen waarden zijn ingevuld
  undef $begintijd if $begintijd eq "";
  undef $eindtijd if $eindtijd eq "";

  #Voer de INSERT QUERY uit
  $db->do("
    INSERT INTO tarieven (prefix, begintijd, eindtijd, tarief,
starttarief)
    VALUES (?, ?, ?, ?, ?);", {},
    $prefix, $begintijd, $eindtijd, $tarief, $starttarief);

  #Leidt men terug naar de webpagina
  print redirect("tarieven_instellen.pl");
}
```

```
#Wanneer er op de knop Verwijder is gedrukt
elsif ($verwijder_id) {
  $db->do("DELETE FROM tarieven WHERE tarieven_id = ?", {},
$verwijder_id);
  print redirect("tarieven_instellen.pl");
}

else {
  #Definieer array $data{'tarieven'} met de resultset
  $data{'tarieven'} = $db->selectall_arrayref("
  SELECT tarieven_id, prefix, begintijd, eindtijd, tarief,
starttarief
  FROM tarieven
  ORDER BY prefix DESC, begintijd IS NULL, (begintijd -
eindtijd) DESC, begintijd", {Slice => {}});

  my $template = Template->new({
    INCLUDE_PATH => 'templates',
  });

  print header("text/html");
  #Gebruik de template en geef de waarden mee die in Hash data
is meegegeven
  $template->process("tarieven_instellen.html.templ", \%data)
    || die $template->error();
}
```

Kosten opvragen.pl

```
#!/usr/bin/perl -w
# Author: Yong Bing Hu
# Studentnumber: 20024721
# Filename: kosten_opvragen.pl
# Date: 2005-05-16

#Geef de modulen die ik ga gebruiken
use strict;
use CGI qw(:standard);
use CGI::Carp qw/fatalsToBrowser/;
use Template;
use DBI;

#Verbind met de database
my $db = DBI->connect('dbi:Pg:dbname=asterisk;host=localhost;',
'root', undef,
   { RaiseError => 1, AutoCommit => 1});

#defineer een lege hash om gegevens mee te geven aan de
templates
my %data = ();

my $maand = param("maand");
my $jaar = param("jaar");

$data{'query_opgegeven'} = 1 if defined $maand && defined $jaar;

#Wanneer gegevens zijn ingevoerd
if($data{'query_opgegeven'}){
  #Maak een array aan waarin de resultset in komt
  $data{'cdr'} = $db->selectall_arrayref("
    SELECT src, dst, billsec, calldate, billsec,
      tarieven.starttarief + tarieven.tarief * billsec AS kosten
    FROM
      cdr LEFT JOIN tarieven ON (tarieven.tarieven_id =
        ( SELECT tarieven.tarieven_id
```

```
          FROM tarieven
          WHERE POSITION(tarieven.prefix IN cdr.dst) = 1
            AND (begintijd IS NULL
              OR (CAST(cdr.calldate AS time) > tarieven.begintijd
                AND CAST(cdr.calldate AS time) < tarieven.eindtijd))
          ORDER BY prefix DESC, begintijd IS NULL, (begintijd -
eindtijd) DESC, begintijd
          LIMIT 1))
    WHERE EXTRACT(YEAR FROM calldate) = ?
      AND EXTRACT(MONTH FROM calldate) = ?", {Slice => {}},
    $jaar, $maand);
}

print header("text/html");

my $template = Template->new({
  INCLUDE_PATH => 'templates',
  PRE_PROCESS  => 'config.templ',
});

#Gebruik de template en geef de waarden mee die in Hash data
is meegegeven
$template->process("kosten_opvragen.html.templ", \%data)
  || die $template->error();
```

Template “tarieven_instellen.html.templ”

```
[% title = "Tarieven instellen" %]

[% INCLUDE "header.html.templ" %]

<form method="post" action="tarieven_instellen.pl">
  <table>
    <thead>
      <th>Prefix</th>
      <th>Begintijd</th>
      <th>Eindtijd</td>
      <th>Tarief</th>
      <th>Starttarief</th>
    </thead>
    <tbody>
        <td><input type="text" name="begintijd"/></td>
        <td><input type="text" name="eindtijd"></td>
        <td><input type="text" name="tarief"/></td>
        <td><input type="text" name="starttarief"/></td>
        <td> </td>
        <td><input type="submit" name="toevoegen"
value="Toevoegen"/></td>
      </tr>
      [% FOREACH tarief = tarieven %]
      <tr>
        <td>[% tarief.prefix %]</td>
        <td>[% tarief.begintijd %]</td>
        <td>[% tarief.eindtijd %]</td>
        <td>[% tarief.tarief %]</td>
        <td>[% tarief.starttarief %]</td>
        <td><input type="submit" name="verwijder_[%
tarief.tarieven_id %]" value="Verwijder"/></td>
      </tr>
      [% END %]
    </tbody>
  </table>
</form>

[% INCLUDE "footer.html.templ" %]
```

Template “tarieven_instellen.html.templ”

```
[% title = "Kosten opvragen" %]

[% INCLUDE "header.html.templ" %]

<form method="get" action="kosten_opvragen.pl">
  <table>
    <thead>
      <th>Maand</th>
      <th>Jaar</th>
    </thead>
    <tbody>
      <tr>
        <td><select name="maand">
            <option>Selecteer een maand...</option>
            <option value="1">Januari</option>
            <option value="2">Februari</option>
            <option value="3">Maart</option>
            <option value="4">April</option>
            <option value="5">Mei</option>
            <option value="6">Juni</option>
            <option value="7">Juli</option>
            <option value="8">Augustus</option>
            <option value="9">September</option>
            <option value="10">Oktober</option>
            <option value="11">November</option>
            <option value="12">December</option>
            </select>
        <td><input type="text" name="jaar"/></td>
        <td><input type="submit" name="opvragen"
value="Opvragen"/></td>
      </tr>
    </tbody>
  </table>
</form>
[% IF query_opgegeven %]
<table>
   <thead>
     <th>Bron</th>
     <th>Bestemming</th>
     <th>Datum</th>
     <th>Duratie gesprek</th>
     <th>Kosten</th>
   </thead>
   <tbody>
     [% FOREACH cdr_record = cdr %]
     <tr>
       <td>[% cdr_record.src %]</td>
       <td>[% cdr_record.dst %]</td>
       <td>[% cdr_record.calldate %]</td>
       <td><center>[% cdr_record.billsec %]</center></td>
       <td>[% cdr_record.kosten %]</td>
     </tr>
     [% END %]
    </tr>
  </tbody>
</table>
[% END %]
[% INCLUDE "footer.html.templ" %]
```

Mind XML output

```
<?xml version="1.0" ?>
- <eExact xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="eExact-Schema.xsd">
    - <Invoices>
        - <Invoice type="V" code="100">
            <Description> </Description>
              - <OrderedBy>
                  <Debtor code="15959" />
                </OrderedBy>
        - <InvoiceLine>
          <Item code="BI" />
            <Quantity>1</Quantity>
              - <Price>
                  <Currency code="EUR" />
                  <Value>56.18</Value>
                </Price>
                <Text>Gesprekskosten Binnengebied</Text>
          </InvoiceLine>
        - <InvoiceLine>
          <Item code="BU" />
            <Quantity>1</Quantity>
              - <Price>
                  <Currency code="EUR" />
                  <Value>42.48</Value>
                </Price>
                <Text>Gesprekskosten Buitengebied</Text>
          </InvoiceLine>
        - <InvoiceLine>
          <Item code="INT" />
            <Quantity>1</Quantity>
          -     <Price>
                  <Currency code="EUR" />
                  <Value>2.51</Value>
                </Price>
                <Text>Gesprekskosten Internationaal</Text>
          </InvoiceLine>
-         <InvoiceLine>
            <Item code="GSM" />
            <Quantity>1</Quantity>
          - <Price>
              <Currency code="EUR" />
              <Value>852.52</Value>
            </Price>
            <Text>Gesprekskosten Mobiel</Text>
          </InvoiceLine>
    </Invoice>
    </Invoices>
    </eExact>
- <!-- Totaalbedrag: 959.35 -->
```

MIND CSV output

Werknemer ID	Best. Type	Kosten (€)
14233	3	0,05
14233	2	0,15
14233	5	0,05
14233	5	0,05
14233	5	0,05
14233	5	0,07
14233	5	0,05
14233	5	0,06
14233	8	0,67
14233	8	0,67
14233	2	0,05
14233	2	0,05
14233	2	0,22
14233	2	0,07
14233	2	0,18
14233	5	0,42
14233	5	0,16
14233	2	0,05
14233	2	0,04
14233	3	0,05
14233	2	0,04
14233	2	0,05
14233	5	0,42
14238	3	0,28
14233	2	0,06
14233	3	0,07
14233	2	0,12
14238	2	0,13
14233	2	0,04
14233	2	0,04
14233	5	0,59
14238	2	0,08
14233	2	0,04
14233	5	0,22
14233	5	0,05
14233	5	0,05
14233	5	0,05
14233	2	0,06
14233	5	0,05
14233	5	0,29
14233	2	0,04
14233	2	0,04
14238	2	0,21
14233	2	0,04
14233	3	0,05
Totaal voor alle 192 Gesprekken		28,31