Afstudeerverslag

In opdracht van Metrix B.V.

De Ontwikkeling van de modules 'Facturatie' en 'Beheer'

Auteur:	Ruben Maas
Studentnummer:	99008637
Datum:	09-01-2004
In opdracht van:	Metrix B.V.

Colofon

Naam	Ruben Maas
Studentnummer	99008637
Afstudeerperiode:	1 september 2003 t/m 9 januari 2004
Opleiding:	Informatica & informatiekunde, Haagse Hogeschool
Afstudeerrichting:	Vormgeving en Ontwerp van Interactie
Afstudeerblok:	2003-2.1
Project:	Ontwikkeling modules ‘Facturatie’ en ‘Beheer’
Stagebedrijf:	Metrix b.v.
bedrijfsmentor:	dhr. D. Stoikof
Examinatoren:	mevr. A. D. Van Duuren dhr. V.H.F. Hermans

Metrix B.V.
Laan van nieuw oost indië 16
2593 BT
Den Haag

Telefoon: +31 (0)70 383 41 14
Telefax: +31 (0)70 383 58 30

Haagse Hogeschool
Sector Informatica

Bezoekadres: Johanna Westerdijkplein 75
2521 EN Den Haag

postadres:
Postbus 19070
2500 CB Den Haag

Telefoon: +31 (0)70 445 84 00
Telefax: +31 (0)70 445 84 05

Referaat

Maas, R. Stageverslag, ontwikkeling van de modules ‘Facturatie’ en ‘Beheer’. Metrix B.V. te Den Haag, januari 2004.

Dit stageverslag is geschreven in het kader van de het afstuderen aan de opleiding Informatica & Informatiekunde, afstudeervariant Vormgeving en Ontwerp van Interactie, aan de Haagse Hogeschool te Den Haag.

De auteur geeft in dit verslag zijn ervaringen en problemen weer die zijn opgedaan en de daarbij gevonden oplossingen en gemaakte keuzes in het kader van deze stage in de periode 1 september 2003 tot 9 januari 2004.

Dit verslag dient om de stagedocent en andere lezers van dit document inzicht te geven in de omvang van de opdracht en de diepgang waarmee de verschillende aspecten van de opdracht zijn uitgewerkt. Daarnaast geeft dit verslag een beschrijving over de wijze waarop de producten tot stand zijn gekomen en de uiteindelijke opgeleverde producten voor het bedrijf.

De afstudeeropdracht heeft betrekking op het ontwikkelen van de modules ‘Facturatie’ en ‘Beheer’ met behulp van Microsoft Visual Basic 6.0, ASP, en DHTML.

Descriptoren:

- Afstudeerverslag
- Metrix B.V.
- Haagse Hogeschool
- Vormgeving en ontwerp van interactie
- Microsoft Visual Basic
- Active server pages
- ASP
- DHTML
- JavaScript
- Dynamic Link Library

Voorwoord

In de periode van 1 september 2003 tot en met 9 januari 2004 heb ik mijn afstudeerstage vervuld bij het bedrijf Metrix B.V.

Hierbij wil ik graag van de gelegenheid gebruik maken om iedereen te bedanken die mij hebben geholpen tijdens deze opdracht.

Allerteerst gaat mijn dank uit naar Metrix B.V. en in het speciaal Dimitri Stoikof voor de begeleiding en hulp bij lastige vraagstukken. Tevens gaat mijn dank uit naar Ewout Pronk voor het delen van zijn kennis en zijn collegialiteit.

Verder gaat mijn dank ook uit naar mijn examinatoren mevr. A. D. Van Duuren en dhr. V.H.F. Hermans voor de ondersteuning en sturing tijdens dit project.

Den Haag, januari 2004

Inhoudsopgave

1 Inleiding 7
2 Opstartfase 9
2.1 Het afstudeerbedrijf 9
2.1.1 Geschiedenis 9
2.1.2 Huidige werkzaamheden 9
2.1.3 Plaats opdracht binnen Metrix 10
2.1.4 Plaats afstudeerder binnen Metrix 10
2.2 Opdrachtomschrijving 10
2.2.1 Inleiding 10
2.2.2 Probleemstelling 11
2.2.3 Doelstelling 11
2.3 De ontwikkelmethode 11
2.3.1 IAD als ontwikkelmethode 12
2.4 Betrokken partijen 13
2.5 Op te leveren producten 14
2.6 Welke technieken worden gebruikt 14
2.7 De planning 14
3 Definitiestudie 17
3.1 Planning 17
3.2 Inwinnen van informatie 17
3.3 Huidige situatie eindgebruikers 18
3.4 Huidige situatie Key-Task 20
3.5 Systeemeisen 24
3.6 Gewenste situatie 25
3.7 Technische-architectuur 27
3.8 Pilotplan 28
3.8.1 Verdeling Pilots 29
3.8.2 pilot Facturatie 29
3.8.3 pilot Beheer 29
3.8.4 Prioritering Pilots 29
3.8.5 Pilottesten 29
3.8.6 Planning 31
4 Pilot Facturatie 34
4.1 Pilotontwikkelplan 34
4.1.1 Plan van aanpak 34
4.1.2 Workshop 35
4.1.3 Functionele Structuur 35
4.1.4 Ontwerp presentatielaag 36
4.1.5 Ontwerp businesslaag 39
4.1.6 Ontwerp datalaag 41
4.1.7 Ontwerp Database 43
4.1.8 Aanpassingen externe componenten 43
4.1.9 Bouweenheden 43
4.2 Realisatie 44
4.3 Testen 46
4.3.1 Aanpak 46
4.3.2 Resultaten 47
4.3.3 Aanpassingen 47
5 Pilot Beheer 48
5.1 Pilotontwikkelplan 48
5.1.1 Plan van aanpak 48
5.1.2 Functionele Structuur 49
5.1.3 Ontwerp presentatielaag 49
5.1.4 Ontwerp businesslaag 51
5.1.5 Ontwerp datalaag 52
5.1.6 Bouweenheden 53

5.2 Realisatie 54
5.3 Testen 55
5.3.1 Aanpak 55
5.3.2 Resultaten 55
6 Procesevaluatie 57
6.1 Opstartfase en Plan van aanpak 57
6.2 Definitiestudie 58
6.3 Pilotontwikkeling 58
7 Productevaluatie 60
7.1 Opgeleverde documenten 60
7.1.1 Plan van aanpak 60
7.1.2 Definitiestudie 60
7.1.3 Pilotontwikkelplannen 60
7.1.4 Testrapporten 60
7.1.5 Handleidingen 61
7.2 Module Facturatie 61
7.3 Module Beheer 62
8 Evaluatie doelstelling en probleemstelling 64
9 Bronnenlijst 65
10 Woordenlijst 66

1 Inleiding

Gedurende de periode van 1 september 2003 tot 9 januari 2004 heb ik mij beziggehouden met het afstuderen aan de opleiding Informatica/informatiekunde, opleidingsvariant VIA, aan de Haagse Hogeschool. In dit afstudeerverslag licht ik de keuzes nader toe die ik gedurende mijn afstuderen heb gemaakt en beschrijf ik het proces wat ik in het kader van mijn opdracht heb gevolgd. De opdracht luidt 'Het ontwikkelen van de modules Facturatie en Beheer' ten behoeve van het Key-Task systeem en is in opdracht van Metrix B.V uitgevoerd.

Metrix B.V. is een software bedrijf wat zich bezig houdt met het ontwikkelen van servicemanagement systemen. Deze software dient ter ondersteuning van service verlenende activiteiten zoals: probleemafhandeling, helpdesk of het delen van kennis.

Metrix heeft hiervoor twee verschillende systemen ontwikkeld. Een systeem voor grote organisaties en een systeem, genaamd Key-Task, gericht voor organisaties binnen het midden -en kleinbedrijf. Aangezien Key-Task nog steeds in ontwikkeling is en het systeem betrekking heeft op mijn opdracht zal deze term veelvuldig ter spraken komen binnen dit verslag.

Waarom heb ik voor deze opdracht gekozen?
In het kader van mijn vorige twee projecten, heb ik mij bezig gehouden met de ontwikkeling van Key-Task. Gedurende 2½ jaar ben ik hier actief bij betrokken geweest. Nu de daadwerkelijke implementatie van dit systeem nabij is wil ik hier uiteraard mijn medewerking aan verlenen. Naar aanleiding van mijn ervaringen met Key-Task kan ik stellen dat deze opdracht voor mij gezien over alle facetten beschikt en waarbij ik vaardigheden kan beheersen die ik voor mijn toekomst belangrijk vind.

Dit verslag bestaat uit een viertal delen:

DEEL I	Opstartfase
DEEL II	Definitiestudie
DEEL III	Pilotontwikkeling
DEEL IV	Evaluatie

Deel I beschrijft de opstartfase van dit project. Dit deel bestaat uit hoofdstuk 2, opstartfase. Hierin wordt een eerste aanzet gegeven over het kader waarin de opdracht zich bevindt en staat het plan van aanpak beschreven.

Deel II bestaat uit het hoofdstuk 3 'Definitiestudie'. Hierin ga ik in op keuzes die ik maak in de fase waarin de doelen en beperkingen van het systeem in kaart worden gebracht.

Deel III bestaat uit hoofdstuk 4 en 5 waarin de fase pilotontwikkeling beschreven wordt. Hoofdstuk 4 behandelt de pilot 'Facturatie' en hoofdstuk 5 'Beheer'.

Deel IV tenslotte bestaat uit de evaluatie van zowel het proces als het product.

DEEL I

Opstartfase

2 Opstartfase

In dit hoofdstuk wordt beschreven hoe de opstartfase van dit project is verlopen. Allereerst geef ik een korte beschrijving over het bedrijf waar ik mijn afstudeeropdracht heb voltooid en wat mijn plek binnen de organisatie was. Vervolgens beschrijf ik de opdracht en hoe het opstellen van het plan van aanpak is verlopen.

2.1 Het afstudeerbedrijf

Metrix is een Amerikaans bedrijf dat zich bezig houdt met het ontwikkelen van field service management software. Field service management software draait om het automatiseren van operationele processen, waarbij relatiebeheer, contractadministratie, logistiek en workflow een grote rol spelen. Metrix richt zich hierbij op grote organisaties, zoals *Ericsson, Xerox, Vivendi Water, Nokia en Canon*.

2.1.1 Geschiedenis

Metrix is opgericht in 1980 door de amerikanen Larry Laux, Rob Oldenburgh en Jim Elliot. Aanvankelijk was het opgericht als een automatiseringsbedrijf bestaande uit de bovengenoemde ervaren programmeurs. In de periode 1980 tot en met 1983 hebben ze echter een toepassing opgeleverd, die service activiteiten van een organisatie kon managen.

Deze applicatie werd geschreven in de programmeertaal Basic voor de HP250. In die tijd was er nog sprake van dat een programmeertaal die je gebruikt afhankelijk is van een bepaald type computer. In 1987 hebben ze een tweede versie op de markt gebracht die ook de service activiteiten van repair-centers kon managen. Deze versie werd geschreven met behulp van Cobol voor de HP3000. Mede door deze versie heeft het bedrijf tot aan begin jaren negentig kunnen uitgroeien tot een organisatie met ongeveer 100 werknemers.

In het begin van de jaren negentig is Metrix overgestapt op de derde en heel snel daarna de vierde generatie programmeertalen. Respectievelijk werden hiervoor Uniface van Compuware en Forte van Sun Microsystems gebruikt. Het resultaat is een robuuste applicatie genaamd Metrix 4^{e} welke database onafhankelijk is en tot de bovenkant van de markt gerekend kan worden op service management gebied. Tot op de dag van vandaag wordt Metrix 4^{e} verkocht aan klanten en worden uitbreidingen en aanpassing er op gedaan. Doordat Metrix ook buiten Amerika een grote klantenkring opbouwde hebben ze ertoe besloten een Europese vestiging te openen. Metrix Europe of te wel Metrix B.V. werd gevestigd in Den Haag. Metrix B.V. heeft als doel de Metrix software te verkopen en te ondersteunen in landen binnen Europa, Afrika en het Midden-Oosten, ook wel EMEA genoemd.

2.1.2 Huidige werkzaamheden

Metrix is één van de marktleiders in het maken en ondersteunen van field service management producten. Dit beslaat alle aspecten van fieldservice management voor bedrijven met operationele op service gerichte activiteiten. Metrix software vindt men terug als service management oplossing binnen de medische industrie, de kopieerapparaten industrie, telecommunicatie industrie, en reparatiecenters, waarbij voornamelijk technische diensten, helpdesk en logistiek geautomatiseerd worden. De Metrix software is de laatste twee jaar uitgebreid met krachtige planningssoftware en een scala van ‘remote’ oplossing waarbij zogenaamde handhelds en PDA’s gebruikt worden.

Huidige werkzaamheden van Metrix B.V:

- Het leveren van product ondersteuning bij bestaande klanten.
- Het werven van nieuwe klanten binnen Europa, Afrika en het Midden-Oosten.
- Het ontwikkelen van nieuwe applicaties voor het MKB.

Huidige werkzaamheden van Metrix USA:

- Het verder verbeteren van Metrix 4^{e}
- Het leveren van product ondersteuning bij bestaande klanten
- Het werven van nieuwe klanten in Noord en Zuid-Amerika.
- Het ontwikkelen van nieuwe (niet nader genoemde) applicaties.

2.1.3 Plaats opdracht binnen Metrix

De opdracht heeft de titulatuur “proof of concept”. Dit wil zeggen dat Metrix denkt dat er een markt is voor een klein Metrix product aan de onderkant van de markt. Door middel van deze “proof of concept” wil Metrix zijn strategie gaan bepalen voor de onderkant van de markt of besluiten daar niets in te doen. Hier ligt een mooie kans voor een afstudeerder om een eerste aanzet te geven om het product vorm te geven. De opdracht is binnen Metrix uniek omdat Metrix zich tot nu toe puur heeft gericht tot grote organisaties waarbij heel veel functionaliteit moet worden geboden. Het concept om een “beperkt” product aan te bieden voor de onderkant van de markt is met name op Europa gericht.

2.1.4 Plaats afstudeerder binnen Metrix

De afstudeerder is werkzaam op de afdeling software-development, deze afdeling is klein omdat de meeste ontwerpactiviteiten in de Verenigde Staten plaatsvinden. Binnen deze afdeling zijn vier vaste medewerkers werkzaam. Daarnaast is er nog een afstudeerder actief binnen deze afdeling, die zich bezig houdt met de ontwikkeling van een Learn-Management systeem.

2.2 Opdrachtomschrijving

2.2.1 Inleiding

De opdracht heeft betrekking op een reeds bestaand systeem genaamd Key-Task. Alvorens ik u inzicht laat geven in zowel de probleemstelling als de doelstelling, zal ik een korte inleiding geven over het Key-Task systeem.

Doordat Metrix veelal in aanraking kwam met bedrijven, die zich qua vermogen en bezetting te klein achtten voor Metrix 4^{e}, heeft Metrix een CRM pakket laten ontwikkelen voor bedrijven uit het MKB. Hierbij valt Key-Task te vergelijken met Metrix alleen is Key-Task gericht op de behoeftes van kleine organisaties. Dit maakt Key-Task relatief goedkoop en zeer eenvoudig implementeerbaar.

Key-Task is een Service Management systeem. Service Management houdt in: Het beheer van service verlenende activiteiten van een organisatie. Een voorbeeld:

Stel je verkoopt wasmachines en je hebt daarbij 10 monteurs in dienst, die ervoor moeten zorgen dat eventuele reparaties verholpen worden. Een monteur wordt naar een locatie (klant) gestuurd om daar een wasmachine te repareren. Bij deze reparatie verbruikt de monteur onderdelen en besteed daar tijd aan en maakt ook reiskosten om op de locatie te komen. De monteur moet ingepland worden en de klant moet weten wanneer hij de monteur kan verwachten. Na afloop van de reparatie moeten alle gemaakte kosten op een correcte manier in rekening gebracht kunnen worden.

Een service management systeem ondersteunt het bijbehorende proces. Zo moet er bekend zijn waar het probleem zich afspeelt, welke reparateurs er beschikbaar zijn en welke kosten er zijn gemaakt.

Het Key-Task systeem houdt daarom onder andere de volgende gegevens bij:

1. De Klantgegevens
   Wie hebben er allemaal wasmachines gekocht.
2. Productgegevens
   Welke wasmachines bevinden zich bij welke klant.
3. Activiteiten
   Omvat gegevens over het gemelde probleem.
4. Taken
   Bedrijfsmatige opdrachten aan medewerkers, zoals het repareren van een wasmachine.
5. Onkosten
   Kosten die zijn gemaakt bij het uitvoeren van de activiteit (probleem).
6. Agenda
   De werktijden van de medewerker. Wanneer is een reparateur beschikbaar.

Key-Task is een algemeen softwarepakket dat flexibel kan worden ingezet in diverse soorten bedrijven met een verscheidenheid van diensten en producten.

2.2.2 Probleemstelling
Het Key-Task systeem is vooralsnog nog niet geheel bruikbaar voor organisaties binnen het MKB. Er zijn een aantal problemen die een succesvolle implementatie van het systeem in de weg staan.

Allereerst is het in de huidige situatie niet mogelijk het systeem te beheren. Het systeem beschikt over instellingen, die invloed uitoefenen op de werking van het systeem. In de huidige situatie kunnen die instellingen niet door de eindgebruiker worden aangepast. Tevens is er geen overzicht van de instellingen die het systeem gebruikt. Daarom kan het Key-Task systeem niet volledig worden ingericht op de betreffende organisatie.

Tevens blijkt, uit de mening van potentiële klanten, dat het systeem beperkt is in het gebruik, omdat het facturatieproces ontbreekt. Het gevolg hiervan is dat gebruikers in de meeste gevallen moeten terugvallen op het handmatig samenstellen van facturen. Buiten het feit dat dit probleem een dergelijke organisatie veel tijd en geld kost, is er geen overzicht welke kosten er al reeds zijn verrekend.

2.2.3 Doelstelling
De uitvoerder moet ervoor zorgen dat het Key-Task systeem volledig voorziet in de wensen van de eindgebruiker, zodat het systeem succesvol kan worden geïmplementeerd. Hiervoor beschrijft de opdracht de volgende toevoegingen aan het huidige Key-Task systeem:

1. Beheermodule; Binnen dit systeemdeel dient de gebruiker de mogelijkheid te hebben om de bestaande systeeminstellingen op te vragen en te wijzigen. Zodoende heeft de gebruiker de mogelijkheid het systeem naar haar organisatie in te richten.

2. Facturatiemodule; Binnen dit systeemdeel moet de gebruiker de mogelijkheid hebben de gemaakte kosten voor een activiteit aan de klant te verrekenen. Dit facturatieproces dient geheel geautomatiseerd plaats te vinden.

Bij de ontwikkeling van de bovenstaande modules dienen de integriteit, interface, performance en operationele-eisen, die aan Key-Task zijn gekoppeld, te worden gehandhaafd.

Tenslotte dienen de handleidingen van Key-Task te worden uitgebreid met de handleidingen van de te ontwikkelen modules.

2.3 De ontwikkelmethode
Om dit project in goede banen te leiden wordt bepaald welke ontwikkelmethode voor de uitvoer van dit project gebruikt zal worden. De keuze voor een ontwikkelmethode hangt af van het type project en de persoonlijke voorkeuren en ervaringen van de uitvoerder. Aangezien ik veel ervaring heb met de ontwikkelmethode IAD ligt deze ontwikkelmethode in eerste instantie voor de hand.
Echter heb ik gedurende de laatste twee projecten, die ik in opdracht van de opleiding VIA heb uitgevoerd, mij ook verdiept in andere ontwikkelmethodes. Dit werd veroorzaakt doordat IAD naar mijn mening niet optimaal verloopt voor sommigen type projecten. U kunt hierbij denken aan projecten met een korte doorlooptijd of waarbij geen gebruikersinterface van toepassing is. Inzicht in andere ontwikkelmethodes zal mij in staat kunnen stellen te bepalen welke ontwikkelmethode het beste past bij een bepaald type project.

Omdat IAD de enige ontwikkelmethode is die ik beheers, heb ik, voorafgaande aan dit project, een klein onderzoek verricht met als doel te kunnen bepalen welke ontwikkelmethode het best toegepast kan worden voor het uitvoeren van dit project. Bij dit onderzoek heb ik gebruik gemaakt van informatiebronnen zoals internet, bibliotheken en ervaringen van medestudenten. Uiteindelijk bleven er voor mij twee ontwikkelmethodes over, namelijk IAD(*Iterative Application Development*) en RAD(*Rappid Application Development*). Voordat ik een keuze maak heb ik allereerst de voor –en nadelen van beide methodes beschreven.

IAD werkt als	
1.	Er onduidelijke of snel wijzigende systeemeisen zijn.
2.	Een aantal bijna gelijksoortige applicaties worden ontwikkeld, waarbij hergebruik met name tot een stijging van de productiviteit zou kunnen leiden.
3.	Applicaties een grafische gebruikersinterfaces (GUI's) hebben, waarbij de specificaties alleen goed kunnen worden gevalideerd via werkende prototypen.
4.	Een organisatie veel waarde hecht aan een hoge mate van gebruikersparticipatie in het ontwikkelproces.

IAD werkt ***niet*** als	
1.	Het korte-termijn resultaat voorop staat, want dit kan leiden tot mindere kwaliteit van de architectuur en de documentatie.
2.	De klant niet kan leven met het veelvuldig nemen van (kleinere) beslissingen.
3.	Het niet mogelijk is om constant van prioriteit te wisselen.

RAD werkt als	
1.	De applicatie standalone is.
2.	Er al gebruik kan worden gemaakt van bestaande class libraries (API's).
3.	De performance niet kritisch is.
4.	Het systeem in verschillende modules kan worden gemaakt.
5.	De technologie meer dan een jaar oud is.

RAD werkt ***niet*** als	
1.	De applicatie samen moet kunnen werken met bestaande programma's.
2.	Er niet of nauwelijks gebruik kan worden gemaakt van plugin componenten.
3.	RAD wordt vervangen door QADAD (Quick And Dirty Application Development).
4.	RAD wordt gebruikt voor het maken van operating systems of games.

Mijn bevindingen wezen uit dat zowel RAD als IAD voor mijn project goed te hanteren zijn. Een keuze tussen deze twee varianten was voor dus erg lastig. In eerste instantie zal ik me graag willen verdiepen in een andere ontwikkelmethode, daar staat echter tegenover dat ik een risico neem voor wat betreft de te hanteren methodiek. Doordat ik geen enkele ervaring heb met RAD bestaat het risico dat ik veel tijd kwijt ben met het mezelf eigen maken van de ontwikkelmethode. Doordat ik in een vrij korte periode veel werk moet leveren wil ik dit risico niet lopen. Daarom heb ik gekozen voor een tussen weg, zijnde IAD toe te passen als RAD variant.

De ontwikkelmethode kent diverse strategieën. Een ontwikkelstrategie zegt veel over de wijze waarop er wordt ontwikkeld. IAD beschrijft de volgende strategieën:

- Evolutionair ontwikkelen
- Incrementeel ontwikkelen
- Incrementeel opleveren
- Big bang invoeren

Incrementeel ontwikkelen is een strategie die erg lijkt op RAD. Het voordeel van deze strategie is dat deze zeer geschikt is voor de ontwikkeling van modules. Het hanteren van IAD met daarbij als ontwikkelstrategie incrementeel ontwikkel is voor mij dus een prima oplossing. Zo wordt voorkomen dat ik me moet gaan verdiepen in een ontwikkelmethode waar ik geen ervaring mee heb en kan ik mijn ontwikkelproces zodanig uitvoeren dat ik daar als ontwikkelaar baat bij heb. In *paragraaf 2.3.1* zal verder ingegaan worden in de ontwikkelmethode IAD en hoe ik deze ga toepassen.

2.3.1 IAD als ontwikkelmethode

Nu ik heb besloten welke ontwikkelmethode ik ga toepassen en welke strategie ik daarbij ga hanteren zal ik aan u uitleggen hoe dit binnen mijn project wordt toegepast.

De ontwikkelmethode IAD beschrijft een drietal fases, namelijk *definitiestudie, pilotontwikkeling en invoering*. Afhankelijk van de gekozen strategie kunnen deze fases worden geïtereerd. Een iteratie wil zeggen dat het volledige proces van een bepaalde fase nogmaals wordt uitgevoerd. Binnen de fase definitiestudie worden de doelen en beperkingen

van het systeem beschreven. Zodoende weet de ontwikkelaar aan welke eisen het systeem moet voldoen. De fase pilotontwikkeling wordt gebruikt om de systeemdelen te ontwerpen en te bouwen. Doormiddel van een gebruikerstest kan bepaald worden of er een nieuwe iteratie plaatsvindt op deze fase. De fase invoering moet er tot slot voor zorgdragen dat de software succesvol kan worden geïmplementeerd.

In *figuur 2.1* staat weergegeven hoe de ontwikkelmethode IAD wordt toegepast op mijn project.

Figuur 2.1 Incrementeel Ontwikkelen

De bovenstaande figuur laat de drie fases zien en de volgorde waarop ze worden uitgevoerd. Opvallend hierbij is dat enkel op de fase pilotontwikkeling een iteratie kan plaatsvinden. Dit wordt bepaald door de ontwikkelstrategie die ik heb gekozen. Incrementeel ontwikkelen wil zeggen dat er een vrij korte aanloop fase is tot aan de pilotontwikkeling, waarop de fase pilotontwikkeling iteratief wordt gevolgd. Dat wil zeggen dat de focus van dit project komt te liggen op de fase pilotontwikkeling. Eén van de gevolgen van deze ontwikkelstrategie is dat er naar verhouding minder tijd wordt ingepland voor de definitiestudie.

2.4 Betrokken partijen

De betrokken partijen zijn de personen of instanties die invloed uitoefenen op dit project. De ontwikkelmethode beschrijft de volgende betrokken partijen:

1. Ontwikkelaar
2. Opdrachtgever
3. Gebruikers

Voor een groot gedeelte is reeds voor aanvang van dit project bekend welke personen of instanties de betreffende betrokken partijen zullen gaan vertegenwoordigen, namelijk:

1. Ontwikkelaar Ruben Maas
2. Opdrachtgever Dimitri Stoikof

Echter is tot op heden nog niet bekent wie de gebruikers gaan vertegenwoordigen. Omdat het Key-Task systeem een algemeen systeem betreft wat in meerdere organisaties kan worden ingezet, kun je niet per definitie zeggen wie de eindgebruikers zijn. Bij een voorgaand project werd dit opgelost door de opdrachtgever tevens als eindgebruiker naar voren te schuiven. Ik ben er echter van overtuigd dat dit geen goede oplossing is, omdat een opdrachtgever met een andere gedachten goed naar een applicatie kijkt dan een werkelijke eindgebruiker. Een opdrachtgever zal over het algemeen denken vanuit de applicatie, terwijl een werkelijke eindgebruiker vanuit het bedrijfsproces denkt. Het wegnemen van bepaalde problemen die invloed hebben op het bedrijfsproces is voor hen belangrijker dan het technische vernuft van een applicatie.

Het is dus zoeken naar een bedrijf die geïnteresseerd is in Key-Task en die de rol als eindgebruiker, binnen mijn project, op zich wil nemen.
Metrix beschikt over een netwerk van bedrijven die geïnteresseerd zijn in een dergelijk systeem. Hierdoor heb ik de mogelijkheid om met bedrijven in contact te komen die Key-Task willen gebruiken ter ondersteuning van hun bedrijfsproces en tevens willen deelnemen met dit project. Dit wil zeggen dat het bedrijf een actieve functie moet gaan vervullen door zich open te stellen voor ontmoetingen in het kader van workshops, interviews en testsessies. Uiteraard wordt van hen ook verlangd feedback te kunnen geven op datgene wat ontwikkeld wordt.

Zodoende ben ik in contact gekomen met een tweetal bedrijven die de rol van eindgebruiker op zich willen nemen. De bedrijven waren respectievelijk Darumath Services en Do-Company.

Ik moet dus een keuze maken tussen deze twee bedrijven welke er als eindgebruiker fungeert. Doordat de ontwikkelmethode die ik hanteer een hoge gebruikersparticipatie voorschrijft, is veelvuldig contact met de eindgebruiker noodzakelijk. Aangezien Do-Company gevestigd is in Rotterdam en Darumath in Den Haag heb ik, gezien de afstand, uiteindelijk ervoor gekozen om Darumath als eindgebruikers van dit project aan te nemen.

Nu de keuze voor de eindgebruikers op Darumath is gevallen zal binnen dit verslag veel verwezen worden naar dit bedrijf. De medewerkers van Darumath zijn de eindgebruikers waardoor Darumath invloed uitoefent op dit project. Het is daarbij wel van belang dat het Key-Task systeem in meerdere organisaties implementeerbaar blijft.

2.5 Op te leveren producten

Nu de strategie vastligt kan ik aan de hand van IAD bepalen welke producten er worden opgeleverd. Gedurende de looptijd van een project worden documenten/producten opgeleverd ter ondersteuning van de uiteindelijk realisatie. Dit zijn de volgende producten:

- Plan van aanpak
- Definitiestudie
- Pilotontwikkelplannen
- Testrapporten

Tevens worden er producten opgeleverd die het uiteindelijke product vormen of daar een ondersteuning in zijn:

- Module ‘Facturatie’
- Module ‘Beheer’
- Handleiding module ‘Facturatie’
- Handleiding module ‘Beheer’

Reeds in de opstartfase heb ik ervoor gekozen om voor alle op te leveren producten precies vast te leggen welke eisen er aan de documenten worden gesteld. Dit heb ik gedaan om zo de opgeleverde documenten te kunnen verifiëren.

2.6 Welke technieken worden gebruikt

Om verdere invulling te geven aan de ontwikkelmethode die wordt gehanteerd wordt er gebruik gemaakt van zogenaamde technieken. U kunt hierbij denken aan interview technieken, objectmodellering of taakanalyse. Voorafgaande dit project moest ik vastleggen welke technieken ik bij voorbaat ging toepassen. Ik heb ervoor gekozen zoveel mogelijk technieken te kiezen waarmee ik al reeds ervaring heb. Het zijn technieken die over het algemeen in elk project inpasbaar zijn. De volgende technieken zullen van toepassing zijn op dit project:

- Interview
- Workshop
- OMT
- Taakdiagrammen
- Use-Cases
- Gebruikerstesten
- Relationeel database model
- DFD (*Data Flow Diagram*)
- Toestandsdiagrammen
- Beslissingsmodellen

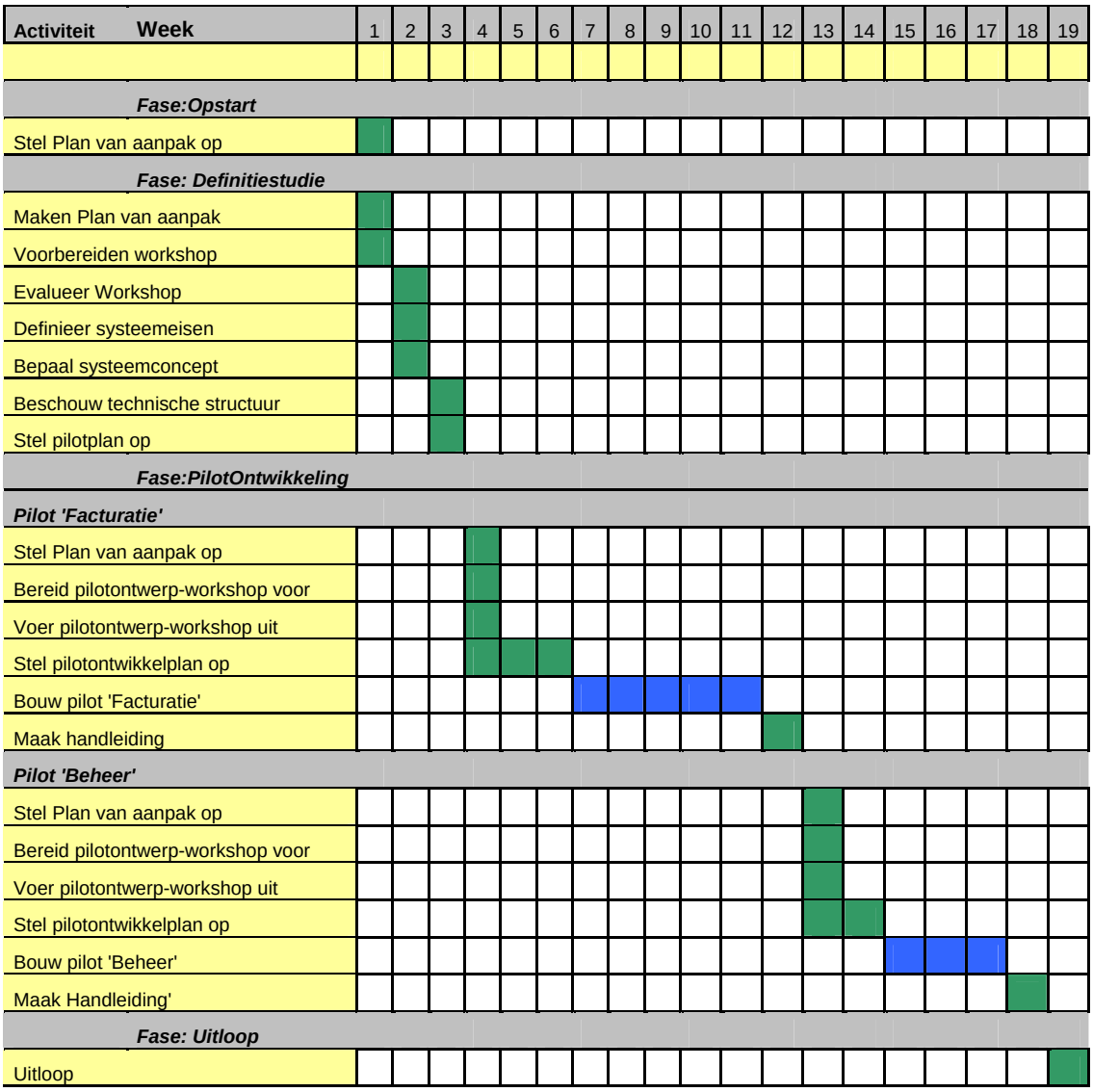
Indien een dergelijke techniek daadwerkelijk wordt toegepast zal ik de keuze voor de betreffende techniek verder beargumenteren.

2.7 De planning

Dit hoofdstuk heeft als doel de activiteiten, die door IAD worden voorgeschreven, binnen de looptijd van dit project in te plannen. Het inplannen van deze activiteiten ervaar ik telkens weer als zeer lastig, omdat je vooraf nooit zeker weet waar de mogelijke knelpunten zich bevinden. In de loop van dit deze paragraaf zal ik beschrijven hoe ik aan deze planning ben gekomen.

Voor ik kan overgaan tot plannen heb ik moeten wachten tot ik de juiste planning van de afstudeerperiode van school aangeleverd kreeg. Toen ik deze had bleek de tijd die ik voor het

afstuderen heb slechts 19 weken te zijn, althans op papier. Er van uitgaande dat ik geen vrije dagen opneem hou je 91 (*19 * 5, – 4 feestdagen*) werkdagen over. Dit betekent dus 18 weken voor het realiseren van de opdracht.



Figuur 2.2 Gedetailleerde planning

In tegenstelling tot voorgaande projecten heb ik vrij weinig tijd ingepland voor het maken van de definitiestudie. Dit was een direct gevolg van de gekozen ontwikkelstrategie. Bij het incrementeel ontwikkelen wordt namelijk meer invulling gegeven aan het pilotontwikkelplan dan aan de definitiestudie, dit komt doordat de ontwikkelstrategie de ontwikkelaar in staat stelt zo snel mogelijk tot een ontwerp van de software te komen. In totaal heb ik 2½ week uitstaan voor het opstellen van de definitiestudie.

De pilotontwikkeling omvat het grootste deel van het project, namelijk 14 weken. Binnen deze 14 weken zijn 9 weken besteedt aan de pilot ‘Facturatie’ en 5 weken aan de pilot ‘Beheer’. De pilot ‘Facturatie’ heeft voor mij meer tijd gekregen, omdat deze pilot beschikt over ingewikkeldere processen, daar waar de pilot ‘Beheer’ beschikt over redelijk oppervlakkige processen.

Voor beide pilots is er een periode ingepland voor de realisatie van de pilot. Binnen deze periode vallen tevens de gebruikerstesten en mogelijke iteraties. Meer hierover kunt u lezen in *paragraaf 3.8*, Het pilotplan. Binnen deze paragraaf kunt u onder andere lezen wanneer het eerste testmoment zal plaatsvinden.

DEEL II

De Definitiestudie

3 Definitiestudie

Binnen de fase definitiestudie worden de doelen en beperkingen van het systeem in kaart gebracht. Tevens zal in deze fase worden aangeduid hoe het verdere verloop van het project eruit zal komen te zien.

Voordat de doelen en beperkingen vastgelegd kunnen worden dien ik te beschikken over de juiste informatie. Hoe ik aan deze informatie ben gekomen en waarvoor ik het ga gebruiken kunt u lezen in *paragraaf 3.2*.

Het beschouwen van de doelen en beperkingen bestaat uit het beschrijven van de huidige situatie, de systeemeisen, de gewenste situatie en de technische-architectuur. De huidige situatie heb ik opgesplitst in twee gedeelten, namelijk huidige situatie Key-Task en huidige situatie eindgebruikers.
De huidige situatie Key-Task geeft mij inzicht in het huidige Key-Task systeem dit is onder andere noodzakelijk om de technische-architectuur te kunnen samenstellen. De huidige situatie van de eindgebruikers daarentegen richt zich op het werkelijke probleem dat opspeelt bij de eindgebruikers.

Tenslotte zal in *paragraaf 3.8* het pilotplan worden beschreven. Hierin staat vermeldt hoe het verdere verloop van het project eruit ziet.

3.1 Planning

De fase definitiestudie bestaat in totaal uit 12 werkdagen, zoals staat beschreven in het globale plan van aanpak:

Activiteit per dag	1	2	3	4	5	6	7	8	9	10	11	12
Stel plan van aanpak op												
Bereid interview & workshop voor												
Interview & workshop houden												
Evalueer interview & workshop												
Beschouw ontwikkelscenario												
Huidige situatie in kaart brengen												
Defineer systeemeisen												
Bepaal systeemconcept												
Beschouw technische structuur												
Beschouw organisatorische inrichting												
Stel pilotplan op												
Uitloop												

Figuur 3.1 Planning fase definitiestudie

Binnen het plan van aanpak heb ik een keuze gemaakt welke activiteiten er worden uitgevoerd. Deze activiteiten worden beschreven en aangeraden door IAD. Ik kan stellen dat de aangeraden activiteiten grotendeels terugkomen in de definitiestudie. Binnen deze fase zijn alle activiteiten afhankelijk van het interview. Ik heb dan ook voldoende tijd ingepland om mijzelf goed voor te bereiden op dit interview. Hoe dit in zijn werk is gegaan beschrijf ik in *paragraaf 3.2*.

3.2 Inwinnen van informatie

Voordat de doelen en de beperkingen kunnen worden gedefinieerd moet er eerst informatie vanuit de betrokken partijen worden ingewonnen. Deze informatie heeft betrekking op zowel de opdrachtgever als de gebruikers. Van de eindgebruikers wil ik graag weten hoe de huidige situatie eruit ziet en welke eisen ze graag doorgevoerd willen hebben. De opdrachtgever heb ik nodig voor het achterhalen van de systeemeisen. Dit betekent dat de informatiewinning volgt vanuit twee verschillende partijen.

Naar mijn mening verdient het inwinnen van informatie bij de opdrachtgever een andere aanpak dan het inwinnen van informatie bij de eindgebruikers. Daar waar de opdrachtgever een individu betreft sluit ik het houden van een workshop uit. De functie van een workshop komt naar mijn mening pas tot zijn recht bij minimaal drie deelnemers.

Het bedrijf Darumath bestaat echter uit een vijftal medewerkers die ieder hun eigen ideeën hebben over het Key-Task systeem. Om deze vijf medewerkers apart van elkaar te interviewen en vervolgens tot één concept te komen is een vrij ingewikkelde en tijdrovende zaak. Deze gebruikersgroep kan ik naar mijn mening dus het beste benaderen aan de hand van een workshop, waarbij ze op een geleidelijke wijze tot een centraal standpunt kunnen komen over de doelen en beperkingen van het systeem.

Voorafgaande aan de workshop hebben de medewerkers van Darumath gebruik mogen maken van het huidige Key-Task systeem. De intentie hiervan was om de medewerkers van Darumath bewust te laten worden wat de functie is van Key-Task en te laten beseffen waar de gebreken liggen. Tijdens de workshopsessie bleek eens te meer dat iedere deelnemer het probleem uit een andere invalshoek bekeek. Waaruit bleek dat het houden van een workshop een uitstekende manier is om de medewerkers tot een eenduidig standpunt te laten komen.

Een interview wil ik gebruiken om de opdrachtgever, die ook zijn inspraak heeft in dit project, te ondervragen. Met het interview wilde ik vooral de opdrachtgever zelf aan het woord laten. Juist daaraan kan ik beoordelen wat zijn gedachtegoed inhoud wat betreft de te ontwikkelen modules en de aanpak daarvan. Ik heb daarom geprobeerd het stellen van suggestieve vragen achterwegen te laten en het interview te laten leiden doormiddel van het stellen van open vragen. Voorbeelden van dergelijke vragen waren:

- Waar moet het systeem aan voldoen als het gaat om performance?
- In welke mate dienen de te ontwikkelen modules qua stijl aan te sluiten op het Key-Task systeem?

Het stellen van openvragen heb ik gedaan, omdat ik van mening ben dat suggestieve vragen de geïnterviewde uitnodigt tot kort geformuleerde antwoorden. Terwijl ik juist zoveel mogelijk informatie uit zijn antwoord wil halen.

3.3 Huidige situatie eindgebruikers

Het uiteindelijke doel van dit project is het oplossen van een bepaald probleem, die speelt bij de huidige implementatie van het Key-Task systeem. Om dit te bewerkstelligen dient de probleemstelling, zoals deze staat beschreven in *paragraaf 2.2.2*, nauwkeurig te worden geanalyseerd. Aan de hand van de workshop, die ik heb gegeven, heb ik de huidige situatie in kaart kunnen brengen.

Darumath gebruikt Key-Task ter ondersteuning van hun projecten(*opdrachten*). Mogelijke voorbeelden van dergelijke projecten zijn:

- Maken van softwaretoepassingen.
- Verhelpen van systeemfouten.
- Maken van klantspecifieke uitbreidingen, zoals: rapporten of formulieren.

Indien zich een nieuw project aanbiedt dan wordt daarvoor binnen Key-Task een project aangemaakt. Het project beschikt over gegevens zoals *Titel, Beschrijving, Uitvoerder, Startdatum, Einddatum, Status, etc.*
Op een project kunnen vervolgens kosten worden geboekt. Key-Task onderscheid daarbij twee soorten kosten, namelijk:

- Taken
- Materialen

Met een taak wordt bedoeld de tijd die een medewerker besteedt aan een bepaald project. Aan de hand van de taak kan bepaald worden wat de gerelateerde kosten zijn. Een materiaal omvat in principe alle overige kosten die betrekking hebben op het betreffende project. U kunt hierbij denken aan *papier, promotie materiaal of reiskosten*.
Indien een project is afgerond, dan kan deze worden afgesloten. Het gevolg hiervan is dat de status van het project naar ‘Afgesloten’ gaat en er verder geen wijzingen op het project en de gerelateerde taken en materialen kunnen worden doorgevoerd.

De verdere beschrijving van de huidige situatie die binnen dit hoofdstuk staat beschreven heeft enkel betrekking op de processen Factureren en Beheren.

In *figuur 3.2* ziet u de huidige situatie voor wat betreft het factureren. Voor het analyseren van het facturatieproces is het handig om een bepaalde modelleringtechniek te gebruiken. Dit heeft als voordeel dat een bepaald proces eenvoudiger te begrijpen is. Er zijn tal van technieken die je kunt gebruiken om een huidige situatie te beschrijven. De technieken die bij mij bekend zijn, zijn vooralsnog taakdiagrammen(*Guide*) en use-cases(*UML*). Van deze twee technieken heeft de taakdiagram mijn voorkeur, omdat ik de modellering overzichtelijker vind. Daarbij spreekt de hiërarchie, die bij taakdiagrammen van toepassing zijn, mij het meest aan en stelt mij in staat een proces sneller en beter te interpreteren.

De volgende figuur beschrijft het proces 'Aanmaken factuur'. Het aanmaken van een factuur gebeurt op basis van kosten die zijn gemaakt met betrekking tot een project.

Figuur 3.2 Taakdiagram: Aanmaken factuur

In *figuur 3.2* kunt u zien dat de gebruiker de volgende stappen moet doorlopen om een factuur aan te maken. Allereerst dient de gebruiker het betreffende project, materiaal of taak te openen. Aan de hand daarvan kan de gebruiker bepalen welke kosten er mee gemoeid zijn en zodoende deze te verrekenen. Vervolgens opent de gebruiker een Word-template, die de vaste stijl van de factuur bevat, om zo de factuur verdere inhoud te geven.

Kijkend naar *figuur 3.2*, dan ziet u dat er een aantal handmatige handelingen op het proces van toepassing zijn. Zoals het openen van Microsoft Word, om vervolgens handmatig de factuur gegevens in te vullen. Buiten het feit dat deze handelingen de gebruikers een hoop tijd kost, is er een risico aanwezig op menselijk fouten. Ik doel hier op:

- Schrijffouten
- Interpretatiefouten
- Berekeningsfouten

Elk van deze fouten kunnen tot gevolg hebben dat een factuur niet correct wordt aangemaakt. Nou is het bij Darumath zo dat alle uitgaande facturen nauwlettend worden gecontroleerd door een administratief medewerkster, zodat de kans dat er een incorrecte factuur bij een klant verzeild raakt heel erg klein is. Echter botsen we hier wederom op het principe dat deze handelingen overbodig zijn op het moment dat deze worden geautomatiseerd.

Wanneer een factuur is gecreëerd en goedgekeurd, dan dient deze te worden bezorgt bij de klant. Het bezorgen van facturen gebeurt in alle gevallen per post. Het posten van een factuur wordt gedaan door dezelfde persoon die de facturen controleert. Bij het versturen van een

factuur wordt er een kopie van de factuur in een daarvoor bestemde map gedaan. In deze map zitten alle verstuurde facturen. Het versturen van facturen per post heeft een aantal nadelen ten opzichte van het digitaal versturen. Digitale verzending neemt namelijk veel minder kosten met zich mee en verloopt ook nog eens stukken sneller. Daarbij komt dat er steeds meer klanten interesse krijgen in het hanteren van digitale formulieren, waarvan een factuur onderdeel van uitmaakt.

Doordat binnen het huidige Key-Task systeem niet mogelijk is de systeeminstellingen op applicatieniveau aan te passen, waren de medewerkers genoodzaakt de systeeminstellingen direct te wijzigen in de Key-Task database. Darumath heeft voldoende expertise in huis om deze instellingen te wijzigen, maar dat wil niet zeggen dat er geen probleem is met betrekking tot het kunnen wijzigen van de systeeminstellingen. De integriteitbewaking laat geheel te wensen over. Het gevolg hiervan is dat de instellingen kunnen beschikken over onjuiste gegevens. De consequenties op het Key-Task systeem zijn daardoor niet te overzien. Het systeem kan hierdoor crashen of inconsistent raken.

Op het eerste gezicht bestaat het systeem uit zeer veel instellingen. Het is dus verstandig deze instellingen te categoriseren, zodat de gebruiker sneller bij de benodigde instelling komt. Ik onderscheid hierbij dan ook de volgende systeeminstellingen:

- META-Gegevens
- Systeemvariabelen
- Systeemgebruikers
- Foutmeldingen

Met een META-gegeven wordt bedoeld: aanvullende definieerbare informatie omtrent een object. Ik zal u een voorbeeld geven om dit te verduidelijken:

Een project beschikt over een statuscode. Deze statuscode geeft aan in welk stadium het project zich bevindt, zoals open, toegewezen, afgesloten, geannuleerd, etc.
Deze statuscodes staan opgeslagen in een aparte tabel. Dat wil dus zeggen dat een organisatie haar eigen statuscodes kan aanpassen. De gegevens binnen de statuscode tabel worden door mij als META-gegevens opgevat.

Buiten de statuscode van een project vallen BTWcodes, landencodes, klantentypes ook tot het begrip META-gegevens.

Systeemvariabelen zijn parameters die op directe wijze de werking van het systeem beïnvloeden. Er kan hierbij gedacht worden aan:

Naam:	Beschrijving
SessieTimeOut	Geeft aan hoe lang een sessie geldig blijft.
MaxRecords	Geeft aan hoeveel resultaten een zoek procedure mag teruggeven.
OrganisationName	De naam aan van de gebruikende organisatie.
OrganisationLogo	De locatie van het logo van de gebruikende organisatie aan.
MaxUploadSize	De maximale bestandsgrootte aan van wat een gebruiker mag uploaden.

Tenslotte bestaan de systeeminstellingen uit het bijhouden van de gebruikers. Vanuit hier worden gebruikersnamen en wachtwoorden verstrekt.

3.4 Huidige situatie Key-Task

Doordat de te ontwikkelen modules een toevoeging zijn op een al bestaand systeem, is het voor mij als ontwikkelaar noodzakelijk te weten hoe het Key-Task systeem technisch in elkaar zit. Zodoende weet ik hoe ik de modules moet gaan koppelen aan het huidige systeem. Aan de hand daarvan kan ik de technische-architectuur, beschreven in *hoofdstuk 3.7*, samenstellen.

Ter verduidelijking: In dit hoofdstuk worden enkel de technische aspecten van het Key-Task systeem besproken. Voor overige vragen dien ik u te verwijzen naar de handleiding van Key-Task, die bij dit document is bijgevoegd.

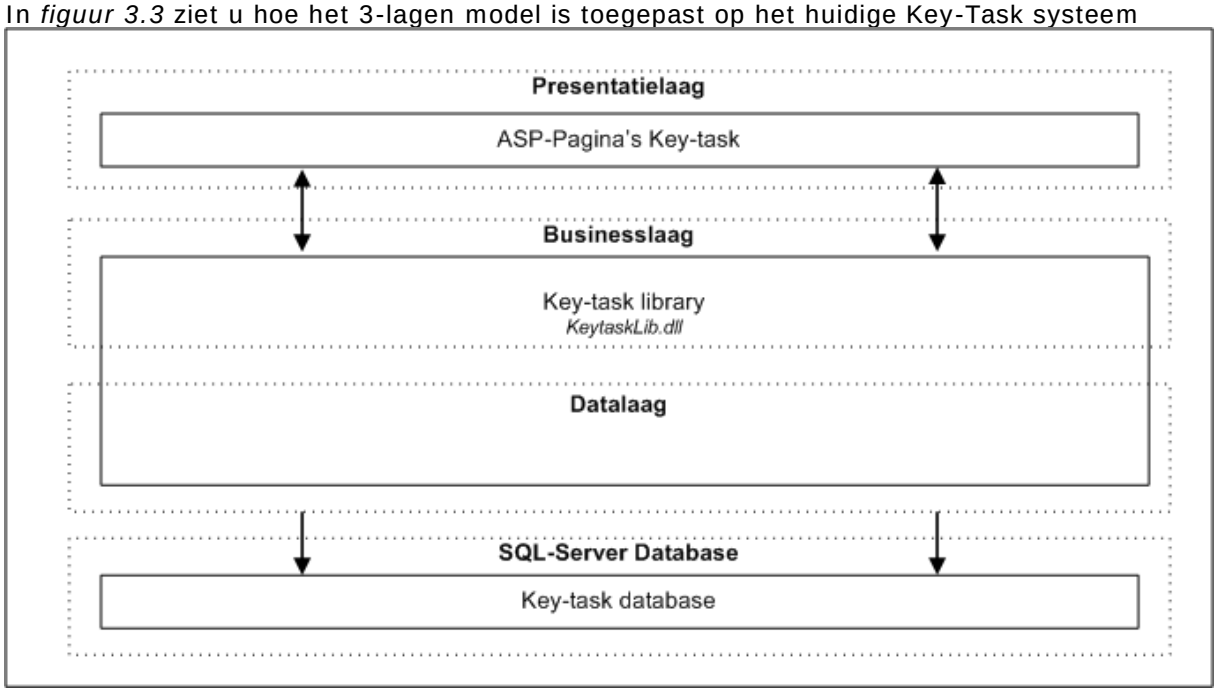
Om dit te bewerkstelligen heb ik de systeemdocumentatie van Key-Task nauwkeurig doorgelezen. Onder deze documentatie vallen de volgende documenten:

1. Definitiestudie Key-Task
2. Pilotontwikkelplannen Key-Task
3. Handleidingen Key-Task

Het huidige Key-Task systeem draait in de vorm van een webapplicatie, dit betekent dat deze via de internet browser Internet Explorer benaderbaar is. Het Key-Task systeem is opgebouwd aan de hand van een 3-lagen model.

Wat is een 3-lagen model?
Een model dat het systeem opdeelt in drie afzonderlijke gedeeltes. Elk van deze systeem gedeeltes hebben zo hun eigen functie.

In *figuur 3.3* ziet u hoe het 3-lagen model is toegepast op het huidige Key-Task systeem



Figuur 3.3 Technische-structuur Key-Task

In de bovenstaande figuur staan in totaal 4 lagen afgebeeld, namelijk Presentatielaag, Businesslaag, Datalaag en SQL-Server Database. De SQL-server Database reken ik echter niet als laag ten behoeve van de technische-architectuur, omdat ik deze laag erken als zijnde een extern component.

Zoals u kunt zien moeten deze lagen met elkaar kunnen communiceren om tot een werkend systeem te komen. Het is daarbij belangrijk om te beseffen dat de communicatie altijd wordt gestart vanuit de bovenliggende laag, dus de presentatielaag kan alleen de communicatie starten met de businesslaag en de businesslaag met de datalaag. Communicatie tussen de lagen kan alleen plaatsvinden indien ze, binnen de hiërarchie, op elkaar aansluiten. De presentatielaag kan dus niet direct communiceren met de datalaag.

De lagen uit *figuur 3.3* hebben de volgende functie:

Datalaag:

Deze laag heeft als doel de integriteit van het systeem te bewaken. Indien er schrijfacties worden uitgevoerd op de database, dan worden deze schrijfacties gecontroleerd, zodat er geen inconsistentie binnen de database optreedt. Ik zal een voorbeeld geven om dit te verduidelijken:

Wanneer een gebruiker uit het systeem wordt verwijderd, dan controleert deze laag eerst of deze gebruiker nog openstaande taken heeft. Is dit het geval dan wordt de schrijfactie geannuleerd.

Hierdoor voorkomt de datalaag dat er records in de database staan die niet meer van toepassing zijn. Tevens controleert deze laag of de gegevens, die worden weggeschreven naar de database, over het juiste formaat beschikken.

De datalaag komt binnen veel systemen voor in de vorm van Stored Procedures in het DBMS. Binnen Key-Task echter komt de datalaag voor in de klassenbibliotheek van Key-Task. Dit is gedaan omdat je op deze wijze niet afhankelijk bent van één specifiek DBMS, zoals SQL-Server of Oracle. De Stored Procedures van een DBMS zijn in de meeste gevallen uniek in hun syntax. Dat wil zeggen dat wanneer je een Stored Procedure uit SQL-server in Oracle zal draaien dit niet gaat werken. Het systeem verliest hierdoor zijn integriteitbewaking.
Door de datalaag op te nemen binnen de klassenbibliotheek van Key-Task maak je de datalaag onafhankelijk van een DBMS, waardoor het systeem met meerdere DBMS'en kan omgaan, zoals *SQL-Server, Oracle, Microsoft Access, My-SQl of Sybase.*

Businesslaag:

Deze laag bevat de eigenlijke functionaliteit van het systeem. In het geval van Key-Task zijn dit de systeemobjecten. Met systeemobjecten worden bedoeld de objecten die van toepassing zijn op het bedrijfsproces, waarvan zich meerdere instanties kunnen voordoen. U kunt hierbij denken aan objecten zoals *Project, Klant, Taak, Materiaal.*

De businesslaag kan alleen met de datalaag communiceren. De businesslaag heeft dus geen directe verbinding met de Key-Task database. Zodoende worden alle database transacties door de datalaag opgevangen.

Presentatielaag:

De presentatielaag heeft als doel de gegevens van het systeem te tonen aan de gebruiker. In het geval van Key-Task bestaat de presentatielaag uit tal van internet pagina's, die doormiddel van ASP *(Active Server Pages)* kunnen communiceren met de businesslaag.

De data –en businesslaag bevinden zich in een klassenbibliotheek, genaamd KeyTaskLib. Deze klassenbibliotheek bestaat op haar beurt uit een DLL-bestand *(Dynamic Link Library)*. Een DLL-bestand wordt onder het Microsoft Windows platform gebruikt om referenties te kunnen leggen naar de objecten binnen die klassenbibliotheek. Op deze wijze hebben ontwikkelaars de mogelijkheid reeds bestaande programmatuur te hergebruiken.

De interactie met de gebruiker zal gebeuren met behulp van JavaScript. JavaScript is een zogehete Client-Site script, dat wil zeggen dat het script wordt uitgevoerd door de browser. De presentatielaag moet ook worden gekoppeld aan de businesslaag. Om dit te bewerkstelligen maak ik gebruik van ASP (Active Server Pages). Naast de gebruikte programmeertalen zal de opmaaktaal HTML gebruikt worden om de gebruikersinterface te realiseren.

De voordelen van een dergelijk model zijn:

- Bij een goede opzet spaart het 3-lagen model ontwikkelmankracht. Éénmalige coding van alle functionaliteit, met krachtig hergebruik.
- Verdeel en heers strategie. Elke laag is op zich vrij makkelijk te ontwikkelen. Liever 3 eenvoudige delen dan één complex geheel.
- Kwaliteit. Voor elke laag kan een specialist zijn expertise inbrengen, een GUI Designer voor de gebruikersinterface, een Visual Basic programmeur voor de data –en businesslaag, en een database specialist voor de tabellen.

De businesslaag bestaat uit een tal van objecten die het systeem vertegenwoordigen. Je kunt ook wel zeggen dat de businesslaag beschikt over de eigenlijke functionaliteit van het systeem.

Voordat er wordt overgegaan aan het ontwikkelen van de beheermodule is het noodzakelijk om te begrijpen wat de verschillende systeeminstellingen inhouden. Binnen dit hoofdstuk, heb ik mij tevens beziggehouden met het analyseren van de systeeminstellingen, zodat ik in een later stadium van dit project weet welke systeeminstellingen van toepassing zijn. Doordat het

totaal aantal systeeminstellingen erg groot was, heb ik ervoor gekozen deze instellingen te groeperen. Deze groepering heb ik gemaakt op basis van hun plaats en functie binnen het Key-Task systeem. De volgende groepering heb is gemaakt:

- META-gegevens
- Systeemvariabelen
- Foutmeldingen
- Gebruikers
- Organisatie

Eén soort systeeminstelling wordt door mij omschreven als META-gegevens. Ik wil van de gelegenheid gebruik maken om uit te leggen wat ik versta onder META-gegevens.

META-gegevens is aanvullende informatie die aan een systeemobject, zoals *taak, materiaal of project*, kunnen worden gekoppeld en die definieerbaar zijn door de eindgebruiker. U kunt hierbij denken aan *BTW-codes, statussen, prioriteiten of projecttypes*. Het META-gegeven BTW vormt dan ook de volgende tabel.

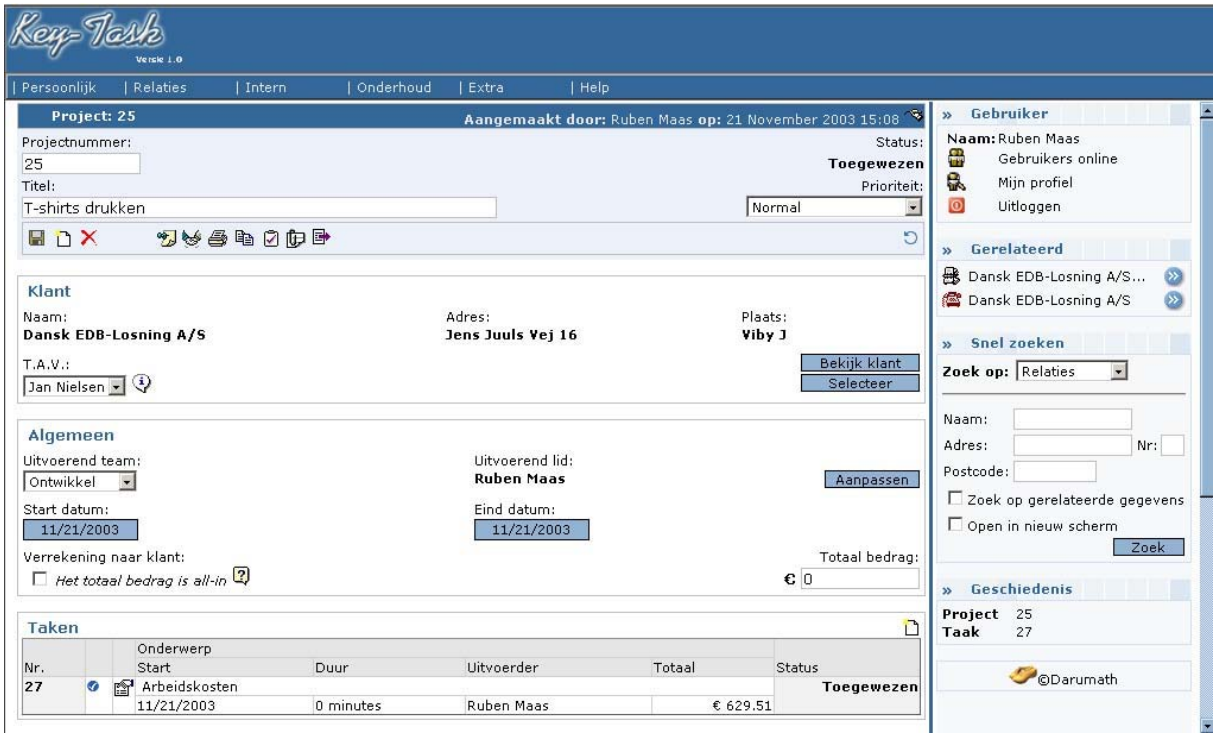
Code	Beschrijving	Percentage
BTW_Laag	Lage BTW tarief	6,00
BTW_Hoog	Hoge BTW tarief	19,00
BTW_Geen	Geen	0,00

Deze META-gegevens worden zodoende getoond in een dropdownlistbox zodat de gebruiker een BTW code kan selecteren. Binnen het Key-Task systeem zijn veel van deze gegevens in gebruik, maar tot op heden moeilijk aan te passen doordat de gebruiker de wijzigingen direct in de database moet doorvoeren.

Een belangrijk gemis in dit geheel is dat er binnen de huidige situatie geen controle mogelijk is bij het bewerken van de META-gegevens. Terugkijkend naar de BTW codes zou je misschien willen dat een gebruiker geen negatieve BTW percentage mee mag geven.

Nu bekend is hoe Key-Task technisch in elkaar zit, is het misschien wel interessant u te laten zien hoe Key-Task daadwerkelijk voor de gebruiker eruit ziet. In *figuur 3.4* ziet u het projectscherm van Key-Task.

Binnen het projectscherm heeft de gebruiker de mogelijkheid een project te beheren. Indien een project wordt uitgevoerd, dan worden daar altijd tijd en onkosten voor gemaakt. U kunt hierbij denken aan arbeidskosten of het gebruik van materialen. *Figuur 3.4* laat zien hoe dit daadwerkelijk op de gebruiker overkomt.



Figuur 3.4 Key-Task Projectscherm

De bovenstaande figuur geeft aan hoe de stijl van het huidige Key-Task systeem eruit ziet. Het scherm is opgesplitst in drie gedeeltes. Aan bovenkant van het scherm bevindt zich het navigatie gedeelte, die bestaat uit een popup-menu. Daaronder, aan de linker kant, bevindt zich het gedeelte waar de gegevens van het betreffende object worden afgebeeld, in dit geval van een project. De informatie binnen dit scherm zijn op haar beurt weer gecategoriseerd, zoals *Klant, Algemeen, Taken, Materialen*. De belangrijkste informatie staat helemaal boven afgebeeld, zoals *Titel, Status, Projectnummer, Prioriteit.* Om hier extra focus aan te geven bevindt deze informatie zich in een kader met een aparte kleur.

Tenslotte staat aan de rechterkant de zogenaamde taskbar, waarin veelgebruikte functies en extra informatie over het betreffende object worden getoond.

3.5 Systeemeisen

Na het in kaart brengen van de huidige situatie worden de systeemeisen gedefinieerd. Hiermee worden de behoeften van de toekomstige gebruikers weergegeven en daarnaast vormen de systeemeisen de basis voor alle ontwerpactiviteiten.

Allereerst heb ik het hoofdstuk systeemeisen onderverdeeld in een tweetal gedeeltes, namelijk: ‘Module Facturatie’ en ‘Module Beheer’.
Verder heb ik om het totaal aan systeemeisen overzichtelijker te maken de systeemeisen onderverdeeld in een vijftal categorieën, namelijk:

- Basissysteem-eisen
- Integriteits-eisen
- Performance-eisen
- Operationele-eisen
- Interface-eisen

Ik heb gekozen voor het hanteren van de bovenstaande categorieën, omdat elk van deze categorieën een bepaalt aspect van de pilotontwikkeling op zich kunnen nemen. Zo beschrijven de basissysteem-eisen de benodigde functionaliteit van de pilots. Op basis hiervan kan de functionele-structuur worden beschreven. De interface-eisen kunnen daarentegen weer gebruikt worden voor het ontwerp van de gebruikersinterface.

Ik zal van de gelegenheid gebruik maken een aantal van deze systeemeisen te behandelen. Wilt u de overige systeemeisen nogmaals inzien dan verwijs ik u naar *hoofdstuk 6* uit de definitiestudie. De volgende systeemeisen zijn onder andere van toepassing:

Module Facturatie	
Categorie	Beschrijving
Basis	De module moet een factuur kunnen genereren op basis van een project,taak of materiaal.
Integriteit	Een taak of materiaal mag alleen worden gefactureerd indien deze behoort tot een all-inclusive project.
Performance	Het zoeken van een factuur mag niet langer dan 1000 ms in beslag nemen.
Operationele	De functionaliteit binnen het systeem moet volledig herbruikbaar zijn.
Interface	De stijl van Key-Task dient gehandhaafd te worden.

Doordat de te ontwikkelen modules een toevoeging zijn op een reeds bestaand systeem, zal er rekening gehouden moeten worden met het huidige systeem. Vanuit de opdrachtgever is het duidelijk geworden dat de systeemeisen voor wat betreft interface, performance en Operationele-eisen gehandhaafd moeten worden. Dit om te voorkomen dat de te ontwikkelen modules leiden tot onder andere: stijlbreuk, lagere performance of veiligheidsschending. Het gevolg hiervan is dat er een aantal eisen direct zijn overgenomen uit de Definitiestudie van Key-Task.

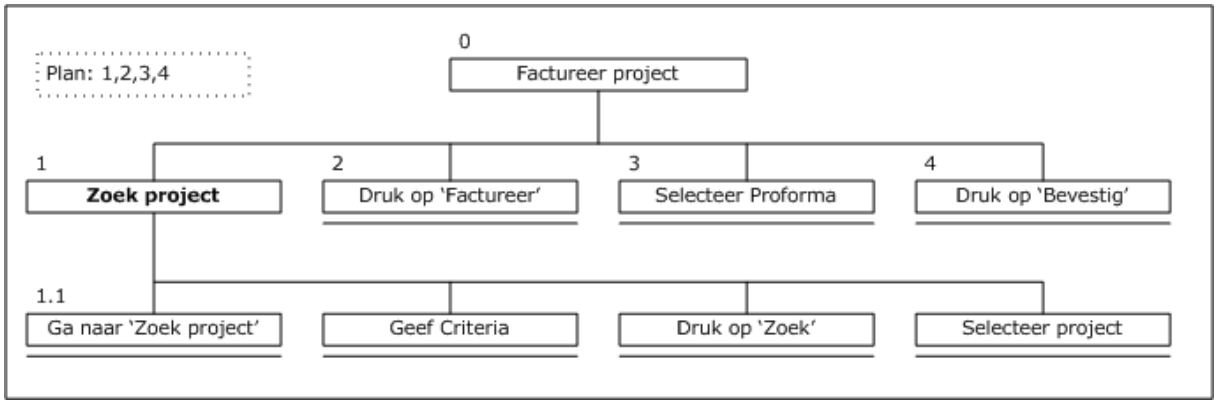
Eén van de performance-eisen houdt in dat het zoeken van facturen niet langer dan 1000 milliseconden in beslag mag nemen. Deze eis is in samenspraak met de opdrachtgever opgesteld naar aanleiding op de vraag wat toelaatbaar is. Deze eis zal aan de hand van een test worden beoordeeld of eraan is voldaan. De testmethode is hierbij erg van belang, omdat ik eventuele gebreken wat betreft de performance of integriteit in een zo vroeg mogelijk stadium wil opsporen. Een gebruikerstest sluit ik hierbij dan ook uit. In *paragraaf 3.8* kunt u lezen welke testmethodes ik zal hanteren.

Tenslotte dient de interface van de te ontwikkelen modules aan te sluiten bij de stijl van het huidige Key-Task systeem. Een vrij logische eis wat betekent dat de bestaande Key-Task schermen de grondslag vormen voor het schermontwerp.

3.6 Gewenste situatie

Binnen de gewenste situatie beschrijf ik een tal van processen, die ik reeds heb behandeld in *hoofdstuk 3.2*, zoals die na uitvoering van dit project moeten gaan verlopen. Je kan dus zeggen dat binnen de gewenste situatie een aanzet wordt gedaan naar de oplossing van het probleem.

Het kernprobleem van de probleemstelling is het feit dat er binnen de huidige situatie teveel menselijke handelingen aanwezig zijn om het proces vlot en vlekkeloos te laten verlopen. Dit leidt ertoe dat het proces vertraging oploopt met alle negatieve effecten ten gevolgen. De gewenste situatie heb ik dan ook gebruikt om dit probleem aan te pakken. Dat betekent dat de menselijke handelingen moeten worden omgezet naar geautomatiseerde processen.



Figuur 3.5 Taakdiagram: Factureer Project

Figuur 3.5 laat een vrij eenvoudig proces zien over het factureren van een project. Om een project te kunnen factureren moet het betreffende project worden opgezocht binnen het systeem. Indien het project is gevonden dan kan deze worden geopend waarna het

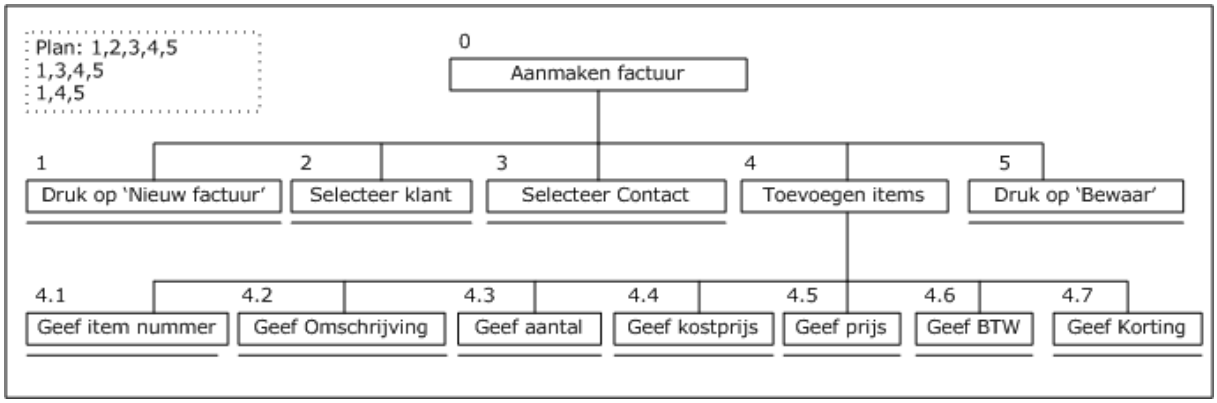
projectscherm wordt geladen. Vanuit hier kan op de knop ‘Factureer’ worden gedrukt. Vervolgens verschijnt er een scherm met daarin een lijst met zogehete pro forma-facturen.

Een pro forma-factuur is een factuur die nog niet is verzonden naar een klant en dus ook nog niet goedgekeurd is door een daartoe bevoegde medewerker. Ik heb hiertoe besloten omdat het controleren van een factuur op correctheid in sommige gevallen niet geheel geautomatiseerd kan worden. Buiten het feit dat een door het systeem gegenereerde factuur kostenmatig correct in elkaar steekt, kan een betreffende organisatie over een specifiek beleid beschikken waardoor de kostenbegroting kan verschillen. In zulke gevallen dienen achteraf handmatige wijzigingen worden doorgevoerd op een door het systeem gegenereerde factuur.

Gezien dit feit heb ik ervoor gekozen om een factuur een bepaalde status te laten hebben, zoals:

- **Pro forma** Een tijdelijke factuur die wacht om gecontroleerd te worden, deze factuur kan verwijderd worden uit het systeem.
- **Verzonden** Een definitieve factuur die niet meer uit het systeem verwijderd kan worden en die reeds is verzonden naar de klant.
- **Betaald** Een definitieve factuur die niet meer uit systeem verwijderd kan worden en reeds door de klant is betaald.
- **Geannuleerd** Een tijdelijke factuur die niet door de goedkeuring is gekomen. Deze factuur kan verwijderd worden uit het systeem.

Aan de hand van de gedefinieerde systeemeisen is het van belang de gebruiker handmatige facturen te laten samenstellen. Dat wil zeggen dat er tevens facturen aangemaakt dienen te worden die niet in relatie staan met een project, taak of materiaal. *Figuur 3.6* beschrijft dit proces.



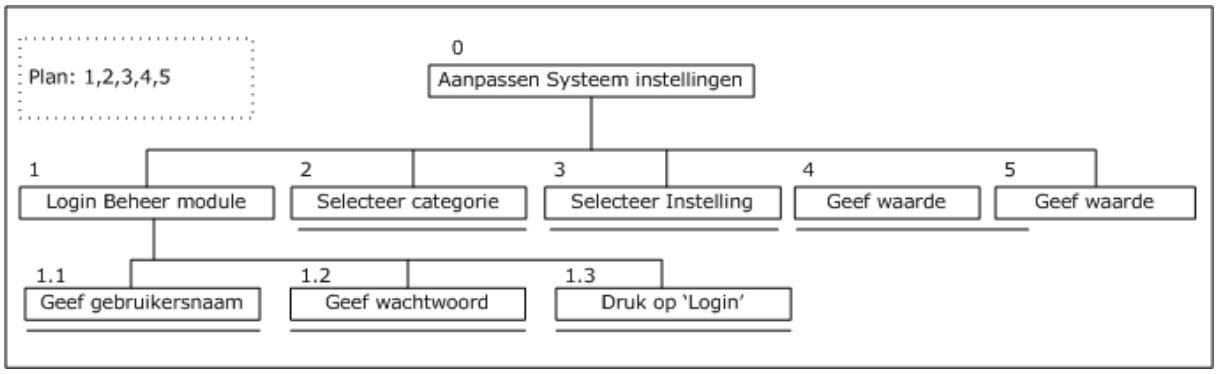
Figuur 3.6 Taakdiagram: Aanmaken factuur

Het aanmaken van een handmatige factuur wordt in werking gesteld door middel van het drukken op de knop ‘Nieuw factuur’. Deze knop is vanuit diverse schermen te benaderen. Eenmaal in het factuurscherm beland heeft u de mogelijkheid om een klant te selecteren. Als u vanuit het project, taak, materiaal of klantscherm op ‘Nieuw factuur’ drukt dan is de klant reeds bekend en kan deze stap worden overgeslagen. Vervolgens kunt u kiezen naar welke persoon de factuur gericht moet zijn. Indien de klant bekend is dan is hier standaard een contactpersoon ingevuld, deze kan uiteraard alsnog worden gewijzigd.

Vervolgens moet de gebruiker de factuuritems meegeven. Deze items bestaan uit de volgende gegevens:

- Itemnummer
- Aantal
- Kostprijs
- Prijs
- BTW percentage
- Korting percentage

Het benaderen van de systeeminstellingen kan alleen worden gedaan door een gebruiker van het type ‘Administrator’. *Figuur 3.7* laat dit proces zien.



Figuur 3.7 Taakdiagram: Aanpassen systeeminstellingen

Wanneer een gebruiker de systeeminstellingen wilt wijzigen dan dient deze persoon eerst in te loggen in de beheers module. Aan de hand van de gebruikersnaam en wachtwoord kan het systeem bepalen of de gebruiker een Administrator is. Zo ja, dan wordt het hoofdscherm getoond. Binnen dit systeem staan alle systeeminstellingen gecategoriseerd. Als er op een categorie gedrukt wordt dan wordt ee scherm geopend waarin de gebruiker de instelling kan wijzigen.

3.7 Technische-architectuur

De Technische-architectuur heeft als doel de fundamenten van het te ontwikkelen systeem te beschrijven. Er wordt hier gekozen voor een manier waarop het systeem zal worden opgebouwd. Tevens wordt de technische-architectuur gebruikt om de verschillende pilots en bouweenheden samen te stellen, omdat uit de technische-architectuur de verschillende systeem onderdelen herkend kunnen worden.

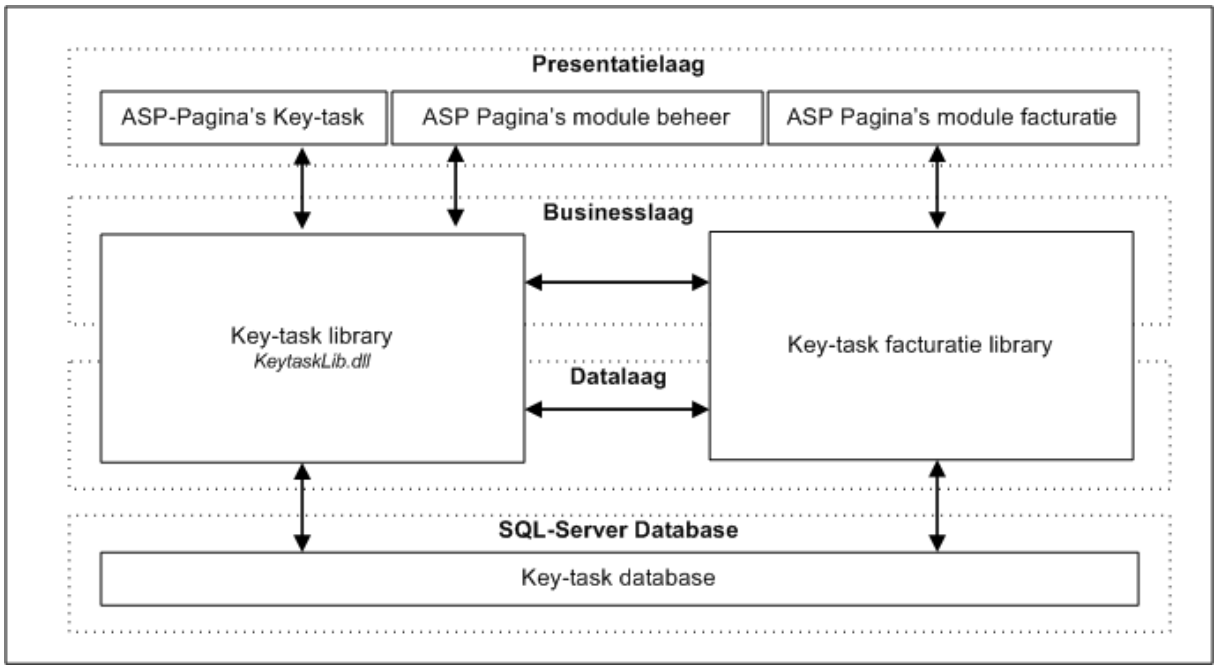
Dit hoofdstuk heeft alleen betrekking op de technische-architectuur van de software. Doorgaans is het gebruikelijk ook de technische-architectuur van de hardware te beschrijven, dat wil zeggen dat de positie van het systeem binnen de informatie infrastructuur wordt omschreven. Dit vond ik echter niet van toepassing voor mijn project. Dit project heeft namelijk geen enkele invloed heeft op de reeds omschreven technische-architectuur van de hardware, die zich bevindt in de definitiestudie van Key-Task.

Doordat de te ontwikkelen modules een toevoeging zijn op een bestaand systeem dien ik als ontwikkelaar rekening te houden met de huidige technische-architectuur. Een van de voordelen van de huidige technische-architectuur is dat het zeer geschikt is om uitbreidingen op te doen. Dit komt doordat het systeem bestaat uit een zogenoemd 3-lagen model, meer hierover kunt u lezen in *paragraaf 3.4*

Na de huidige technische-architectuur te hebben bestudeerd heb ik besloten dit model ook toe te passen op de te ontwikkelen modules. Daarvoor gelden de volgende argumenten:

- Het huidige model is uiterst geschikt voor aanpassingen en uitbreidingen.
- Persoonlijk heb ik veel ervaring met het ontwikkelen aan de hand van een 3-lagen model.
- Vind ik het te risicovol twee verschillende modellen toe te passen op één en dezelfde systeem.

Nu ik heb besloten welk model ik ga toepassen, voor wat betreft de technische-architectuur, dien ik verdere invulling te geven aan de diversen lagen van het systeem. *Figuur 3.8* laat de indeling zien.



Figuur 3.8 Technische-architectuur gewenste situatie

In *figuur 3.8* ziet u de technische-architectuur afgebeeld. Heel duidelijk hierbij is dat het systeem uit 3-lagen (*met de databaselaag niet meegerekend*) bestaat. De presentatielaag wordt uitgebreid met een tweetal sets van ASP-pagina's, die beiden de interface van de betreffende module vertegenwoordigen. Het moge duidelijk zijn dat zonder deze laag geen interactie mogelijk is tussen systeem en gebruiker.

De business -en datalaag wordt uitgebreid met een aparte klassenbibliotheek voor de facturatiemodule. Deze wordt op geheel dezelfde wijze ontwikkeld als de huidige Key-Task klassenbibliotheek. De module beheer wordt zodoende toegevoegd aan de huidige Key-Task klassenbibliotheek. In eerste instantie is het gebruikelijk eventuele toevoegingen binnen dezelfde klassenbibliotheek te verwerken. In dat geval waren de modules beheer en facturatie beiden in de Key-Task klassenbibliotheek verwerkt. Hierdoor worden eventuele communicatie stoornissen tussen diversen klassenbibliotheken vermeden. Echter was ik van mening dat ik de module facturatie liever in een aparte klassenbibliotheek wil opleveren. Doordat de module Facturatie over ingewikkelde processen beschikt, waarvan ik niet zeker weet of deze bij iedere organisatie op de juiste wijze van toepassing zijn, heb ik besloten deze als aparte klassenbibliotheek op te leveren. Dit maakt het systeem en zeker de betreffende module schaalbaarder. Indien een bepaalde organisatie een aanpassing wil voor wat betreft het factureren dan hoeft slechts enkel de betreffende klassenbibliotheek te worden aangepast. Hiermee wordt voorkomen dat het hele systeem ge-update moet worden. Dit besluit heb ik puur genomen met het oog op de toekomst en is dus niet noodzakelijk binnen het huidige project.

Verdere invulling aan de lagen wordt gegeven in de fase pilotontwikkeling startend vanaf *hoofdstuk 4.* De presentatielaag zal worden beschreven aan de hand van navigatieschema's en de business en datalaag worden aan de hand van object modellering belicht. Verder voorziet de pilotontwikkeling van een uitwerking welke toevoegingen er op de database worden toegepast.

3.8 Pilotplan

De laatste activiteit van de fase definitiestudie is het opstellen van het pilotplan. Het pilotplan heeft als doel in kaart te brengen hoe het verdere verloop van dit project eruit ziet. Binnen de pilotontwikkeling wordt verdere invulling gegeven aan de technische-architectuur zoals deze staat beschreven in de vorige paragraaf. Binnen deze paragraaf worden de ontwikkelactiviteiten opgedeeld in zogehete pilots.

3.8.1 Verdeling Pilots
Zoals reeds in de opdrachtomschrijving staat beschreven dien ik voor de uitvoer van deze opdracht een tweetal modules te ontwikkelen. Ik heb ervoor gekozen per module één pilot uit te voeren. Ik ben allereerst hiertoe gekomen omdat er een duidelijke scheiding heerst tussen de functionaliteiten die beiden modules te bieden hebben. Indien ik ertoe besloten had een pilot toe te passen, dan had ik het naar eigen zeggen mezelf te ingewikkeld gemaakt.

Een andere keuze had kunnen zijn dat ik de modules zou opsplitsen in meerdere pilots, maar ik vond dat daarmee teveel tijd gepaard ging in verband met de documentatie en het testen.

Kortom, de pilotindeling luidt als volgt:

- Pilot Facturatie
- Pilot Beheer

3.8.2 pilot Facturatie
Binnen de pilot 'Facturatie' zal een gedeelte van de probleemstelling worden opgelost. Deze pilot zal er voor zorgdragen dat de gebruikers in staat zijn op geheel geautomatiseerde wijze hun facturen te genereren en te versturen. Daarbuiten wordt het mogelijk gemaakt handmatig aanpassingen te verrichten op deze facturen. Het generen van deze facturen gebeurt op basis van een project, taak of materiaal die binnen het Key-Task systeem een te verrekenen bedrag aan de klant vertegenwoordigen. In de vorige paragraaf is reeds een aanzet gemaakt tot de ontwikkeling van deze module. Echter zal het pilotontwikkelplan 'Facturatie' ervoor zorgen dat aan deze structuur verdere invulling wordt gegeven.

De ontwikkeling van de pilot zal voorafgaan met een workshop, zodat er tot een overeenstemming gekomen kan worden over het verdere loop van deze pilot. De deelnemers van deze workshop zijn de medewerkers van Darumath.

3.8.3 pilot Beheer
Ook deze pilot heeft tot doel een gedeelte van de probleemstelling tot zijn rekening te nemen. De pilot 'Beheer' moet ervoor zorgdragen dat de daartoe toegewezen gebruiker de instellingen van het systeem kan wijzigen, zodat diegene het Key-Task systeem kan africhten naar haar/zijn organisatie.

3.8.4 Prioritering Pilots
Nu bekend is welke pilots uitgevoerd worden, is het aan mij als ontwikkelaar de taak aan te geven welke pilot de hoogste prioriteit krijgt. Het prioriteren van pilots is noodzakelijk indien de planning niet aangehouden kan worden door de ontwikkelaar. Mocht dit het geval zijn, dan wordt hiermee verzekerd dat de belangrijkste systeemdelen beschikken over voldoende kwaliteit.

De prioritering van de pilots heb ik gemaakt in samenspraak met zowel de opdrachtgever als de eindgebruikers. Beide partijen lieten blijken dat de pilot 'Facturatie' hun hoogste prioriteit genoot. Dit komt doordat het facturatie proces door beide partijen als zeer essentieel wordt gezien op het bedrijfsproces. Ik heb daar als ontwikkelaar op in gespeeld en de volgende prioritering vastgesteld:

- Hoogste prioriteit: Pilot 'Facturatie'
- Laagste prioriteit: Pilot 'Beheer'

3.8.5 Pilottesten
Binnen de ontwikkelmethode IAD is het testen van de pilot een zeer essentieel onderdeel. Een test zal uitwijzen of een pilot succesvol ontwikkeld is. Mocht dit niet het geval zijn dan biedt zich een nieuwe iteratie aan waarin de noodzakelijke aanpassingen worden gemaakt aan het systeem.

Er zijn heel erg veel testmethodes en elk van deze methode heeft zo weer zijn voor -en nadelen, waarbij elke methode weer is gericht op een type systeem en stadium van het project. Aan mij dus de keuze om tussen al deze testmethodes de juiste voor mijn pilots te kiezen. In het kader van de opleiding VIA heb ik van een aantal testmethodes gebruik mogen maken, zoals:

- De SUMI-test
- Cognitieve Walkthrough

- Thinking Aloud
- Use-Cases
- Heuristic Evaluation

Eerlijkheid gebied mij te zeggen dat de verschillen en toepasbaarheid van deze testmethodes mij in de loop van de tijd is ontschoten. Ik heb daarom deze testmethodes op een rijtje gezet en bekeken welke testmethode het beste te hanteren is. Dit heb ik gedaan aan de hand van een aantal criteria, namelijk:

- Werkwijze
- Toepasbaarheid
- Fase

Criteria	Beschrijving
SUMI	
Werkwijze:	De werkende versie van de software wordt beoordeeld door een tiental representatieve gebruikers. Op basis van de vragenlijst wordt de score van een product bepaald.
Toepasbaarheid:	Pas werkende software wordt getest, dus erg laat in het proces. Het vergelijken van producten en versies is mogelijk; wordt de usability beter of slechter.
Fase:	Gebruik
Use-Cases	
Werkwijze:	Door een beschrijving in scenario's wordt getoetst of de gebruikerstaken met het systeem kunnen worden uitgevoerd.
Toepasbaarheid:	Vanwege de benodigde inspanning (kosten hoog) zijn use cases alleen voor de belangrijkste functionaliteiten zinvol. Use cases worden ook gebruikt in analyse en ontwerpfase van software. Gebruik van use cases als toets- en testtechniek is met name zinvol als deze in de verschillende fasen van het ontwikkelproces hergebruikt worden.
Fase:	Pilotontwikkeling
Heuristic Evaluation	
Werkwijze:	Een kleine groep experts (2-5) beoordeelt onafhankelijk van elkaar de usability van een product op basis van de "ten usability Heuristic" (van Jakob Nielsen).
Toepasbaarheid:	Is al toepasbaar als de ontwerpen van de user-interface, taken en context (A.O.) klaar zijn.
Fase:	Pilotontwikkeling
Cognitieve Walkthrough	
Werkwijze:	In een Cognitieve Walkthrough worden aan de hand van een vragenlijst aspecten van de interface voor en na een interactie systematisch vastgelegd, ofwel door een gebruiker, ofwel door een expert.
Toepasbaarheid:	Circa 5 testpersonen wordt gevraagd gebruik te maken van het systeem
Fase:	Pilotontwikkeling
Thinking Aloud	
Werkwijze:	Thinking Aloud is een methode die informatie verschaft over de psychologische mechanismen en kennisstructuren die ten grondslag liggen aan veel cognitieve taken. Daarnaast kan de methode gebruikt worden om attitudes en meningen van proefpersonen te achterhalen.
Toepasbaarheid:	De testpersonen wordt gevraagd hardop te denken bij het uitvoeren van bepaalde taken. Aan de hand hiervan wordt een analyse gemaakt over de mogelijke knelpunten in de software.
Fase:	Pilotontwikkeling

De bovenstaande testmethodes zijn zogenaamde gebruikerstesten. De testen worden uitgevoerd aan het einde van iedere iteratie. Uiteraard zal ik ook in een vroeger stadium testen uitvoeren. Omdat de technische-architectuur uit verschillende lagen bestaat is het onverstandig deze met slechts één gebruikerstest te testen. Indien zich namelijk systeemfouten voordoen is het als ontwikkelaar lastig te achterhalen binnen welke laag van het systeem de fout opspeelt. Daarom dien ik bij iedere laag een test uit te voeren om te

kunnen beoordelen of de betreffende laag goed functioneert. Deze testen worden uitgevoerd op de data -en businesslaag doormiddel van een UNIT-test. Bij een UNIT-test wordt iedere procedure binnen het systeem getest op het juiste resultaat. Deze procedure zal uitgevoerd worden voordat ik een procedure oplever en zal bestaan uit een aantal scenario's. De procedure wordt pas opgeleverd indien alle scenario's succesvol zijn uitgevoerd. Het UNIT-testen wordt gehanteerd over zowel de pilot 'Facturatie' als de pilot 'Beheer'.

Nu ik de gebruikerstesten nogmaals heb doorgenomen kan ik bepalen welke gebruikerstest het best gebruikt kan worden voor de pilot 'Facturatie'. Allereerst wil ik dat de pilot wordt getest op functionaliteit, omdat de pilot ingewikkelde processen bevat en ik wil weten of de processen goed zijn doorgevoerd. Hierdoor zie ik sowieso af van de SUMI-test. Heuristic Evaluation is voor mij ook geen optie doordat ik het heel nadrukkelijk wil testen op de eindgebruikers en niet op experts. Ik wil er verzekerd van zijn dat het probleem van de eindgebruikers wordt verholpen. Voor mij als ontwikkelaar vallen de methodes Thinking Aloud en Cognitieve Walkthrough ook af doordat die testmethodes naar mijn mening te veel ongestructureerde gegevens met zich mee brengt. Dit heeft tot gevolg dat de testmethodes erg arbeidsintensief zijn. Ik heb daarom gekozen om de Use-Cases als testmethode te nemen voor wat betreft de pilot 'Facturatie'. Ik heb hiervoor gekozen, omdat Use-Case testen voor mij een snelle wijze is om de functionaliteit van de pilot te testen. Tevens is deze testmethode uitermate geschikt indien deze in verschillende fasen van het ontwikkelproces hergebruikt kunnen worden. Als testpersonen worden aangewezen de medewerkers van Darumath.

De pilot 'Beheer' wil ik op haar beurt testen door experts en niet door de reguliere eindgebruikers. Ik wil dat het beheer gedeelte van het systeem algemeen toegankelijk is voor administratoren van het systeem. Hierdoor heeft het voor mij geen nut deze af te richten op de reguliere eindgebruikers. De Heuristic Evaluation lijkt mij hiervoor een prima keuze, omdat testmethode afgericht is op experts van het systeem. Als test personen worden aangewezen:

- De opdrachtgever
- De systeembeheerder van Darumath
- De systeembeheerder van Metrix B.V.

Kortom, de volgende testmethodes worden toegepast voor de volgende pilots:

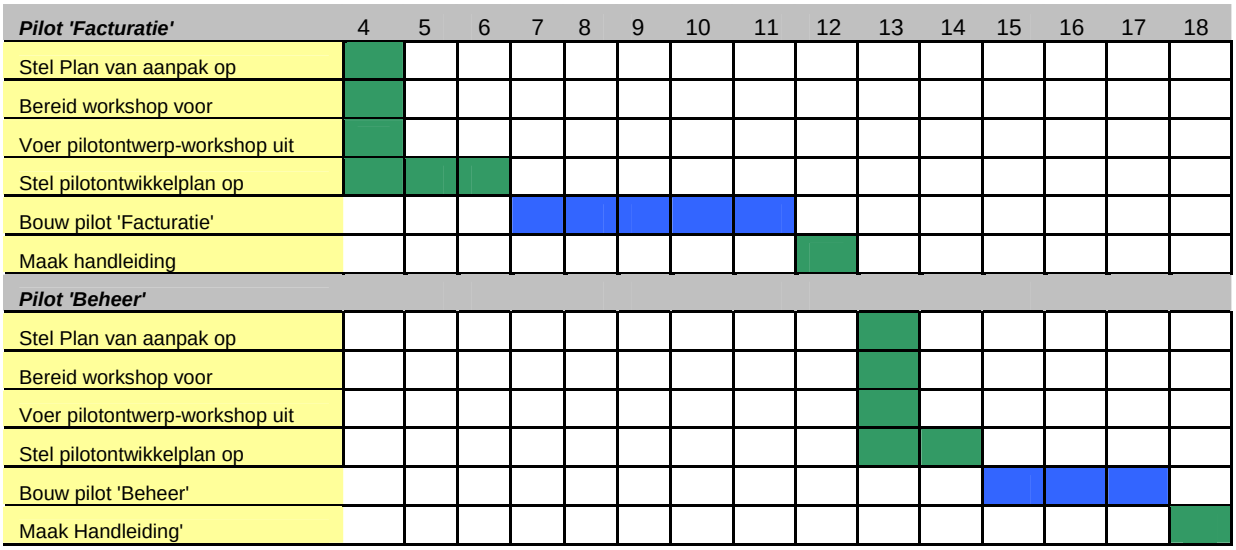
Testmethode	Pilot
UNIT-testen	Facturatie *(Alleen op data- en businesslaag)*
UNIT-testen	Beheer *(Alleen op data -en businesslaag)*
Use-Cases	Facturatie
Heuristic Evaluation	Beheer

3.8.6 Planning

Tenslotte is er nog een planning die aangeeft wanneer de pilots moeten worden uitgevoerd. Ik heb besloten om de pilots sequentieel te laten ontwikkelen, wat betekent dat ik de pilots achterelkaar in zijn geheel ontwikkel. Dit heeft mijn voorkeur, omdat ik zodoende mij beter kan richten op datgene wat ik moet ontwikkelen.

De pilot 'Facturatie' zal als eerst worden uitgevoerd, pas daarna zal de module 'Beheer' tot zijn beurt komen. In eerste instantie heb ik hiervoor gekozen omdat ik denk dat de pilot 'Facturatie' nieuwe systeeminstellingen met zich mee kan brengen. De nieuwe instellingen kan ik zodoende direct toepassen binnen de pilot 'Beheer'. Tevens heb ik hiervoor gekozen omdat de pilot 'Facturatie' mijn hoogste prioriteit geniet. De pilot 'Facturatie' wordt zodoende als eerste uitgevoerd. Wanneer de planning uitloopt, dan zal de pilot 'Beheer' in zijn planning moeten inleveren.

Zodoende ben ik gekomen tot de volgende planning:



Figuur 3.9 Planning pilotontwikkeling

Voor de pilot ‘Facturatie’ zijn 9 weken ingepland. Van die 9 weken worden er 5 weken besteedt aan het bouwen van deze pilot. Deze bouwfase is inclusief het testen van de pilot en de mogelijke iteraties. De eerste test, dus afsluiting eerste iteratie verwacht ik aan het begin van week 10.
De pilot ‘Beheer’ kost in totaal 6 weken, waarvan 3 weken de bouwfase betreft. Net als de pilot ‘Facturatie’ geldt ook hier het principe dat het testen en de mogelijke iteraties zijn meegenomen in de bouwfase. Waarbij ik de eerste test verwacht aan het einde van week 16.

Ik heb de pilot ‘Facturatie’ meer tijd gegeven, omdat:

- Deze pilot meer functionaliteiten bevat.
- Over een ingewikkelder process beschikt.
- Mijn hoogste prioriteit geniet.

Een gedetailleerde planning van de bouwfase wordt beschreven in de fase pilotontwikkeling. Nadat het systeem is ontworpen kan ik bepalen welke bouweenheden ik ga toepassen. Deze bouweenheden worden vervolgens met behulp van time-boxing ingepland.

DEEL III

De Pilotontwikkeling

4 Pilot Facturatie

Binnen dit hoofdstuk beschrijf ik het verloop van de fase pilotontwikkeling voor wat betreft de pilot 'Facturatie'. Alvorens ik kan beginnen met het bouwen van deze pilot zal ik eerst een ontwerp moeten maken. Dit ontwerp wordt bewerkstelligd aan de hand van een aantal activiteiten die IAD voorschrijft. Welke activiteiten ik heb uitgevoerd zal worden beschreven in *paragraaf 4.1*.

Wanneer het ontwerp vastligt ben ik gereed voor de realisatie van deze pilot. Ik zal in *paragraaf 4.2* beschrijven hoe de realisatie van deze pilot is verlopen.

Tenslotte worden in hoofdstuk 4.3 de gebruikerstesten en de mogelijke iteraties beschreven.

4.1 Pilotontwikkelplan

Het pilotontwikkelplan wordt gebruikt als specificatie van de te ontwikkelen pilot. Aan de hand van het pilotontwikkelplan weet de ontwikkelaar precies wat hij moet bouwen. Het pilotontwikkelplan is opgebouwd uit een aantal activiteiten:

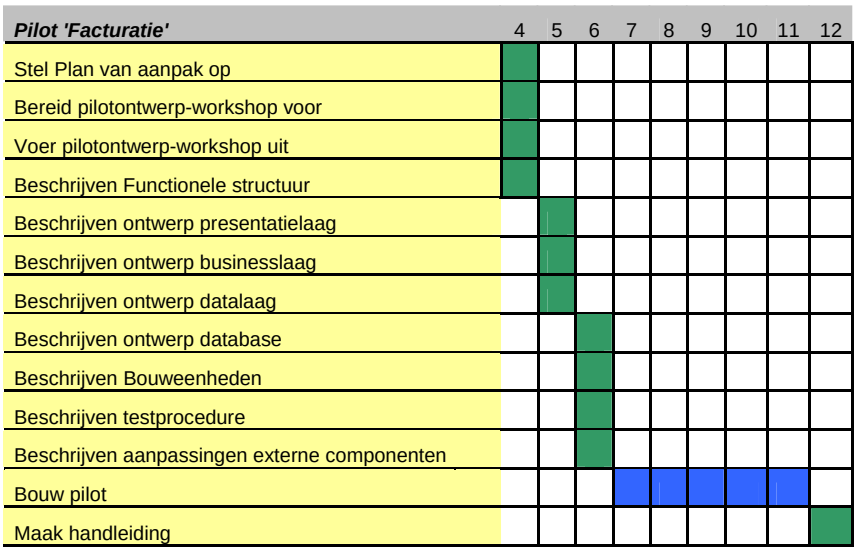
- Maken plan van aanpak
- Voorbereiden workshop
- Beschrijven Functionele structuur
- Beschrijven ontwerp presentatielaag
- Beschrijven ontwerp businesslaag
- Beschrijven ontwerp datalaag
- Beschrijven ontwerp database
- Beschrijven bouweenheden

Binnen dit hoofdstuk zal ik de bovenstaande activiteiten nader beschrijven.

4.1.1 Plan van aanpak

Het plan van aanpak dient als richtlijn voor het verloop van deze fase. Allereerst heb ik voor deze pilot een doelstelling vastgesteld. De doelstelling luidt als volgt:
De pilot 'Facturatie' moet ervoor zorgen dat de menselijke handelingen, binnen het huidige facturatieproces, worden geautomatiseerd.

Verder heb ik binnen het plan van aanpak een planning gemaakt, die staat afgebeeld in *figuur 4.1*.



Figuur 4.1 Dagplanning pilot 'Facturatie'

De activiteit beschrijven technische-structuur heb ik opgedeeld in drie gedeeltes, namelijk:

- Beschrijven ontwerp presentatielaag
- Beschrijven ontwerp businesslaag
- Beschrijven ontwerp datalaag

Ik heb ervoor gekozen te beginnen met het ontwerpen van de presentatielaag, waarna de business –en datalaag zullen volgen. Als ontwikkelaar heb ik de voorkeur om de functionaliteit definitief vast te stellen aan de hand van een statisch scherm. Hiermee wil ik voorkomen dat ik in een later stadium van deze fase, aanpassingen moet verrichten op datgene wat ik reeds eerder ontworpen heb. Terugkijkend naar de technische-architectuur van de te ontwikkelen module wil dat zeggen dat er vanaf boven naar beneden ontworpen wordt.

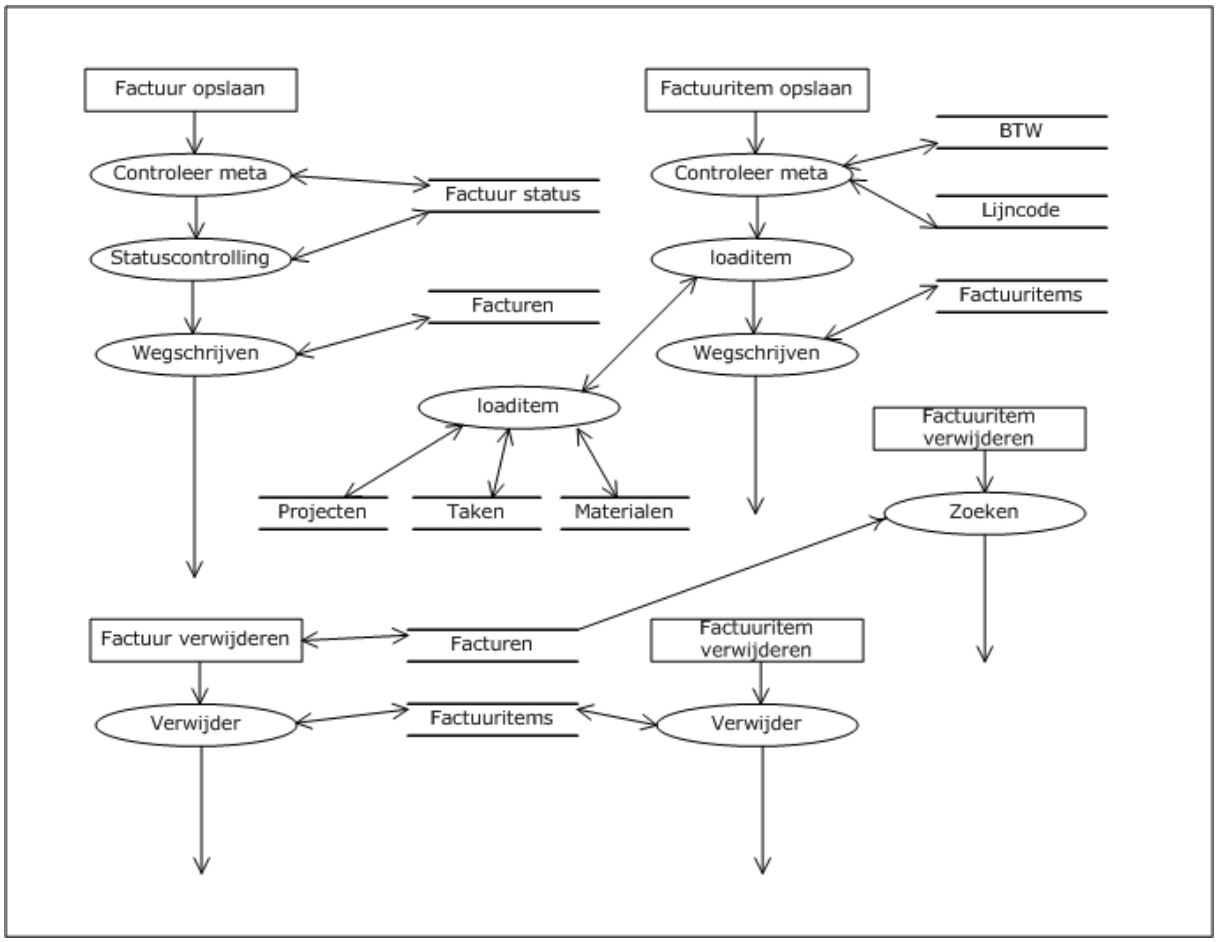
4.1.2 Workshop

Na het opstellen van het plan van aanpak is het tijd voor de workshop. Het doel van de workshop is om allereerst tot overeenstemming te komen met de eindgebruikers over de tot nu toe gemaakte keuzes. Tevens dient de workshop als gegevensbron voor de pilot 'Facturatie'. Binnen de workshop wilde ik graag een overeenstemming bereiken over hoe het facturatieproces binnen Key-Task geïntegreerd moet worden. Ik, als sessieleider, heb daarvoor een aanzet gedaan aan de hand van een navigatieschema, waarop de deelnemers konden reageren. Zodoende heb ik de schermen en hun functionaliteiten kunnen vastleggen.

4.1.3 Functionele Structuur

Met het beschrijven van de functionele-structuur probeer ik de functionaliteit van deze pilot in kaart te brengen. Dit moet op zodanig niveau van detail dat de ontwikkelaar hieraan een duidelijke, ondubbelzinnige basis voor de bouw van de pilot heeft.
De functionele-structuur is gemaakt aan de hand van de opgestelde definitiestudie en de ingewonnen informatie uit de workshop.

Er zijn een aantal technieken die ik kan gebruiken om de functionaliteit in kaart te brengen, ik gebruik een DFD (*Data Flow Diagram*). De keuze voor het gebruik van DFD's lag voor mij erg voor de hand, aangezien ik veel ervaring heb met het hanteren van deze techniek.



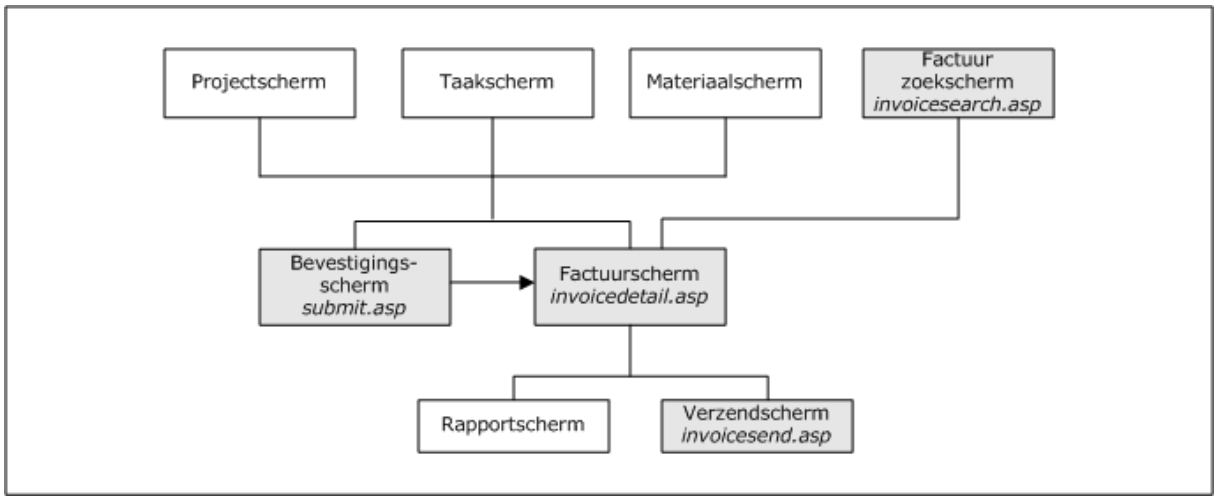
Figuur 4.2 DFD functionaliteiten pilot facturatie

Kijkend naar de DFD in *figuur 4.2* dan ziet u een aantal functionaliteiten afgebeeld, zoals *factuur opslaan, factuuritem opslaan, factuur verwijderen* en *factuuritem verwijderen*. Elk van deze functionaliteiten zijn afkomstig uit de systeemeisen die staan gedefinieerd in *hoofdstuk 6* uit de definitiestudie. De DFD laat zien wat er komt kijken bij een bepaalde functionaliteit, zoals welke actie/processen moeten worden uitgevoerd en welke gegevensbronnen worden er gebruikt. Zodoende vormt deze activiteit de basis voor alle overige ontwerp activiteiten in het kader de pilot 'Facturatie'.

4.1.4 Ontwerp presentatielaag

Binnen dit hoofdstuk staat beschreven hoe ik tot de gebruikersinterface, van de pilot 'Facturatie', ben gekomen. Het eerste wat mij te doen staat is te inventariseren welke schermen er gemaakt dienen te worden. Deze inventarisatie kan gemaakt worden aan de hand van onder andere de functionele structuur beschreven in *paragraaf 4.1.3.* Wanneer ik weet welke schermen er gemaakt moeten worden, dan ga ik over tot het schermontwerp. Hierin zal ik aangegeven hoe ik tot de interface ben gekomen.

Figuur 4.3 laat het navigatieschema zien. Dit schema toont de te ontwerpen schermen en hun onderlinge relatie.



4.3 Navigatieschema presentatielaag pilot 'Facturatie'

De grijs getinte entiteiten uit het navigatieschema houden in dat deze nog niet ontwikkeld zijn. De overige entiteiten worden al gebruikt binnen het huidige Key-Task systeem.

Centraal in dit schema staat het factuurscherm. Dit scherm kan op tal van manieren benaderd worden. Indien de gebruiker een project, taak of materiaal wilt factureren, dan komt hij of zij allereerst in het bevestigingsscherm terecht. Vanuit dit scherm kan de gebruiker kiezen of het item naar een pro forma factuur gestuurd moet worden of naar een geheel nieuwe factuur. Wanneer de gebruiker deze keuze heeft gemaakt dan komt hij in het factuurscherm terecht. Vanuit dit scherm kan zowel naar het rapportagescherm als het verzendscherm worden gegaan. Het rapportagescherm biedt de mogelijkheid een rapport te selecteren om deze vervolgens uit te printen. Het verzendscherm daarentegen verzorgt de verzending van de factuur per e-mail. Binnen dit scherm kan aangegeven worden naar wie de factuur verstuurd moet worden.

Het factuurscherm kan ook benaderd worden vanuit het factuur zoekscherm. Er wordt gezocht aan de hand van een aantal criteria zoals *factuurnummer, klantnummer, status, klantnaam, adres, projectnummer, taaknummer materiaalnummer.*

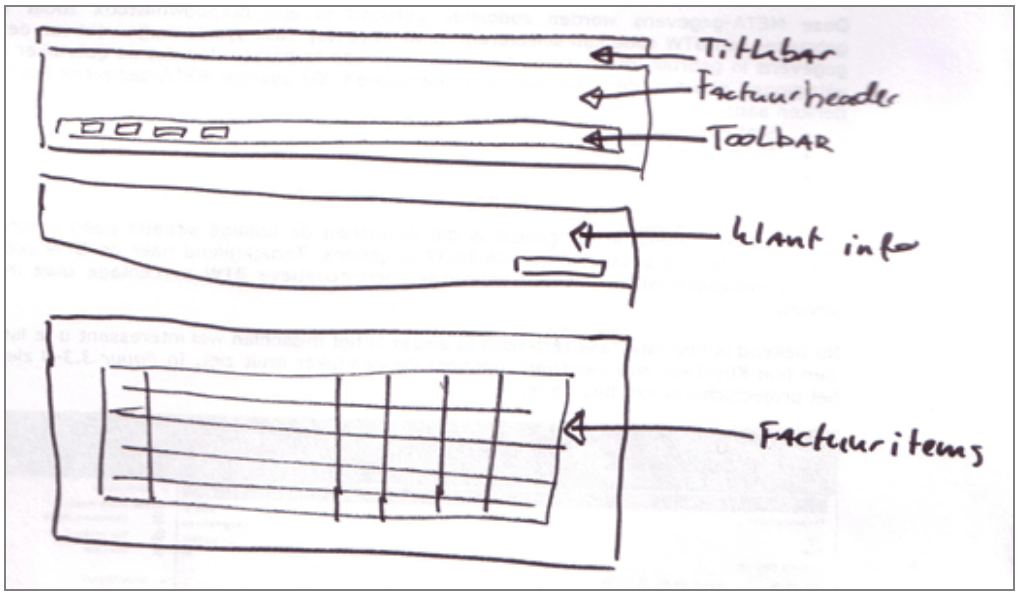
Kijkend naar het navigatieschema kun je tot de conclusie komen dat er, voor wat betreft de pilot 'Facturatie', nieuwe schermen ontwikkeld moeten worden, namelijk:

- Factuur zoekscherm
- Bevestigingsscherm
- Factuurscherm
- Verzendscherm

Ik zal nu van de mogelijkheid gebruik maken te beschrijven hoe ik tot het factuurscherm ben gekomen.

Het factuurscherm wordt voorzien van een flink aantal functionaliteiten. Het moet namelijk niet alleen mogelijk zijn om vanuit het factuurscherm een factuur te bekijken, tevens dien je vanuit dit scherm een factuur te kunnen bewerken. Ik heb ervoor gekozen hiervoor geen apart scherm te ontwerpen, omdat dit naar mijn mening te veel schermen zal opleveren, wat de onderhoudbaarheid van het systeem niet te goede komt.

Vervolgens heb ik mijn beziggehouden met het schermontwerp. Binnen het schermontwerp leg ik de hele gebruikersinterface van deze pilot vast. Ik ben begonnen met het maken van schetsen. Deze schetsen hebben tot doel de schermindeling vast te stellen. De schets, afgebeeld in *figuur 4.4*, laat een voorbeeld zien van het factuurscherm.



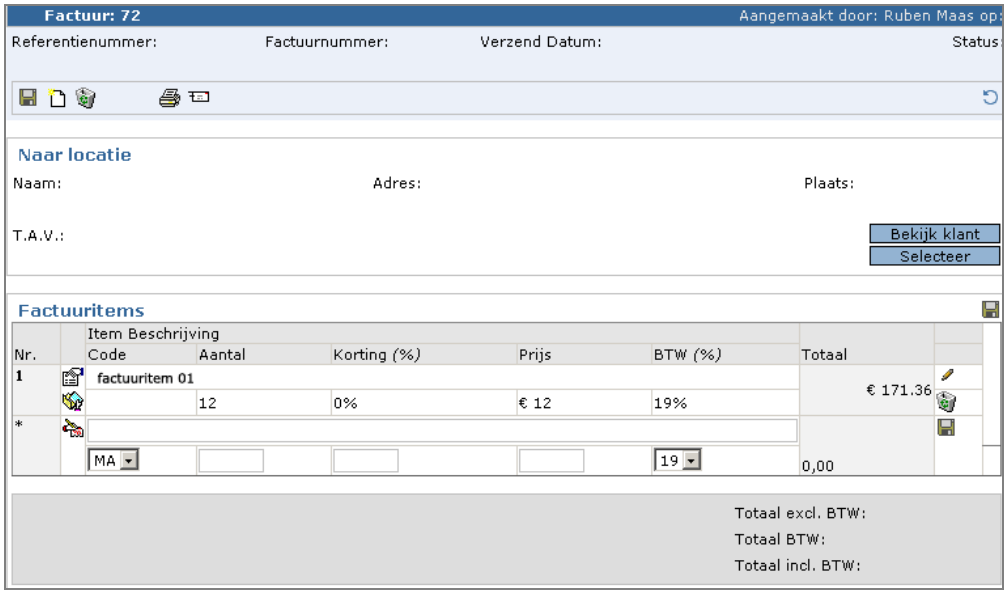
Figuur 4.4 Schets factuurscherm

Het scherm is opgedeeld in de volgende delen:

- Titlebar — Geeft inzicht in welk scherm de gebruiker zich bevindt.
- Factuurheader — Geeft de factuurheader informatie weer.
- Klant info — Laat de benodigde klant gegevens zien.
- Factuuritems — Geeft de betreffende factuuritems van de factuur weer.

In het factuurheader gedeelte bevindt zich tevens de toolbar. Vanuit de toolbar heeft de gebruiker de mogelijkheid acties uit te voeren op de betreffende factuur.

Nu de schermindeling zo goed als vast ligt, richt ik mij op de verdere invulling van het factuurscherm. U kunt hierbij denken aan: Welke kleuren, lettertypes, lettergroottes ga ik gebruiken. Key-Task beschikt over een eigen stijl die staat gedefinieerd in een stylesheet. Zoals in de definitiestudie staat beschreven moeten de te ontwikkelen modules beschikken over dezelfde stijl als het Key-Task systeem. De invulling van de Key-Task stijl in de schetsen bereik ik aan de hand van statische HTML-pagina's. Op basis van de schets maak ik een HTML-pagina om zodoende de stijl eraan te koppelen. In *figuur 4.5* ziet u een voorbeeld van de statische pagina 'Factuurscherm'.



Figuur 4.5 Digitaal schermontwerp factuurscherm

Een belangrijk onderdeel van het scherm is de toolbar. De toolbar staat geplaatst in het bovenste kader en bevat diverse iconen. De iconen worden gebruikt voor het aanbieden van functionaliteit van het systeem aan de gebruiker. Het aanbieden van de functionaliteit kan op diversen manieren, zoals doormiddel van beeld(*iconen/metaforen*), tekst of de combinatie van de twee eerder genoemde. Mijn voorkeur gaat hierbij uit naar het gebruik van de icoon, omdat het gebruik van tekst de schermen onoverzichtelijk maakt. Daarbij voorziet beeld in het principe dat de gebruiker vaak een directe associatie heeft met het beeld en de taak die hij wil gaan uitvoeren. Het hanteren van iconen, visuele representaties van functionaliteit, brengt daarentegen één probleem voor mij als ontwikkelaar met zich mee. Het is lastig om de iconen te kiezen zonder daarbij de consistentie te beschadigen. Hiermee bedoel ik dat je er voor moet zorgen dat een bepaald icoon enkel gebruikt wordt voor een bepaalde taak, waarmee voorkomen wordt dat de icoon de verkeerde associaties met zich meebrengt.

Gezien het feit dat ik niet over de vaardigheden beschik voor het ontwerp en maken van dergelijke iconen, ben ik aangewezen op reeds bestaande iconen. Ik heb ervoor gekozen om gebruik te maken van standaard Windows iconen. Een groot voordeel hiervan is dat deze iconen bij vrijwel iedere gebruiker, die bekend is met een besturingssysteem van Microsoft, de juiste associaties met zich meebrengt.

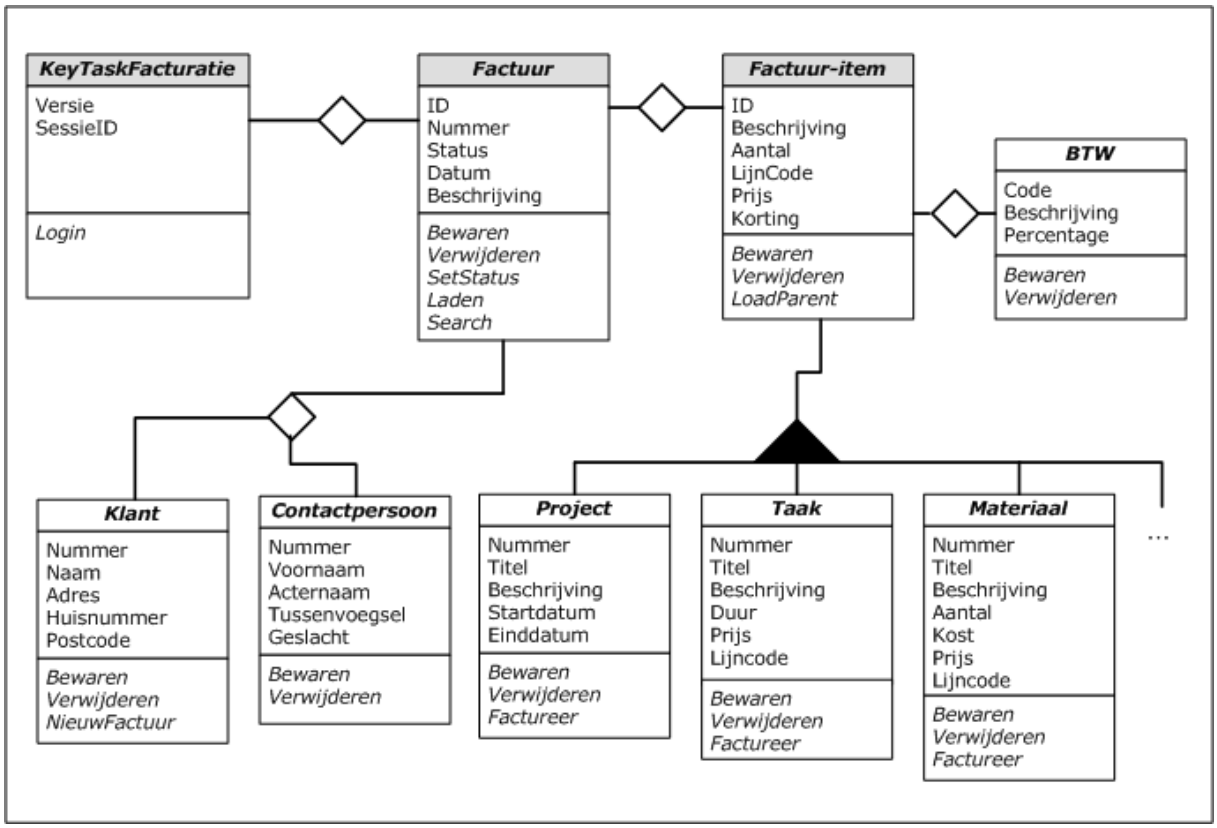
De kleuren die binnen het scherm zijn gebruikt zijn afkomstig uit de huidige stijl van Key-Task. Het hanteren van deze stijl is zeer eenvoudig. Aan de hand van een zogenaamde stylesheet (*css-bestand*), die gekoppeld is aan de HTML-pagina, kan de stijl zodoende worden geïmplementeerd in de te ontwikkelen schermen.

4.1.5 Ontwerp businesslaag

De businesslaag bestaat uit een aantal objecten die de functionaliteit van de module vertegenwoordigen. Het vastleggen van deze objecten doe ik aan de hand van OMT (*Object Modelling Technique*). Het voordeel van OMT ten opzichte van andere object modelleringtechnieken moet ik u hellaas verschuldigd blijven. OMT is een techniek die ik al gedurende jaren hanteer en het bevalt mij prima, mede doordat het binnen elk project wel te hanteren valt. In *figuur 4.6* ziet u het OMT model die ik heb toegepast op de businesslaag van de pilot ‘Facturatie’.

Zoals in de technische-architectuur staat beschreven wordt de module ‘Facturatie’ in een aparte klassenbibliotheek opgeleverd. Het probleem waarop ik stuitte was, dat zowel ik als de opdrachtgever het inloggen vergeten waren bij het definiëren van de systeemeisen. Daar waar het inloggen uiterst noodzakelijk is om toegang te krijgen tot de Key-Task database. Gelukkig is er een aparte klassenbibliotheek die voorziet in de functionaliteit van het inloggen. Daarmee wordt voorkomen dat ik de inlog procedures opnieuw moet maken. Echter zal deze koppeling meegenomen moeten worden in het ontwerp van de businesslaag.

Het aanspreekpunt van de klassenbibliotheek wordt gevormd door het KeyTaskFacturatie-object. Vanuit dit object kan er worden ingelogd en de facturen benaderd worden. De inlogprocedure geeft een sessieobject terug. Het sessieobject beschikt over de gebruikersnaam en wachtwoord, waarmee ingelogd kan worden op de Key-Task database. Bij elke database transactie die binnen de klassenbibliotheek wordt uitgevoerd, wordt er gebruik gemaakt van het de betreffende inlog gegevens. Ik heb hiervoor gekozen, omdat de klassenbibliotheek zodoende geheel onafhankelijk van het huidige Key-Task systeem kan functioneren.

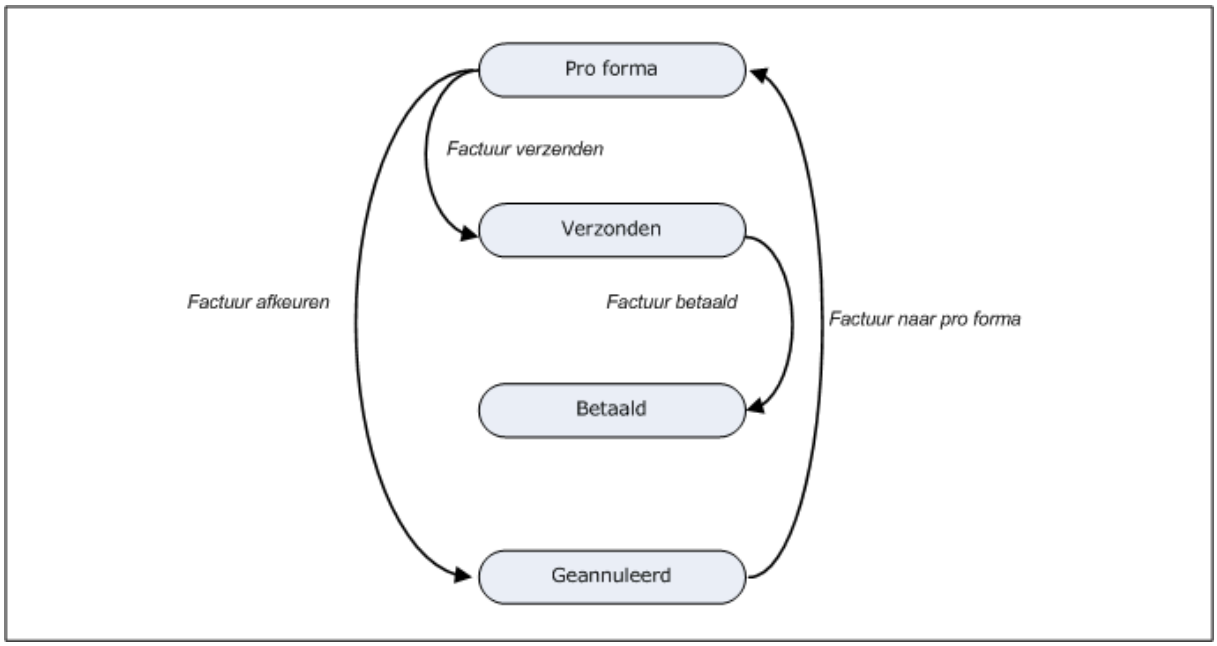


Figuur 4.6 OMT Businesslaag pilot 'Facturatie'

Het OMT model laat drie nieuwe objecten zien namelijk *KeyTaskFacturatie*, *Factuur en Factuuritem*. De overige objecten zijn afkomstig uit de huidige Key-Task klassenbibliotheek en hoeven in dit geval niet ontwikkeld te worden. Een Factuur bestaat uit één of meerdere factuuritems. Een factuur bestaat uit zowel een factuurnummer als een factuurid, oorspronkelijk was mijn idee om een factuur te laten beschikken over enkel het id-attribuut dat op haar beurt het factuurnummer representeerde. Dit attribuut is een numeriek veld en wordt automatisch gecreëerd door het DBMS op het moment dat er een nieuw record wordt weggeschreven. Een belangrijke misvatting van mij was, dat indien een factuur zich in de pro forma stadium verkeerde deze nog niet over een factuurnummer mocht beschikken, omdat de factuur nog niet is erkend door de organisatie. Het factuurid en factuurnummer moesten dus twee aparte eigenschappen worden, waarop de gebruiker geen invloed op mag uitoefenen.
Een factuur kan worden geladen doormiddel van het toekennen van het attribuut 'Id' en daarna de procedure 'Laden' uit te voeren. Aan de hand van het betreffende attribuut kan het factuurobject worden geladen.

Het object factuuritem bestaat uit een *project, taak, materiaal*, of geen object. Deze objecten worden ook wel aangeduid als het child-object van factuuritem. Wat wil zeggen dat er een relatie bestaat tussen de kosten die zijn gemaakt in het kader van een project en het factuuritem. Tevens bestaat het factuuritem uit een geldige BTW -en lijncode.

Aangezien het factuurobject zich in meerdere stadia kan verkeren, zoals *pro forma, verzonden, betaald of afgekeurd*. Heb ik aan de hand van een toestandsdiagram de cyclus van het factuurobject weten vast te leggen. Het hanteren van toestandsdiagrammen vind ik uiterst geschikt voor dergelijke situaties.



4.7 Toestandsdiagram factuur object

Het toestandsdiagram laat zien dat een factuur zich in vier fases kan bevinden. Een factuur start altijd in de fase pro forma. Vanuit deze fase kan de factuur verzonden of geannuleerd worden. Wanneer een factuur verzonden is dan kan deze niet meer terug naar de pro forma status. Dit heb ik gedaan doordat anders de consistentie van de factuurnummers in gevaar komt. Een geannuleerde factuur kan daarentegen wel weer naar een pro forma status overhevelt worden.

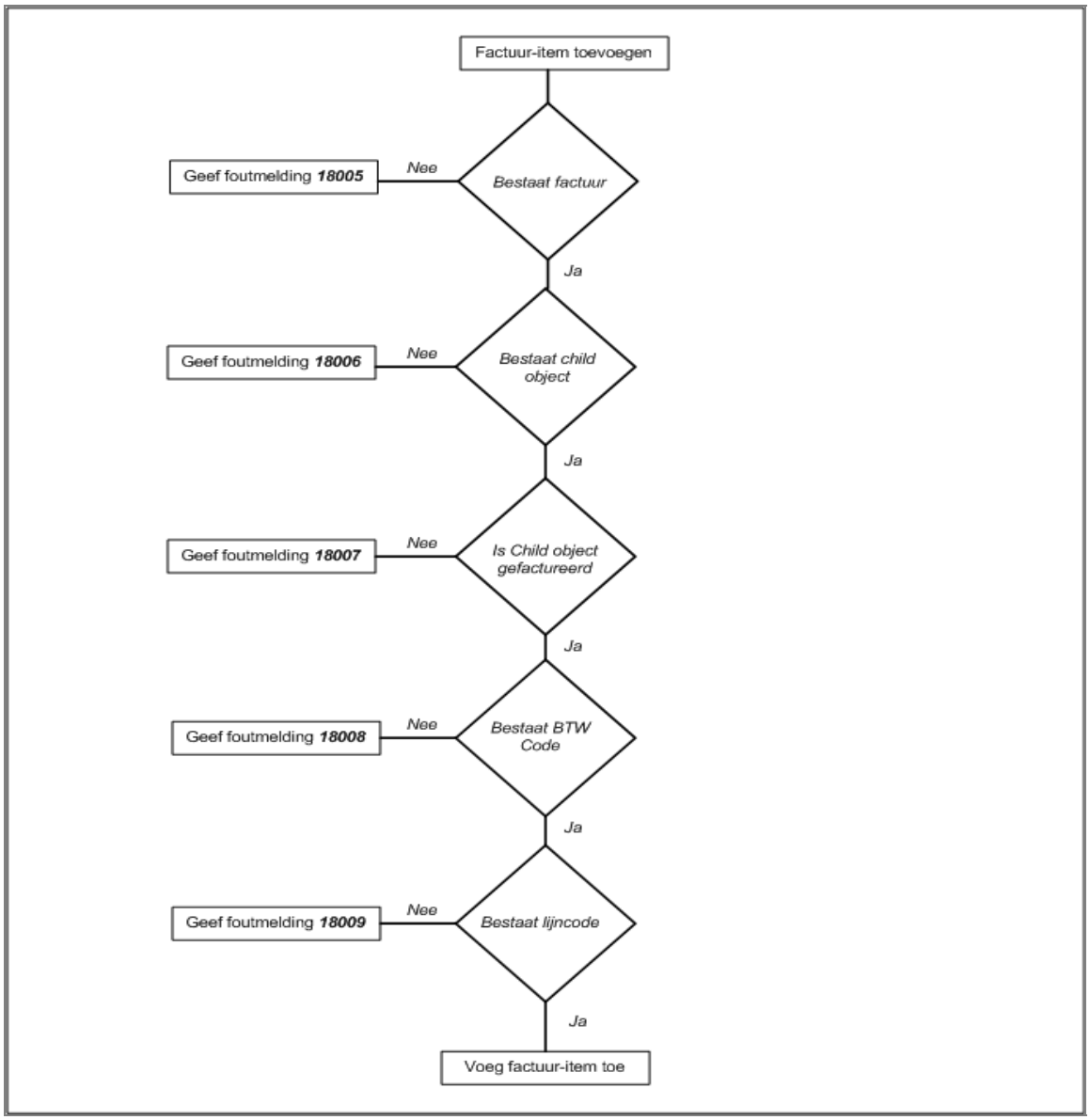
Het daadwerkelijk hanteren van het toestandsdiagram binnen de facturatiemodule wordt bewerkstelligd door de procedure ‘setStatus’.

4.1.6 Ontwerp datalaag

De datalaag heeft als doel de integriteit van de databasegegevens te bewaken. Dat wil zeggen dat de datalaag moet voorkomen dat er onjuiste gegevens in de database staan opgeslagen. Deze onjuiste gegevens hebben betrekking op ongeldige relaties en onjuist formaat.
Ongeldige relaties kunnen ontstaan wanneer er gegevens worden verwijderd die in relatie staan met andere records in de database. Een onjuist formaat ontstaat wanneer er gegevens worden weggeschreven die niet voldoen aan een bepaald formaat, bijvoorbeeld: Een telefoonnummer moet bestaan uit (*landcode)-(regio)-(nummer)*.

Deze integriteitbewaking wordt gedaan door procedures die zich voordoen in modules. Ik heb ervoor gekozen deze procedures te laten voorkomen in modules en niet in zogenaamde objecten (klassenmodules). De reden hiervoor is dat deze procedures niet uit meerdere instanties hoeven te bestaan.

Deze procedures nemen elk een taak voor hu rekening. Voorbeelden van mogelijke taken zijn*: toevoegen factuur, verwijderen factuur, toevoegen factuuritem* etc…



Figuur 4.8 Beslissingsmodel Factuuritem laden

Deze activiteit heeft dan ook als doel het in kaart brengen van de systeemtaken, voor wat betreft de pilot 'Facturatie' en per procedure te bepalen welke controles er per taak dienen te worden uitgevoerd.

De procedures uit de datalaag heb ik ontworpen aan de hand van beslissingsmodellen. Een beslissingsmodel laat de keuzes en acties van de betreffende procedure zien. Bij het ontwerpen van dit soort modules vind ik het erg prettig om te werken met een dergelijk model. Gezien mijn ervaring met deze werkwijze heb ik dan ook niet gezocht naar alternatieven voor deze techniek.

Figuur 4.8 laat ter illustratie het beslissingsmodel voor de procedure factuuritem laden zien. Deze taak koppelt het factuuritem aan een taak, materiaal of project. Om dit te bewerkstelligen dienen er allerlei controles te worden uitgevoerd om de gegevens van het systeem consistent te houden.

De procedure bestaat zoals u ziet uit tal van keuzes, zoals bestaat factuur, bestaat child object, etc… Afhankelijk van het resultaat wordt overgegaan tot de volgende actie. Zo wordt er gecontroleerd of het zogenaamde child-object(taak, project of materiaal) is gefactureerd. Mocht dit reeds het geval zijn dan dient de procedure de foutmelding 18007 zal oproepen en vervolgens wordt de procedure gestopt. Indien alle controles succesvol zijn verlopen dan kan

de procedure overgaan tot de database transacties. Deze transacties worden bewerkstelligd met behulp van SQL (*Structured Query Language*).

4.1.7 Ontwerp Database

Door de ontwikkeling van deze pilot zal tevens de database van Key-Task uitgebreid moeten worden. Deze uitbreidingen worden binnen dit hoofdstuk besproken.

Nu ik weet welke objecten er worden toegevoegd aan het Key-Task systeem kan ik bepalen welke tabellen er moeten worden toegevoegd aan de database. Aan de hand van een Relationeel Database model worden de uitbreidingen va het systeem beschreven.

Factuur (factuur_id, factuur_nummer, factuur_status, factuur_datum, factuur_adres, factuur_contact, factuur_opmerking)
Foreign key factuur_adres references "adres_id" in adres. Nulls not allowed
Foreign key factuur_contact references "contact_id" in contact. Nulls not allowed

Factuuritem (factuur_item_id, factuur_item_factuur, factuur_item_prijs, factuur_item_aantal, factuur_item_korting, factuur_item_btw, factuur_item_beschrijving factuur_item_lijn_code, factuur_item_parent_id, factuur_item_parent_type)
Foreign factuur_item_factuur references "factuur_id" in factuur. Nulls not allowed
Foreign factuur_item_btw references "btw_code" in btw. Nulls not allowed
Foreign factuur_item_lijn_code references "lijncode_code" in lijncode. Nulls not allowed

Factuurstatus (factuur_status_code, factuur_status_beschrijving)

Uit het relationele databasemodel valt te concluderen dat de Key-Task database uitgebreid wordt met een drietal nieuwe tabellen, namelijk factuur, factuuritem en factuurstatus. Opmerkelijk aan de naamgeving van de tabellen is dat elk veldnaam een unieke naam heeft. Dit heb ik gedaan om eventuele aliassen binnen de SQL-queries te vermijden. Ik heb hiervoor besloten omdat dit naar mijn inziens de SQL-queries overzichtelijker maakt.

4.1.8 Aanpassingen externe componenten

De ontwikkeling van de pilot 'Facturatie' staat niet geheel op zichzelf. Dat wil zeggen dat er een aantal aanpassing verricht moeten worden op een aantal externe componenten. Ik spreek hier echter over kleine aanpassingen, zoals: *knoppen toevoegen in bestaande ASP pagina's of eigenschappen toevoegen aan bestaande objecten*. Doch deze aanpassing kleinschalig zijn is het van belang deze te documenteren en mee te nemen in de planning.

Binnen deze activiteit heb ik dan ook van de gelegenheid gebruik gemaakt alle aanpassingen te beschrijven zodat ik deze kan uitvoeren in een daarvoor aparte bouweenheid.

4.1.9 Bouweenheden

De laatste activiteit van het pilotplan is het beschrijven van de bouweenheden. Het hanteren van bouweenheden is noodzakelijk om zodoende de bouwfase te kunnen plannen. Allereerst heb ik mij beziggehouden met het onderscheiden van de diverse bouweenheden waarna ik deze ga inplannen doormiddel van time-boxing. Time-boxing is een projectmanagementtechniek om een iteratieve ontwikkelcyclus te beheersen. Ik heb gekozen voor deze techniek omdat de techniek mij in staat stelt de planning af te richten op datgene waar de meeste prioriteit ligt. Gebruikmakend van deze techniek wordt grotendeels voorkomen dat essentiële systeemonderdelen over onvoldoende functionaliteit beschikken. Daarvoor dien ik allereerst de bouweenheden te beschouwen en te prioriteren. Dit heeft geresulteerd in de volgende lijst van bouweenheden:

Nummer	Naam	Code	Prioriteit
1.	Realisatie datalaag	BE01	Hoog
2.	Realisatie businesslaag	BE02	Hoog
3.	Realisatie factuurscherm	BE03	Hoog
4.	Realisatie zoekscherm	BE04	Gemiddeld
5.	Realisatie bevestigingsscherm	BE05	Gemiddeld
6.	Realisatie verzendscherm	BE06	Laag
7.	Realisatie aanpassingen externe componenten	BE07	Laag

De volgorde waarop de bouweenheden in de bovenstaande tabel staan vermeld worden in dezelfde volgorde gerealiseerd. Ik was grotendeels genoodzaakt deze volgorde aan te houden omdat er diverse afhankelijkheden binnen de technische-architectuur zitten. Een businesslaag kan namelijk niet functioneren zonder dat de datalaag goed functioneert. Hetzelfde geldt namelijk ook voor de presentatielaag. Verder heeft de prioriteit ook een doorslag gegeven. Dit betekent dat een bouweenheid met een hoge prioriteit altijd eerder wordt uitgevoerd dan een bouweenheid met een lagere prioriteit.

De prioriteit samenstelling is gemaakt op basis van de prioritering van de systeemeisen, beschreven in hoofdstuk 6 van de definitiestudie, en de plaats van de bouweenheid binnen de technische-architectuur. Doordat een de hele presentatielaag afhankelijk is van zowel de datalaag als de businesslaag heb ik zodoende beide bouweenheden een hoge prioriteit meegegeven. Nu de bouweenheden vastliggen kan ik beginnen met het inplannen van deze bouweenheden. Zoals ik eerder beschreven heb wordt hiervoor time-boxing gebruikt. De bouwfase bestaat uit vijf weken, dus 25 werkdagen. De eerste test heb ik gepland in het begin van de vierde week om zodoende voldoende tijd te hebben om een mogelijke iteratie te kunnen uitvoeren. Dit betekent dat de bouweenheden ingepland moeten worden over 15 werkdagen.

Code	Geschatte tijd	Werkelijke tijd
BE01	3 dagen	4 dagen
BE02	3 dagen	3,5 dag
BE03	2 dagen	2 dagen
BE04	3 dagen	3 dagen
BE05	2 dagen	2 dagen
BE06	2 dagen	1 dag
BE07	1 dag	0,5 dag

Doordat ik uitloop had op o.a bouweenheid BE01 en BE02 heb ik tijd moeten inleveren voor de realisatie van een bouweenheid met een lage prioriteit. Dit heeft geresulteerd in dat de bouweenheid BE06 niet voldoende functioneerde. Overigens betekent dit niet dat de bouweenheid BE06 onvolledig wordt opgeleverd want eventuele aanpassing aan deze pilot kunnen plaatsvinden in een volgende iteratie. Ik heb mij in dit geval strak aan de planning gehouden om zodoende voldoende tijd over te hebben voor een gebruikerstest. Het hanteren van een gebruikerstest heeft voor mij persoonlijk een zeer hoge prioriteit omdat mijn ervaring is dat er, bij het automatiseren van dergelijke bedrijfsprocessen, altijd calamiteiten naar voren komen binnen de eerste gebruikerstest.

4.2 Realisatie

Nu de pilot ‘Facturatie’ in zijn geheel ontworpen is, wordt er vervolgens overgegaan op de realisatie oftewel het bouwen van de pilot. Voor mij als ontwikkelaar is het nu geheel duidelijk wat er ontwikkeld moet worden en wanneer dit moet plaatsvinden. Binnen dit hoofdstuk zal dan ook worden ingegaan op de problemen die ik tegen kwam en de gemaakte keuzes tijdens het realiseren van de pilot ‘Facturatie’.

Het bouwen van de data –en businesslaag verliep voorspoedig. De procedures heb ik één voor één gebouwd. Zoals ik reeds in het pilotplan heb beschreven wordt elke procedure uit de data –of businesslaag voor oplevering onderworpen aan een UNIT-test. Aan de hand van deze test

kan bepaald worden of het resultaat van de procedure correct is. Tevens wordt er bij een UNIT-test gelet op de tijd die de procedure in beslag neemt, ook wel performance genoemd.

Tijdens het testen van een procedure uit de businesslaag stuitte ik op een performance probleem. Wanneer er een factuur werd opgevraagd dan nam dit teveel tijd in beslag. Zoals in de systeemeisen stond beschreven mocht het ophalen van een factuur niet langer dan 1 seconden in beslag nemen. Het ophalen van een factuur nam daarentegen 1800 milliseconden(*1,8 seconden*) in beslag. Ik moest dus opzoek naar een manier om de performance te verbeteren.

Wanneer het systeem een factuur opvraagt, dan wordt daarmee bedoeld dat het systeem een database record omzet naar in dit geval een factuurobject. Dat wil zeggen dat de waarden van de databaseattributen worden geladen in de eigenschappen van het object. Ik onderscheid echter twee soorten eigenschappen, namelijk eigenschappen met een datatype als waarde en eigenschappen met objecten als waarden.

Ik heb vervolgens navraag gedaan bij diversen Visual Basic forums om te kunnen bepalen waardoor het probleem zich voordoet en nog belangrijker, hoe kan ik dat oplossen. Ik kwam daarbij tot de volgende conclusies:

- Performance problemen kunnen voorkomen indien er slordig wordt gecodeerd. Wanneer er slordig wordt gecodeerd dan wil dat zeggen dat code onoverzichtelijk overkomt op de programmeur. Dit kan tot gevolg hebben dat er bepaalde foutjes over het hoofd worden gezien. Er zijn speciale richtlijnen beschikbaar die de code voor de programmeur overzichtelijk maakt. Een van de bekendste richtlijnen is de Hungarian Notation.
- Het gebruik van speciale statements kan de code sneller doen laten verlopen. Een voorbeeld van zo'n statement is het With-Statement.
- Het gebruik van loops, waarbij een stuk code een X-aantal keren wordt herhaald moet zoveel mogelijk worden vermeden.

Aan de hand van deze bevindingen heb ik allereerst mijn code aangepast zoals de Hungarian Notation voorschrijft. Ik heb voor de Hungarian Notatian gekozen om het feit dat dit de meest gehanteerde richtlijn is. Het aanpassen van mijn code verliep vervolgens zeer vlot en ik heb daarbij dus geen noemenswaardige achterstand opgelopen. Het toepassen van de Hungarian Notation had geen effect voor wat betreft de performance. Dit wil niet zeggen dat het hanteren van deze richtlijn zinloos is geweest. De code is niet alleen voor mij, maar ook voor andere programmeurs nu een stuk duidelijker. Mogelijke toekomstige aanpassingen kunnen nu beter worden doorgevoerd.

Nu de code overzichtelijker is heb ik het geanalyseerd op onnodige loops. Ik ben er toen achter gekomen dat het probleem zich voordoet doordat eigenschappen, van het type object, direct worden geladen bij het opvragen van het factuurobject. Ik zal een voorbeeld geven om dit te verduidelijken.

Een factuurobject bestaat op zijn beurt weer uit een adresobject. Indien een factuur object wordt aangevraagd dan wordt niet alleen het factuurobject geladen maar ook het adresobject.

Ik heb hierdoor geconcludeerd dat het performance probleem wordt veroorzaakt doordat gerelateerde objecten niet op het juiste moment worden geladen. De enige manier om het performance probleem te verhelpen is het laden van de gerelateerde objecten, zoals *adres en contact*, achterwegen te laten op het moment dat het factuurobject wordt aangevraagd. Dit mag echter niet tot gevolg hebben dat beide gerelateerde objecten niet meer opvraagbaar zijn.

Ik heb daarom gekozen voor de volgende oplossing. De gerelateerde objecten worden pas geladen op het moment dat er daadwerkelijk naar gevraagd wordt. Na het doorvoeren van deze aanpassing is uit de UNIT-test gebleken dat de tijd die nodig is voor het opvragen van het factuurobject is gereduceerd tot maar slechts 210 milliseconden (*0,2 seconden*), waarbij deze taak voldoet aan de gestelde eisen.

De bovenstaande aanpassing had gevolgen voor de rest van de realisatie. Bij de bouw van de overige procedures is lering genomen van het reeds eerder ontstaande probleem. Van

dergelijke problemen waren dan ook in het vervolg geen spraken. Dat wil niet zeggen dat elke procedure vlekkeloos uit de UNIT-test is gekomen, maar er hebben zich geen noemenswaardige problemen voorgedaan.

Op het moment dat de data –en businesslaag klaar waren ben ik mij bezig gaan houden met het realiseren van de schermen. Ik heb allereerst van elk scherm een statische versie gemaakt. Dat wil zeggen dat er binnen dat scherm nog geen gebruik gemaakt wordt van een server-side script zoals ASP. Pas hierna ben ik deze pagina's met behulp van ASP gaan koppelen aan de Key-Task Facturatie Klassenbibliotheek. Persoonlijk vind ik deze manier van werken erg prettig doordat ik geen twee dingen, het koppelen business –en presentatielaag en schermontwerp, tegelijk wil doen. Doordat ik zonder gebruikersinterface geen koppeling kan maken met de businesslaag, is de enige mogelijkheid om allereerst een statische versie van het scherm op te leveren alvorens deze gekoppeld wordt aan de businesslaag.

4.3 Testen

Nu het bouwen van de pilot erop zit dient tenslotte de pilot te worden onderworpen aan een gebruikerstest. Deze gebruikerstest is noodzakelijk om te bepalen of er wel of geen iteratie volgt op de pilot 'Facturatie'. De gebruikerstest die ik hanteer is de zogenaamde Use-Cases. Dit zijn een aantal scenario's die de gebruikers moeten uitvoeren.

4.3.1 Aanpak

De aanpak voor wat betreft het testen staat beschreven in het testrapport 'Facturatie'. Hierin ben ik ingegaan op het beschrijven van de testprocedures. De volgende onderwerpen zijn in het testrapport ter sprake gekomen.

- Testpersonen
  Welke personen gaan de test uitvoeren. In dit geval waren dat de medewerkers van Darumath.

- Werkwijze
  Welke stappen worden er genomen voor het testen van de pilot. Binnen dit gedeelte heb ik onder andere beschreven dat de experts eerst de pilot gaan beoordelen, waarna ze hun bevindingen gaan verzamelen en ordenen. Tenslotte zullen ze hun bevindingen aan mij presenteren.

- Voorwaarden
  Hierin staat beschreven aan welke voorwaarden de testpersonen moeten voldoen. Ik heb hier onder andere aangegeven dat de testpersonen ervaring moeten hebben van systeem -of applicatiebeheer.

Tevens heb ik mij binnen het testrapport beziggehouden met het beschrijven van de testscenario's. Deze scenario's zijn gericht om de functionaliteit van het systeem te testen. De daadwerkelijke eindgebruikers zullen deelnemen in deze test. Om te kunnen bepalen of alle functionaliteiten naar behoren zijn doorgevoerd moet ik ervoor zorgen dat alle functionaliteiten terugkomen in de testscenario's. In de onderstaande tabel ziet u een voorbeeld van een dergelijk scenario.

Scenario 1: Taak factureren (A)	
Beginsituatie:	1. Gebruiker is ingelogd 2. Taakscherm geopend 3. Taak afkomstig van niet all-inclusive project 4. Taak heeft willekeurige status 5. Taak niet gefactureerd
Uitvoer:	1. Gebruiker drukt op de factureer icoon 2. Gebruiker selecteert factuur 3. Gebruiker drukt op bevestigen
Resultaat:	Het factuurscherm wordt geopend met daarin het betreffende item aangemaakt.

De testscenario's zijn vervolgens overhandigd aan de testpersonen. Die de scenario's hebben uitgevoerd om zodoende hun bevindingen aan mij voorteleggen.

4.3.2 Resultaten

De resultaten van de eerste gebruikerstest waren negatief. Dat wil zeggen dat er een nieuwe iteratie plaatsvindt op de pilotontwikkeling van de pilot 'Facturatie'. Deze iteratie is noodzakelijk om correcties te kunnen uitvoeren op de pilot. Wanneer het proces van de pilotontwikkeling opnieuw is doorlopen kan het systeem wederom worden getest. Hiervoor heb ik dezelfde testscenario's gebruikt. Doordat ik het gehele proces nogmaals moest doorlopen heb ik wat achterstand opgelopen ten opzichte van mijn planning. Dit betekent dat ik de start van de pilot 'Beheer' met een aantal dagen moest uitstellen.

Binnen de eerste testsessie verliepen een aantal scenario's niet naar behoren. Het verzenden van een factuur bijvoorbeeld was niet mogelijk doordat het systeem geen factuurnummer aan kon maken. Wanneer een factuur wordt verzonden dan betekent dat de status van de betreffende factuur verandert van 'Pro forma' naar 'Verzonden'. Tevens wordt er op dat moment een factuurnummer gegenereerd.

Het was voor mij als ontwikkelaar een raadsel waarom het systeem op dat punt faalde. Aangezien het in mijn omgeving uitstekend deed. Doordat de functionaliteit van het systeem zich in de business –en datalaag bevindt heb ik de broncode van de klassenbibliotheek nader geanalyseerd. Ik kwam tot de conclusie dat de fout werd veroorzaakt door een programmeerfout van mijn kant. Het herstel van deze fout wordt besproken in het volgende hoofdstuk.

Het doorvoeren van de aanpassingen en herstellen van systeemfouten wordt bewerkstelligd door het uitvoeren van een nieuwe iteratie op deze pilot. Dit betekent dat het gehele proces opnieuw gevolgd en getest moest worden. Ik vond het niet nodig om een nieuwe test hiervoor te maken, daarom heb ik gebruik gemaakt van de bestaande testscenario's.

4.3.3 Aanpassingen

Aan de hand van de resultaten uit de gebruikertest wordt een iteratie gedaan op de pilotontwikkeling van de pilot 'Facturatie'. Deze iteratie heeft tot gevolg dat er aanpassingen worden verricht op zowel het pilotontwikkelplan als de betreffende software. Eén van deze aanpassingen betreft het herstellen van de programmeerfout die tot gevolg heeft dat er geen factuurnummers kunnen worden gegenereerd. Het factuurnummer wordt gegenereerd aan de hand van een SQL-query. Deze query vraagt het hoogste factuurnummer op en telt daar één bij op. Dit werkt uitstekend indien het systeem beschikt over minimaal één factuur met een factuurnummer. Wanneer dit niet het geval is dan geeft de query geen resultaten terug. Het systeem geeft vervolgens een systeemfout.

Om dit probleem aan te pakken dient de procedure, die verantwoordelijk is voor het generen van het factuurnummer te worden aangepast. Deze aanpassing houdt in dat er eerst een check moet worden uitgevoerd die bepaald of het systeem beschikt over factuurnummers. Als dit niet het geval is, dan moet altijd "1" worden teruggegeven.

5 Pilot Beheer

De tweede en laatste pilot van dit project is de pilot ‘Beheer’. Alvorens ik ben begonnen met de ontwikkeling van deze pilot, heb ik allereerst een doelstelling voor deze pilot geformuleerd, deze doelstelling luidt als volgt:

De pilot ‘Beheer’ moet ervoor zorgdragen dat de gebruiker de mogelijkheid heeft systeeminstellingen vanuit het Key-Task systeem te bewerken.

Voordat ik kan beginnen met het bouwen van de pilot moet ik eerst een pilotontwikkelplan opstellen. Hoe ik dit heb aangepakt staat beschreven in *paragraaf 5.1.* In *paragraaf 5.2* staat beschreven hoe de uiteindelijke realisatie van deze pilot is verlopen. Tenslotte zal ik in *paragraaf* 5.3 beschrijven hoe ik het testen van deze pilot heb aangepakt en wat daarvan de resultaten waren.

5.1 Pilotontwikkelplan

Net als de vorige pilot wordt deze pilot voorafgegaan door het maken van een pilotontwikkelplan. In het algemeen lijkt het proces wat ik voor deze pilot heb gevolgd erg op de voorgaande pilot. Toch zijn er een aantal afwijkingen die in *paragraaf 5.1.1* zullen worden beschreven.

5.1.1 Plan van aanpak

De pilot ‘Beheer’ neemt 7 weken in beslag. Ten opzichte van de pilot ‘Facturatie’ is de tijd die ik voor deze pilot heb vrijgemaakt een stuk korter. De pilot moet weliswaar voorzien in een groot aantal systeemeisen, ik ben van mening dat de moeilijkheidsgraad, wat betreft de technische aspecten van deze pilot, van dusdanig niveau is dat ik dit gedurende de ingeplande tijd kan bewerkstelligen.

Pilot 'Beheer'	13	14	15	16	17	18
Stel Plan van aanpak op						
Beschrijven Functionele structuur						
Beschrijven ontwerp presentatielaag						
Beschrijven ontwerp businesslaag						
Beschrijven ontwerp datalaag						
Beschrijven Bouweenheden						
Beschrijven testprocedure						
Beschrijven aanpassingen externe componenten						
Bouw pilot						
Maak handleiding						

Figuur 5.1 Planning pilot ‘Beheer’

Kijkend naar de activiteiten en de volgorde waarop ze worden uitgevoerd dan lijkt deze planning op die van de pilot ‘Facturatie’. Er zijn echter een tweetal activiteiten die hierbij ontbreken. Ten eerste ontbreekt de activiteit interview/workshop. De activiteit ‘Ontwerp database’ is ook niet van toepassing, doordat de database structuur al reeds in het huidige Key-Task systeem is vastgelegd.

Zoals gebruikelijk wordt de fase pilotontwikkeling voorafgegaan aan een interview of workshop om zodoende overeenstemming te bereiken over de lijn van het project en om informatie in te winnen omtrent de huidige projectfase. Ik had besloten om een interview met de opdrachtgever toe te passen, omdat hij naar mijn mening de algemeenheid van de module voorop stelde. Ik stuitte echter op een probleem omdat de opdrachtgever 3 weken naar Amerika moest rond de periode dat ik het interview had gepland. Dit was echter lang van tevoren bekend en ik moet eerlijk bekennen dat het mij geheel ontschoten was. Ik heb besloten dit interview te annuleren. Gezien het feit dat ik gedurende mijn afstudeerperiode veel contact had met de opdrachtgever en dat wij dezelfde mening hadden voor wat betreft

de te ontwikkelen pilot. We hebben daarom ook afgesproken dat ik contact met hem zal zoeken middels *E-mail, Telefoon of MSN*. Indien zich problemen of twijfels voordoen.

5.1.2 Functionele Structuur

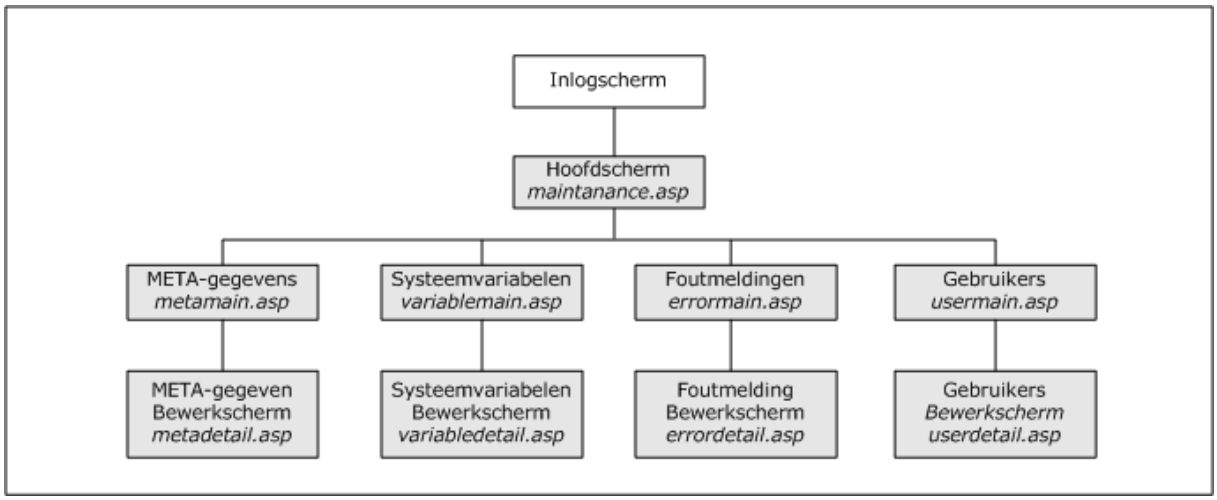
De activiteit 'beschrijf functionele-structuur' heb ik gebruikt om een overzicht te creëren van de functionaliteit, die binnen deze module geboden moet worden.

5.1.3 Ontwerp presentatielaag

Nu voor mij vastligt over welke functionaliteiten de pilot moet beschikken, kan ik beginnen met het ontwerpen van de presentatielaag. Een mogelijk probleem wat hierbij komt kijken is dat de presentatielaag over teveel schermen gaat beschikken. Dit wordt veroorzaakt doordat er veel instanties van een type systeeminstelling zijn. Zo zijn er bijvoorbeeld verschillende instanties van een META-gegeven, zoals: *projectstatus, BTW, lijncode of taakstatus.* Het is dus noodzaak al deze instanties op te vangen in één scherm om zo het aantal schermen te minimaliseren. Doormiddel van het minimaliseren van het aantal schermen wordt onder andere de onderhoudbaarheid vergroot en de structuur overzichtelijker.

Het hanteren van een minimaal aantal schermen moet worden gerealiseerd door de presentatielaag. De uitgangssituatie hierbij moet zijn dat doormiddel van parameters, die worden gepost op de betreffende pagina, bepaald kan worden welke instantie van een bepaalde instelling geladen moet worden. Binnen de bouwfase zal hier meer inhoud aan gegeven worden.

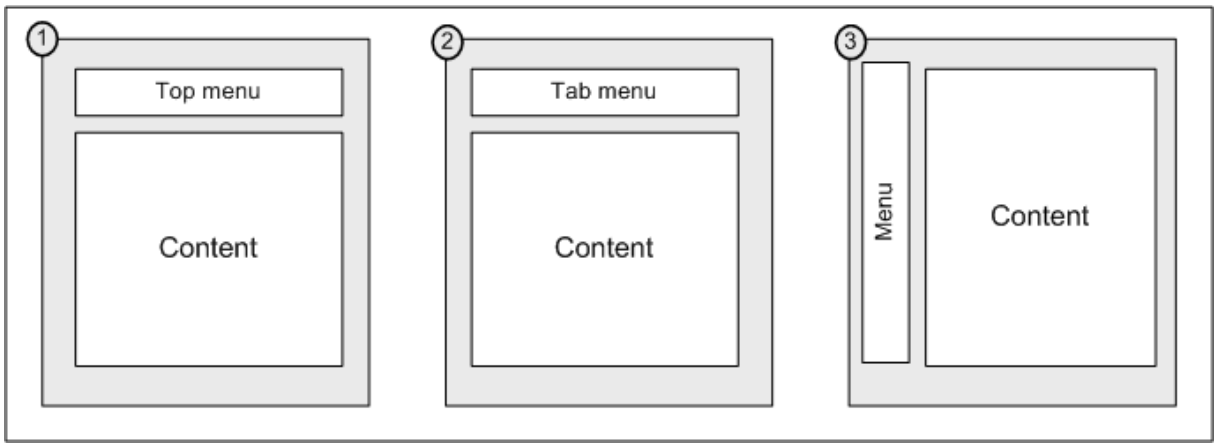
Uitgaande van deze gedachten heb ik het volgende navigatieschema opgesteld.



Figuur 5.2 Navigatieschema pilot 'Beheer'

Zoals het navigatieschema in *figuur 5.2* laat zien dient er eerst te worden ingelogd om zodoende in het hoofdscherm te komen. Vanuit dit scherm kan gekozen worden om een type systeeminstelling te bewerken. Er zijn in totaal vier categorieën, namelijk: *META-gegevens, systeemvariabelen, foutmeldingen* en *gebruikers*. Elke categorie heeft op haar beurt een apart scherm om een gegeven te bewerken.

Een duidelijke en snelle navigatie vind ik een speerpunt binnen deze pilot. Er zijn diverse manieren waarop dit bewerkstelligd kan worden. Ik heb allereerst een aantal manieren geschetst hoe dit bewerkstelligd kan worden. In *figuur 5.3* ziet u hiervan het resultaat.



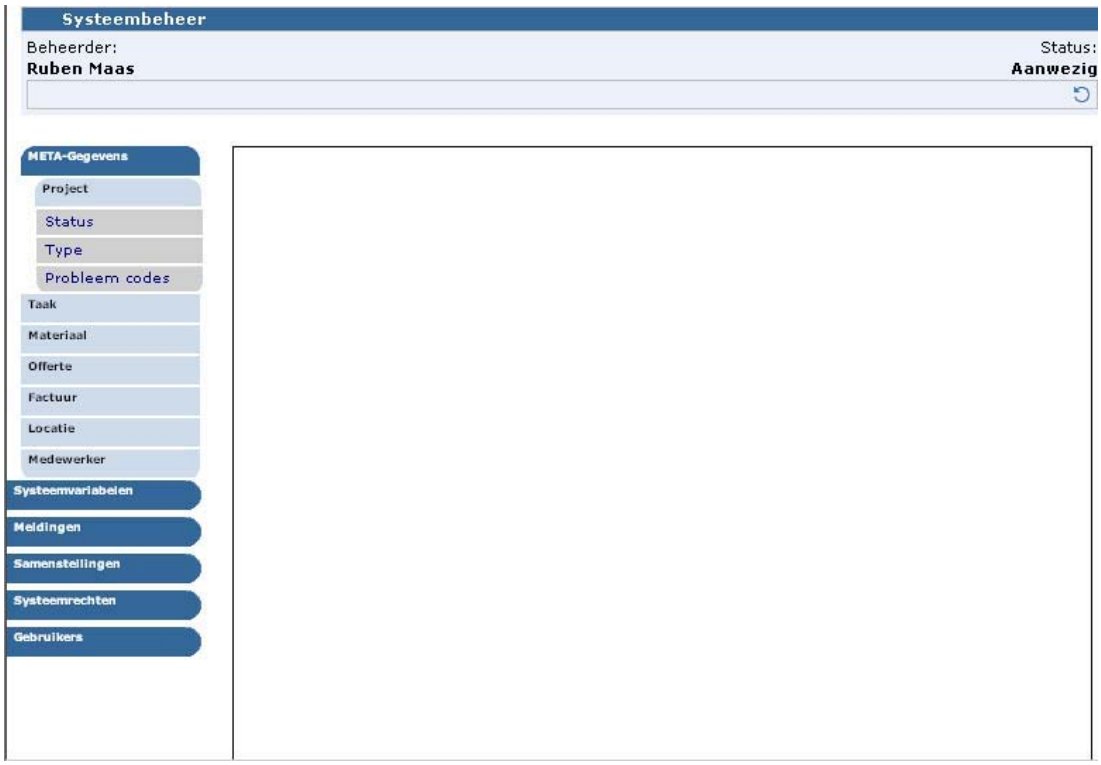
5.3 Schets indeling hoofdscherm

Kijkend naar de bovenstaande schetsen, dan ziet u dat bij elke schets gebruik wordt gemaakt van een menu en een contentscherm. Het idee was een menu altijd zichtbaar te laten gedurende het gebruik van deze module. Alle beheersschermen kunnen zodoende geladen worden in het contentgedeelte. Met het menu moet het mogelijk zijn direct instanties van een instelling te kunnen laden in het contentscherm.

In de drie scheten heb ik het menu op diverse locaties gepositioneerd. Ik heb uiteindelijk ervoor gekozen schets 3 toe te passen binnen mijn ontwerp, omdat een menu aan de bovenkant naar mijn mening onoverzichtelijk lijkt. Dit komt doordat de module wordt geplaatst binnen de shell van Key-Task, die reeds in het bezit is van een menu aan de bovenkant. Bij het toepassen van schets 1 en 2 krijg je naar mijn mening een onoverzichtelijk effect.

Het uitgangspunt voor het hanteren van zo weinig mogelijk schermen is in eerste instantie gericht op de toekomst, waarbij je de aanpasbaarheid en onderhoudbaarheid vergroot. Echter zal de gebruiker ook hiervan de voordelen merken doordat de gebruiker minder schermen hoeft te doorlopen, waardoor hij of zij sneller de systeeminstelling tot zich neemt.

Voordat er verder gegaan kan worden met het maken van een statische HTML-pagina dien ik meer duidelijkheid te krijgen over het menu dat wordt toegepast. Mijn voorkeur ging uit naar een verticaal uitklapbaar menu, doordat deze variant over het algemeen minder problemen met zich mee zal brengen dan een horizontaal uitklapbaar menu. Het overlappen van het contentscherm is hierbij een van de mogelijke problemen. Ik ben vervolgens gaan zoeken naar een gepast verticaal uitklapbaar menu. Ik heb allereerst gekeken bij DHTMLcentral.com. Uit ervaring weet ik dat ze hier heel veel verschillende soorten menu's gratis aanbieden die stuk voor stuk prima werken. Ik heb hier uiteindelijk een menu kunnen vinden waarvan u in *figuur 5.4* het resultaat kunt zien.

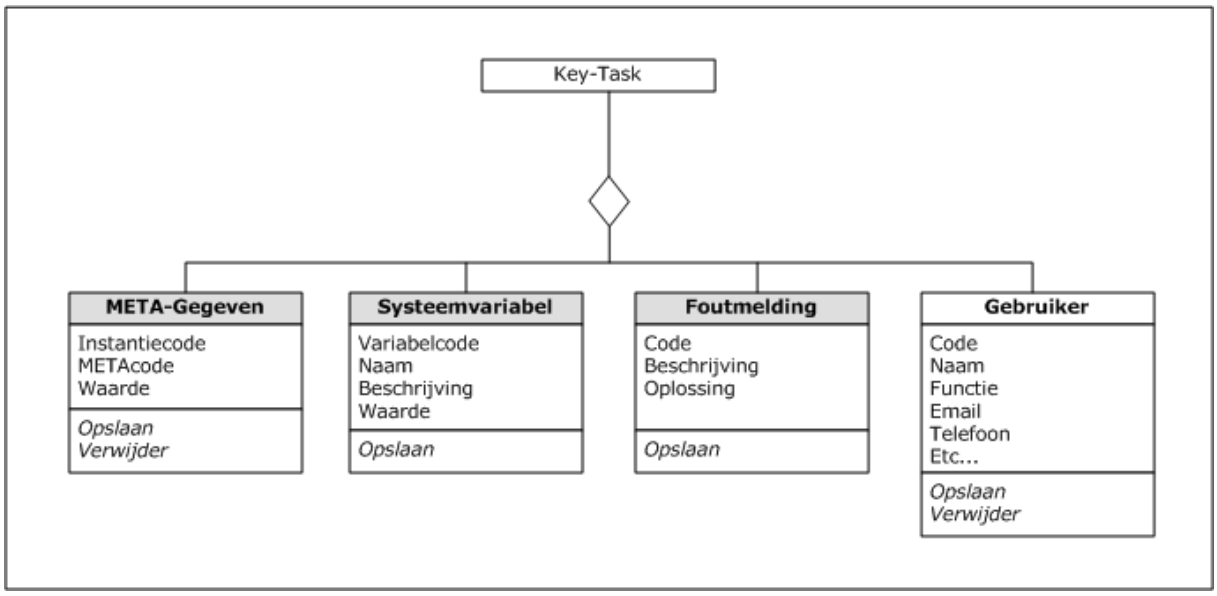


5.4 Digitaal schermontwerp Hoofdscherm

Figuur 5.4 laat het uiteindelijke resultaat zien van de gekozen schets. Aan bovenkant van het scherm is te zien wie er is ingelogd in de module. De toolbar bevat slechts één icoon en dat is het Terug-icoon. Aan de linkerkant bevindt zich het menu. Deze is verticaal uitklapbaar, zoals het voorbeeld laat zien. De kleuren die ik het menu heb meegegeven passen in de algemene stijl van Key-Task. Voor elk sub-menu heb ik een andere kleur gekozen om de menu-items goed van elkaar te laten scheiden. Daarnaast bevindt zich het contentscherm. Het contentscherm bestaat uit een zogenaamd iFrame. Een iFrame is een kader binnen een scherm, dat op elk gewenste positie geplaatst kan worden en waarin internet pagina's geladen kunnen worden. iFrames worden ondersteund vanaf Internet Explorer 5.5, dus vallen ze binnen mijn ontwikkelkader.

5.1.4 Ontwerp businesslaag

De businesslaag bestaat uit een aantal objecten die de functionaliteit van deze pilot vertegenwoordigen. In *figuur 5.5* ziet u een OMT model afgebeeld die de te ontwerpen modellen beschrijft.



Figuur 5.5 OMT businesslaag pilot 'Beheer'

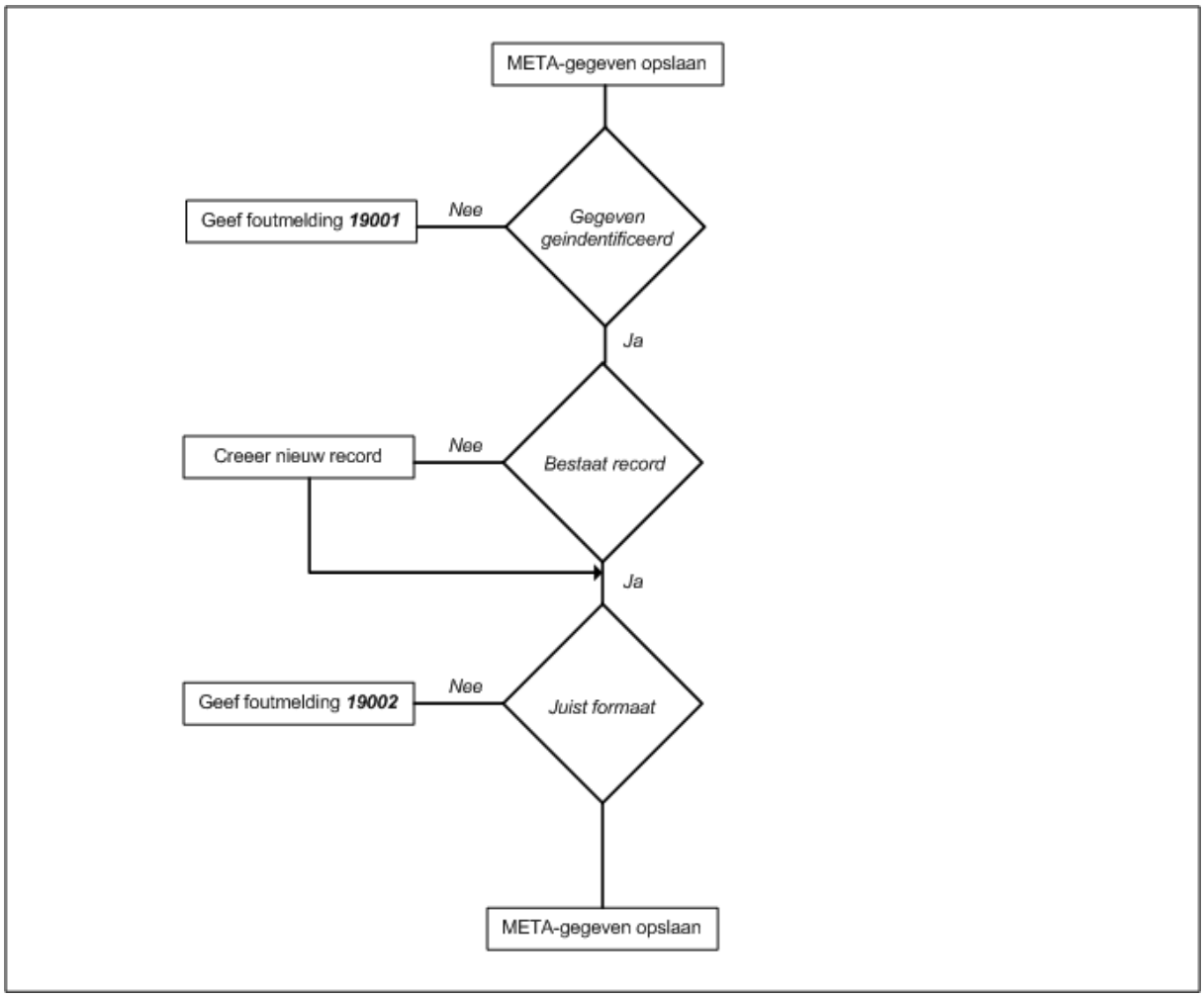
Ik heb de systeeminstellingen gehuisvest in een viertal verschillende objecten:

- META
- Foutmelding
- Gebruiker
- Systeeminstelling

Ik ben tot deze objecten gekomen doordat de betreffende objecten een instantie vormen van een type systeeminstelling. Het gebruik van objecten is voor mij een uitgangspunt, omdat zodoende het de programmacode meer gestructureerd maakt waardoor makkelijker uitbreidingen of aanpassingen op het systeem gemaakt kunnen worden. Indien bijvoorbeeld men spreekt over een instantie van het object 'foutmelding', dan wil dat zeggen dat het betreffende object beschikt over een *code, beschrijving en een oplossing*. Doormiddel van de acties waarover het object beschikt kunnen de eigenschappen van de instantie worden bewerkt.

5.1.5 Ontwerp datalaag

De datalaag van de pilot 'Beheer' moet erop toezien dat alle systeeminstellingen correct worden weggeschreven naar de database. Het is uiterts belangrijk dat de systeeminstellingen in een juist formaat worden bewaard, omdat de gevolgen, indien dit niet gebeurt, niet zijn te overzien. Met een juist formaat bedoel ik dat de variabele(*systeeminstelling*) beschikt over het correcte datatype, grootte of eenheid. De procedures binnen de datalaag hebben de taak om deze bewaking te handhaven. Bij iedere actie vanuit de businesslaag, waarbij er contact is met de Key-Task database, worden er controles uitgevoerd met behulp van zogenaamde functies. Dit zijn stukken programmacodes die een waarde kunnen teruggeven. De waardes die zo'n functie teruggeeft is in dit geval van het datatype boolean, dat wil zeggen dat de betreffende functie waar of onwaar teruggeeft. Zodoende kan ik bepalen of een bepaalde waarde voldoet aan de consistentie en integriteit. Met het ontwerpen van de datalaag heb ik mij dus tevens beziggehouden met het ontwerp van de procedures die de integriteit en de consistentie moet bewaken en waarvan u in *figuur 5.6* een voorbeeld van kunt zien.



Figuur 5.6 Beslissingsmodel META-gegeven opslaan

In de bovenstaande figuur staat een beslissingsmodel afgebeeld die het proces van de procedure META-gegeven opslaan beschrijft. Een beslissingsmodel vind ik, voor het ontwerpen van dergelijke procedures, een zeer handig middel. Met het behulp van het beslissingsmodel wordt het programmeerwerk vereenvoudigd.

Binnen het beslissingsmodel ziet u dat de procedure beschikt over een drietal keuzes. Allereerst moet het META-gegeven geïdentificeerd worden. Dat wil zeggen dat de procedure bepaald om welk META-gegeven het gaat. Dit wordt bewerkstelligd aan de hand van zowel de instantiecode als de META-code. Mocht de procedure het gegeven niet kunnen identificeren dan wordt de procedure onderbroken en er wordt een foutmelding opgeroepen. Indien het META-gegeven is geïdentificeerd dan wordt er bepaald of het META-gegeven bestaat of niet. Wanneer dit niet het geval is dan wordt er een nieuw record hiervoor aangemaakt in de database. Vervolgens wordt er bepaald of het META-gegeven beschikt over het juiste formaat. Dit is afhankelijk van de instantiecode.

5.1.6 Bouweenheden

De laatste activiteit van het pilotontwikkelplan is het beschrijven en inplannen van de bouweenheden. Het beschrijven van bouweenheden is noodzakelijk voor het inplannen van de bouwfase. Binnen deze activiteit sta ik dan ook voor de keuze te bepalen welke bouweenheden ik beschouw en wanneer ik deze daadwerkelijk ga bouwen. Ik ben tot de volgende bouweenheden gekomen:

Nummer	Naam	Code	Prioriteit
1.	Realisatie Datalaag	BE01	Hoog
2.	Realisatie Businesslaas	BE02	Hoog
3.	Realisatie Hoofdscherm	BE03	Hoog
4.	Realisatie META-gegevensschermen	BE04	Hoog
5.	Realisatie Systeemvariabelenschermen	BE05	Gemiddeld
6.	Realisatie Gebruikersschermen	BE06	Gemiddeld
7.	Realisatie Foutmeldingschermen	BE07	Laag
8.	Realisatie Aanpassingen externe componenten	BE08	Laag

Ik ben tot deze acht bouweenheden gekomen aan de hand van de technische-architectuur van de pilot 'Beheer'. Binnen deze technische-architectuur zijn diversen onderdelen al duidelijk zichtbaar en ik heb er dan ook voor gekozen deze delen tevens te hanteren als bouweenheden.

Aan de hand van time-boxing heb ik vervolgens de verschillende bouweenheden ingepland. Ik had drie weken beschikbaar voor de realisatie van deze pilot. Daarbij was mijn streven dat ik de pilot in het begin van de derde week zal gaan testen. Dat betekent dat de eerste iteratie 10 dagen in beslag neemt.

Code	Geschatte tijd	Werkelijke tijd
BE01	2 dagen	2 dagen
BE02	1 dag	1,5 dag
BE03	1 dag	1 dag
BE04	1 dag	1,5 dag
BE05	1 dag	1 dag
BE06	2 dagen	2 dagen
BE07	0,5 dag	0,5 dag
BE08	0,5 dag	0,5 dag

Ik heb een krappe planning kunnen toepassen voor wat betreft de realisatie van de bouweenheden mede door het feit dat ik ingewerkt was. Door mijn ervaringen bij de realisatie ben ik ervan uit gegaan dat de realisatie van de huidige pilot mij minder tijd zal kosten.

5.2 Realisatie

Ik ben de realisatie van de pilot 'Beheer' begonnen met het maken van de data –en businesslaag. Gedurende het bouwen heb ik veel hinder ondervonden aan het feit dat de systeeminstellingen in een database worden bijgehouden. Indien er aanpassingen werden verricht op de database zoals het toevoegen van velden, dan moet deze wijziging worden meegenomen in alle instanties van de Key-Task database. Dit bracht vaak problemen met zich mee. Mogelijke alternatieven voor deze aanpak zijn het gebruik van INI -of XMLbestanden. Het gebruik van dergelijke bestanden maakt het systeem wat mij betreft beter aanpasbaar. Kijkend naar de toekomst is het dan ook vrij eenvoudig instellingen aan het systeem toe te voegen.
Ik heb dan ook in overweging genomen de systeeminstellingen over te hevelen naar XML-bestanden, omdat die voor mij als ontwikkelaar het toegankelijks zijn. Gezien de technische ingrepen die deze wijziging met zich mee zal brengen heb ik ervoor besloten geen wijziging, voor wat betreft de opslag van de systeeminstellingen, door te voeren.

Gedurende de realisatie van zowel de datalaag als de businesslaag heb ik de procedures wederom onderworpen aan een UNIT-test. Deze testen hebben echter geen grote fouten geconstateerd. Ook de performance van de procedures was zeer acceptabel. De goede performance en weinig programmeerfouten zijn naar mijn mening het gevolg van de ingrepen die ik tijdens de realisatie van de pilot 'Facturatie' heb gedaan. Deze ingrepen betroffen het hanteren van de Hungarian Notation en het zoveel mogelijke ontlopen van loops. Doordat ik,

bij de realisatie van de pilot ‘Beheer’, vanaf het begin de Hungarian Notation heb gehanteerd heeft dat geresulteerd in een gestructureerde en overzichtelijke broncode.

Toen de data en businesslaag gebouwd waren kon ik gaan beginnen aan de realisatie van de presentatielaag. Ik ben daarbij begonnen met het maken van het hoofdscherm. Een belangrijk onderdeel van dit scherm is het menu, waaruit de diversen systeeminstellingen direct benaderbaar moeten zijn. Ik heb voor de keuze gestaan om zelf een menu te ontwikkelen of een bestaand menu te gebruiken. Beide argumenten hebben zo hun voor –als nadelen. Wanneer je zelf een menu ontwikkelt kan je hem helemaal naar eigen wensen africhten, maar het levert wel extra ontwikkeltijd op. Gezien het laatste argument heb ik ervoor besloten een bestaand menu te gebruiken, doordat ik al wat krap met de tijd zat. Uiteindelijk heb ik een menu gebruikt van DHTMLcentral.com. Deze site beschikt over allerlei freeware menu’s.

5.3 Testen

Zoals er in het pilotplan staat beschreven wordt deze pilot getest aan de hand van de Heuristic Evaluation testmethode. In *paragraaf 5.3.1* staat beschreven hoe ik dit heb aangepakt. Verder staan in *paragraaf 5.3.2* de resultaten van de test beschreven. Aan de hand van de resultaten kan bepaald worden of er een iteratie volgt op de pilot ‘Beheer’.

5.3.1 Aanpak

Heuristic Evaluation is een testmethode waarbij een groep experts (*2 – 5 personen*) de usability van de pilot gaan testen. Dit doen ze gedurende een periode van 2 tot 4 uur. Achteraf zullen ze hun bevindingen aan mij presenteren.

Ik heb ervoor gekozen alleen de usability te testen omdat de functionaliteit al in een eerder stadium is getest doormiddel van een UNIT-test. Aangezien deze pilot niet beschikt over ingewikkelde processen vind ik het niet zinvol om deze functionaliteit nogmaals te testen. Voorafgaand aan de Heuristic Evaluation test ben ik verzekerd dat de functionaliteit, binnen de klassenbibliotheek, correct is doorgevoerd.

Een aantal zaken die binnen het testrapport ter spraken zijn gekomen zijn onder andere:

Testpersonen:

- Jasper Schut
- Frans Mientjes
- Dimitri Stoikof

Ik heb voor deze personen gekozen, omdat ze naar mijn mening veel kennis hebben over de usability aspecten van software systemen of veel met applicatiebeheer in aanraking zijn gekomen.

Werkwijze:

Welke stappen worden er genomen voor het testen van de pilot. Binnen dit gedeelte heb ik onder andere beschreven dat de expert eerst de pilot gaan beoordelen, waarna ze hun bevindingen gaan verzamelen en ordenen om ze vervolgens aan mij te presenteren.

Voorwaarden:

Hierin staat beschreven aan welke voorwaarden de testpersonen moeten voldoen. Ik heb hier onder andere aangegeven dat de testpersonen ervaring hebben met systeem of applicatiebeheer.

5.3.2 Resultaten

Nadat de testpersonen uitvoerig de pilot hebben beoordeeld hebben zij hun bevindingen aan mij getoond. De dingen die ter spraken zijn gekomen hadden voornamelijk betrekking op de schermindelingen en de gehanteerde iconen. Ik heb de aanmerkingen van de testpersonen erg serieus genomen en zodoende in overweging genomen of een volgende iteratie noodzakelijk is. Een iteratie vond ik niet noodzakelijk, aangezien de pilot verder correct functioneerde. Doordat ik reeds was uitgelopen op de planning vond ik een iteratie niet de moeite waard.

DEEL IV

De Evaluatie

6 Procesevaluatie

Binnen dit hoofdstuk zal worden teruggekeken naar het gevolgde proces. De evaluatie van het proces is bestaat uit een drietal onderdelen, namelijk:

1. Opstartfase en plan van aanpak
2. Definitiestudie
3. Pilotontwikkeling

6.1 Opstartfase en Plan van aanpak

Geruime tijd voor aanvang van dit project heb ik samen met de opdrachtgever een opzet gemaakt van de opdrachtomschrijving met betrekking tot dit project. Hierin werden problemen omtrent Key-Task beschreven en tot een doelstelling geformuleerd. Deze opdrachtomschrijving heb ik vlak voor aanvang grotendeels gewijzigd. Door de groei in vraag naar Key-Task waren zowel ik als de opdrachtgever van mening dat er andere problemen waren met een hogere prioriteit. Ik heb daarom de probleem -en doelstelling van de opdrachtomschrijving gewijzigd. Deze wijziging betrof het schrappen van een rapportage -en vertaalmodule en deze te laten plaatsmaken voor een facturatiemodule. Achteraf sta ik nog steeds achter deze wijziging. De wijziging was gezien de prioriteit noodzakelijk. Tevens vind ik de huidige opdracht leerzaam en uitdagend.

Voor aanvang en tijdens het begin van dit project heb ik zeer veel contact gehad met de opdrachtgever. Deze contacten waren in het begin zakelijk, maar werden al gauw informeel van aard. Dit heb ik als ontwikkelaar zijnde als zeer prettig ervaren. Gaande het project werd het contact met de opdrachtgever minder, doordat ik tot zekere hoogte wist wat mij te doen stond. We hebben echter wel een wekelijks terugkomend gesprekje ingevoerd waarbij we de status van het project bespraken. Aan deze gesprekken heb ik veel gehad. Doordat ik inzicht gaf in bijvoorbeeld de ontwerpen van de te ontwikkelen modules, was hij in staat hier en daar adviezen te geven van technische aard.

Binnen het plan van aanpak heb ik zaken vastgelegd zoals de te volgen ontwikkelmethode en de planning. Gedurende dit project ben ik erachter gekomen dat het zeer moeilijk is af te stappen van iets waarbij je veel ervaring hebt en in het algemeen goed toepasbaar is voor jouw project. Ik doel hierbij met name op de ontwikkelmethode. Zoals u in dit verslag heeft kunnen lezen heb ik IAD gekozen als de te volgen ontwikkelmethode. Naar mijn mening heeft deze methode goed uitgepakt. Dit komt mede doordat je een dergelijke ontwikkelmethode af kunt richten op je project doormiddel van bijvoorbeeld de te volgen ontwikkelstrategie. De strategie incrementeel ontwikkelen heeft dan ook goed uitgepakt doordat de focus van het project lag bij de pilotontwikkeling. Ik vind dit erg prettig, omdat er veel te ontwerpen en realiseren viel en ik zodoende meer tijd hiervoor had gecreëerd. Tijd die ik achteraf gezien niet had kunnen missen. Hierdoor vind ik tevens de ontwikkelmethode en haar strategie zeer geschikt voor het ontwikkelen van verschillende modules.

Ondanks ik tevreden ben over de ontwikkelmethode en hoe het heeft uitgepakt, heb ik er gemengde gevoelens bij. Graag zal ik meer kennis willen hebben van andere ontwikkelmethodes en dit project was er misschien wel een goede mogelijkheid voor geweest. Doordat ik onvoldoende kennis heb in andere ontwikkelmethodes zal ik dus nooit eenduidig kunnen stellen of IAD de beste oplossing zal zijn. Ik heb uiteindelijk toch voor het oude vertrouwde gekozen, doordat ik het gezien de tijdsdruk niet zag zitten een andere ontwikkelmethode tot me te nemen. Als toekomstig doel heb ik dan ook meer kennis te verschaffen over andere ontwikkelmethodes en met deze ervaring op te doen.

De planning die ik uiteindelijk in het plan van aanpak heb opgesteld, vond ik achteraf gezien wat aan de strakke kant. Met name de tijd die uitstond voor de pilot ‘Beheer’. Hierdoor heb ik geen iteratie kunnen uitvoeren op de betreffende pilot. Een nadeel van een dergelijke strakke planning was dat het niet veel ruimte bood voor grootschalige problemen. Noch deze problemen parten hebben gespeeld, is het een groot risico dat op de loer ligt. Rekening houdend met diversen externe factoren, die mijn planning hebben beïnvloed, kan ik wel stellen dat de opdracht uitvoerbaar was in de daarvoor beschikbare tijd.

6.2 Definitiestudie

Ten gevolge van de gekozen ontwikkelstrategie was er weinig tijd voor het maken van de definitiestudie, maar de tijd die ervoor uitstond bleek voldoende te zijn. De meeste tijd ben ik kwijt geweest met het beschrijven van de huidige situatie van zowel de gebruikers als Key-Task. De huidige situatie bleek achteraf een zeer belangrijk aspect van de definitiestudie, doordat dit de grondslag vormde van vele ontwikkelactiviteiten, zoals het beschrijven van de technische-architectuur en het pilotplan.

Een constante factor bij het opstellen van de definitiestudie, waarmee ik rekening mee moest houden, was dat Key-Task een algemeen systeem is wat inzetbaar is in verschillende organisaties. Dit betekent dat ik als ontwikkelaar moet kunnen beoordelen of de wensen van de aangestelde gebruikers(*medewerkers van Darumath*) niet de algemene werking schaadt. Deze beoordeling vond plaats in samenspraak met de opdrachtgever. De mening van de opdrachtgever had ik nodig, omdat ik als ontwikkelaar snel geneigd ben in te spelen op wensen van, in dit geval, de huidige gebruikers. Gaande de loop van de definitiestudie heb ik het algemeen belang van het Key-Task systeem voorop weten te stellen.

Het ontwikkelscenario wat ik heb beschreven tijdens de fase definitiestudie vind ik achteraf gezien overbodig. Binnen het ontwikkelscenario wordt er bepaald welke pilots er worden toegepast. Echter maak ik de keuze voor de pilots in een later stadium. De verdeling van de pilots maak ik namelijk vrijwel altijd op basis van de technische-architectuur. Dit betekent dat ik de pilotindeling pas maak bij het opstellen van het pilotplan. Het opstellen van het ontwikkelscenario vond ik dus overbodig.

Over het opstellen van de systeemeisen ben ik tevreden. IAD wijst erop dat systeemeisen geprioriteerd moeten worden en dat heeft mij wel geholpen. Ik kreeg hierdoor een zeer goed inzicht over in welke volgorde bepaalde activiteiten uitgevoerd moeten worden.

Over het hanteren van taakdiagrammen, die ik heb toegepast bij het beschrijven van zowel de huidige als de gewenste situatie, ben ik zeer tevreden. Taakdiagrammen vind ik persoonlijk een zeer goede techniek die mij in staat stelt een proces snel en goed te leren begrijpen. Afstappen van het hanteren van taakdiagrammen zal voor mij dan ook niet snel van toepassing zijn.

Binnen de technische-architectuur staat beschreven hoe de te ontwikkelen modules zullen worden opgebouwd. Hierin is gekozen voor het hanteren van een 3-lagen model. Het hanteren van 3-lagen modellen is mijn stokpaardje waar ik in het algemeen niet makkelijk van af stap. Een dergelijk model zorgt ervoor dat het systeem schaalbaar, aanpasbaar en benaderbaar is voor externe systemen, wat binnen de huidige IT-wereld een streven is. Tevens biedt het 3-lagen model een uitstekende basis voor de overige ontwikkelactiviteiten. U kunt hierbij denken aan de volgorde van de ontwerpactiviteiten en het definiëren van de bouweenheden. Kortom, het 3-lagen model is een model waar ik niet snel van zal afstappen, doordat de voordelen hiervan de nadelen ver overschrijden.

Het pilotplan lag voor wat betreft de indeling en de prioritering van de pilots naar mijn mening erg voor de hand. Dit kwam doordat de technische-architectuur logisch en duidelijk in elkaar stak. Lastiger was de keuze voor de te hanteren testmethodes. De keuze voor de UNIT-test, om de procedures binnen de data –en businesslaag voor oplevering te testen, was achteraf gezien een succes. De meeste problemen werden vroegtijdig opgespoord en gerepareerd. De keuze welke gebruikerstesten ik ga hanteren bij de twee pilots was zeer lastig. Het aanbod van testmethodes is zo groot dat ik echter ook een andere had kunnen gebruiken.

6.3 Pilotontwikkeling

De ontwikkelmethode IAD beschrijft tal van activiteiten die worden aangeraden. De invulling hiervan ligt uiteindelijk bij de ontwikkelaar zelf. Binnen deze fase heb ik tal van activiteiten aangepast naar eigen visie. De technische-structuur heb ik zodoende onderverdeeld in een drietal onderdelen, namelijk:

1. Ontwerp Presentatielaag
2. Ontwerp Businesslaag
3. Ontwerp Datalaag

De reden hiertoe was dat het uitstekend aansloot op de technische-architectuur beschreven in de definitiestudie. Tevens ben ik erg te spreken over de volgorde waarin deze delen werden ontworpen. Ik vond het prettiger eerst te beginnen met de presentatielaag omdat ik zodoende beter kon bepalen over welke functionaliteit de businesslaag moest beschikken.
Beide pilots hebben mij laten inzien dat toevoegingen op een bestaand systeem, zoals het ontwikkelen van modules, geen eenvoudige klus is. Als ontwikkelaar sta je daarbij niet geheel vrij in de keuzes die je maakt bij het ontwerpen van de software. Ik vond het dan ook jammer dat ik binnen de fase pilotontwikkeling een aantal keuzes niet heb kunnen maken, doordat dit teveel aanpassingen met zich mee bracht. U kunt hierbij denken aan bijvoorbeeld de dataopslag van de systeeminstellingen of de te hanteren ontwikkelomgeving.

Bij de realisatie van beide pilots heb ik gebruik gemaakt van Visual Studio 6. Visual Studio is een pakket van Microsoft en voorziet onder andere ontwikkeltools zoals Visual Basic 6, Visual C++, Visual FoxPro. Visual Studio biedt de gebruiker de mogelijkheid de zogenaamde klassenbibliotheek te ontwikkelen evenals de ASP-pagina's. Voor het realiseren daarentegen heb ik gebruik gemaakt van Notepad, een eenvoudige text-editor. Ik had misschien tijd kunnen besparen om een meer geschikte ontwikkeltool te kiezen voor wat betreft het realiseren van de ASP-pagina's. Visual Studio zou hierbij een goede oplossing zijn geweest.

Binnen deze fase heb ik een aantal fouten gemaakt die gelukkig geen grote gevolgen met zich mee brachten. Het interview wat ik, tijdens de ontwikkeling van de pilot 'Beheer', in gedachten had moest geannuleerd worden, doordat de opdrachtgever niet aanwezig was. In het gevolg moet ik dan ook in een eerder stadium nagaan of de benodigde personen beschikbaar zijn gedurende het project.

Over het testen van de pilots ben ik tevreden. Het is moeilijk om te kunnen beoordelen of een andere testmethode een beter resultaat had opgeleverd. Zeker is dat de testen geresulteerd hebben in het opsporen van gebreken. Tevens vond ik de manier van testen zeer effectief, doordat het mij persoonlijk weinig tijd kosten. Ik weet niet of ik deze wijze van testen in ieder project zal gaan toepassen, omdat het zeer afhankelijk is van het type project.

Over het algemeen ben ik tevreden over het proces dat ik gedurende deze fase heb gevolgd. Door het doorlopen van het proces, die grotendeels door IAD is gekenmerkt, vormde dit voor mij een goede leidraad. Deze leidraad heb ik nodig doordat ik de neiging heb het daadwerkelijke probleem te ontzien met allerlei overbodige functionaliteit.

7 Productevaluatie

Binnen dit hoofdstuk zal ik ingaan op de kwaliteit van de verschillende producten. Dit hoofdstuk is opgedeeld in drie delen. Allereerst worden de opgeleverde documenten behandeld. Hierin staan documenten zoals het plan van aanpak, definitiestudie, pilotontwikkelplannen, testrapporten en handleidingen. Vervolgens worden achtereenvolgens de twee opgeleverde modules onder de loep genomen.

7.1 Opgeleverde documenten

De documenten die in het kader van dit project zijn opgeleverd zijn gemaakt ter ondersteuning van zowel het product als het proces. Elk bedrijf hanteert een bepaalde mate waaraan dergelijke documenten moeten voldoen. Bij Metrix ben ik vrij gelaten in de wijze waarop ik, procesmatig gezien, te werk ging. Ik heb aan de hand van de documenten, die ik heb opgeleverd, de betrokken partijen inzicht kunnen verlenen in de status van het project. Denk daarbij aan het *plan van aanpak, definitiestudie en pilotontwikkelplannen*. Maar uiteindelijk waren de documenten vooral bedoeld ter ondersteuning van mijzelf. Het is voor mij dan ook van groot belang de functionaliteit en de planning goed vast te leggen. Kijkend naar het verleden had ik vaak de neiging het niet strikt te willen nemen met zowel de planning als doelstellingen van het systeem. Dit resulteerde in het feit dat het systeem eigelijk nooit afkomt, doordat ik steeds nieuwe functionaliteiten ging toevoegen. Kijkend naar dit project kan ik dan ook stellen dat de functie van een ontwikkelmethode en de daarbij gerelateerde documenten uiterst van belang zijn.

7.1.1 Plan van aanpak

Het plan van aanpak was het eerste document wat ik heb opgeleverd en het vormde tevens de leidraad van mijn project. Over de inhoud van het plan van aanpak ben ik tevreden. Het belangrijkste uit het plan van aanpak vind ik de planning, want dat is datgene uit het document wat ik gaande het project vaak heb nageslagen. Ik vind wel dat de planning gedetailleerder moest, door bijvoorbeeld ook de feestdagen er beter in te verwerken. Zeker kijkend naar het jaar 2003 waarbij alle feestdagen rond kerst doordeweeks plaatsvonden.

7.1.2 Definitiestudie

Voor de definitiestudie had ik weinig tijd ingepland, maar ik vind dat het document voorziet in haar functie en zodoende de doelen en beperkingen goed weergeeft. Ik heb de definitiestudie voornamelijk geraadpleegd voor het inzien van de systeemeisen en de gewenste situatie. Dat wil niet zeggen dat de rest van de inhoud er niet toe doet. Bij het opstellen van de definitiestudie heb ik daarvan informatie tot mij kunnen nemen, waarvan ik de rest van het project profijt heb gehad.

7.1.3 Pilotontwikkelplannen

Ik heb twee pilotontwikkelplannen opgeleverd. Binnen deze pilotontwikkelplannen staat het ontwerp van de betreffende pilot beschreven. Aan de hand van het document heb ik naar eigenzeggen lastige procedures weten te realiseren, dus in die zin kwam de functie van het document tot zijn recht. Binnen de pilotontwikkeling werden weinig worden vuil gemaakt aan verduidelijkingen van toegepaste modellen. Dit heb ik gedaan omdat ik dat niet nodig vind. Wanneer ik een model maak dan zegt dat model mij genoeg en een verduidelijking is voor mij dus overbodig. Ik kan mij voorstellen indien je in een team te werk gaat iedere persoon hierin zijn eigen kwaliteit heb zodat je het pilotontwikkelplan.

7.1.4 Testrapporten

De testrapporten hadden als doel een beschrijving te geven over de aanpak van de test en de daadwerkelijke testscenario's. Tevens zijn de testrapporten voorzien van de zogehete resultaten. De testrapporten waren naar mijn mening kort en bondig. Het primaire doel van de testrapporten was dat ze mij, de ontwikkelaar, moesten ondersteunen in het uitvoeren van de testprocedures. In die hoedanigheid vind ik dat de testrapporten over voldoende inhoud beschikken. Voor personen buiten dit project vind ik de testrapporten te beknopt.

Uiteindelijk heb ik niet voor iedere iteratie een apart testrapport opgeleverd. Aangezien ik binnen de uitgevoerde iteraties niet van testmethode ben veranderd, vond ik het daardoor overbodig om een apart testrapport op te leveren. Wanneer ik ontevreden was geweest met

de gehanteerde testmethode, dan had ik waarschijnlijk in een mogelijke iteratie van deze aanpak afgezien. In dat geval had ik wel een apart testrapport opgeleverd.

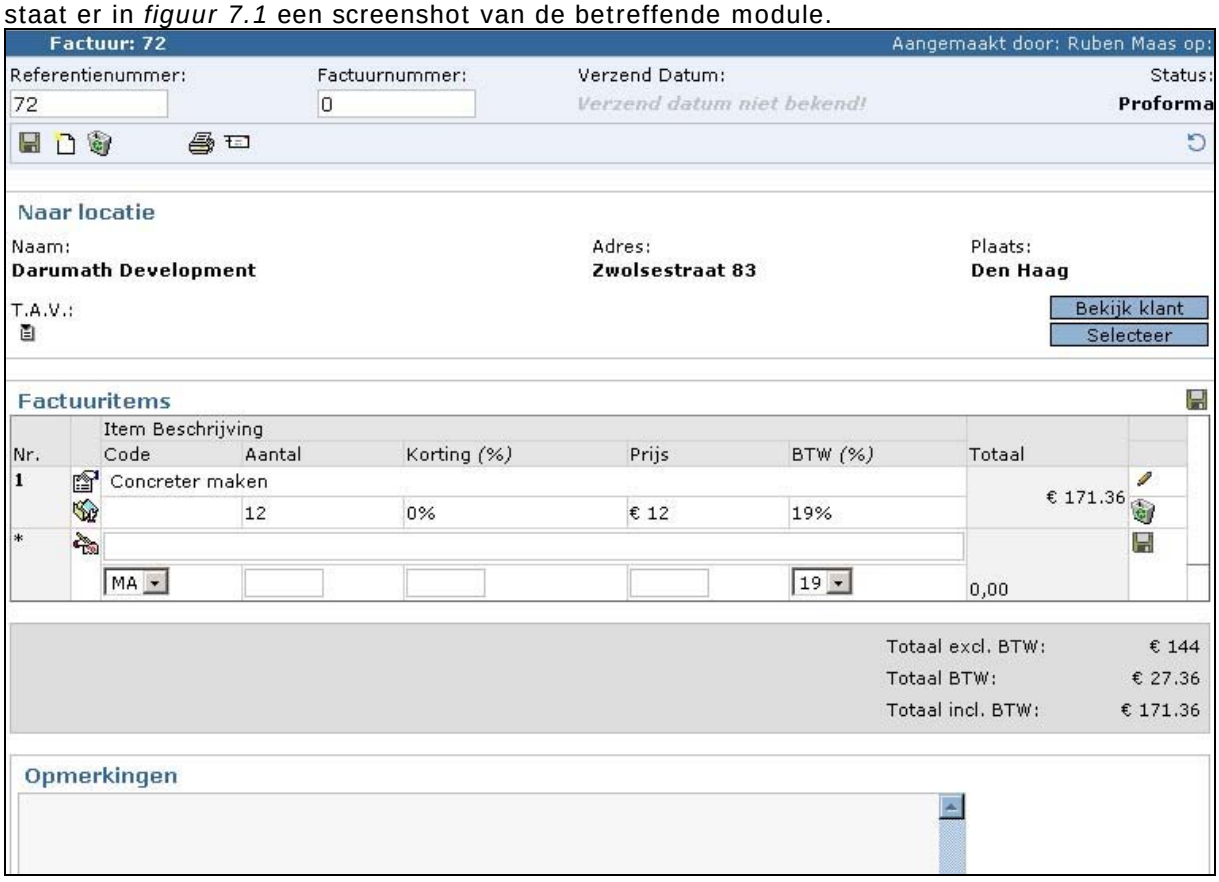
7.1.5 Handleidingen

Over de handleidingen heb ik gemengde gevoelens. Ik ben erg tevreden over de handleiding van de facturatiemodule. Ik heb ervoor gekozen deze digitaal te maken, zodat de lezer tevens gebruik kon maken van voorbeeldfilmpjes. Deze handleidingen staan uitgeprint in de bijlage. Bij deze handleiding heb ik het facturatieproces geïntegreerd binnen het algehele proces.

De handleiding waar ik erg ontevreden over ben is die van de module 'Beheer'. Door veel te weinig tijd is de handleiding te beknopt geworden. Eigenlijk kan ik het nauwelijks een handleiding noemen. Daarom heb ik hem uiteindelijk niet opgeleverd, omdat die op teveel punten tekort schoot. Naar mijn mening is dit een direct gevolg van het feit dat de betreffende pilot de laagste prioriteit genoot. Er is meer aandacht gegeven aan de pilot 'Facturatie' dan aan de pilot 'Beheer'. Hierdoor had ik te weinig tijd om de handleiding van de module 'Beheer' op te opleveren.

7.2 Module Facturatie

Ik ben zeer tevreden over het resultaat van de facturatiemodule. Aan de hand van de twee gebruikerstesten die ik heb uitgevoerd heb ik uiteindelijk kunnen vaststellen dat de module beschikt over voldoende kwaliteit. Om u een beeld te geven van het uiteindelijk resultaat heb staat er in *figuur 7.1* een screenshot van de betreffende module.



7.1 Factuurscherm module 'Facturatie

Figuur 7.1 laat het factuurscherm zien van de module 'Facturatie'. Bovenaan het scherm ziet u de zogenaamde header informatie van een factuur. Onder de header informatie wordt verstaan het referentienummer, factuurnummer, status en verzenddatum. Hieronder bevindt zich de toolbar waar vanuit de gebruiker taken kan uitvoeren zoals factuur bewaren en verwijderen en factuur aanmaken.

Vervolgens ziet u een frame waarin de gebruiker kan zien aan wie de factuur gericht is. Tevens heeft de gebruiker de mogelijkheid om van hieruit meer informatie van de klant op te vragen of een klant te selecteren.

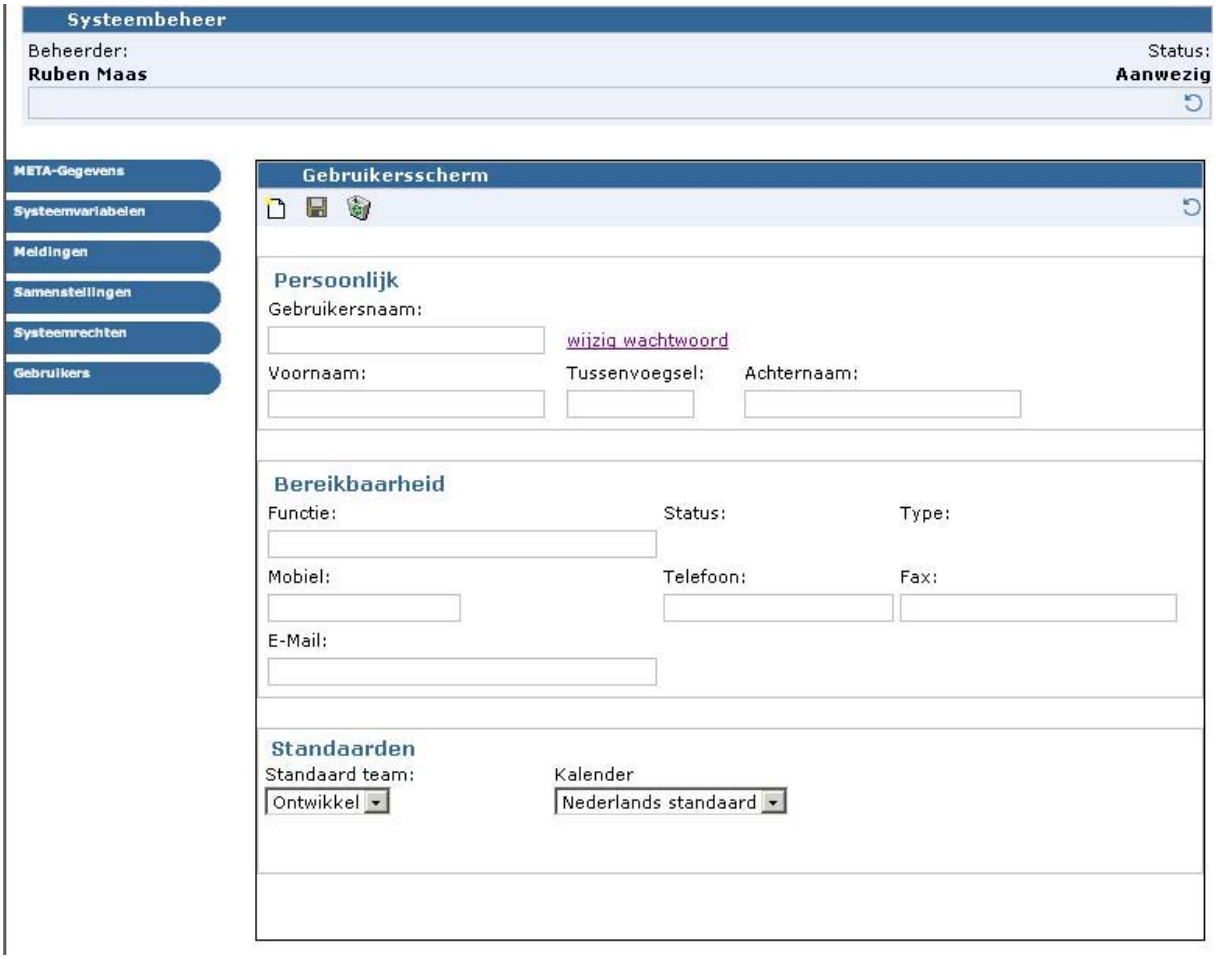
Binnen het gedeelte factuuritems kan de gebruiker de betreffende items bekijken, bewerken en aanmaken mits de factuur nog niet in een verzonden status verkeert. Tenslotte is er een ruimte overgelaten om de factuur totalen te bekijken en een opmerking te plaatsen bij de factuur.

Over het resultaat van de facturatie module ben ik erg tevreden. Terugkijkend naar de resultaten uit de gebruikerstesten durf ik te stellen dat de eindgebruikers er ook tevreden mee zijn. Het meest tevreden ben ik over de gehanteerde architectuur van de software. De wijze waarop de module is ontwikkeld biedt veel mogelijkheden voor de toekomst. Doordat functionaliteit en presentatie los van elkaar ontwikkeld zijn, bestaat de mogelijkheid diversen presentaties op zeer vlotte wijze te ontwikkelen. Ook de onderhoudbaarheid van de module is op deze wijze geoptimaliseerd.

Een negatief punt van de module facturatie vind ik, dat het facturatieproces, die deze module hanteert, moeilijk is aan te passen indien een organisatie een ander proces wil aanhouden. Ik heb getracht het facturatieproces zo standaard mogelijk te houden, zodat hier diverse organisaties gebruik van kunnen maken. Het is voor mij dan ook onduidelijk of dit in de toekomst enige belemmeringen met zich mee zal brengen. Mogelijke oplossingen hiervoor zal zijn geweest dat je het facturatieproces kan instellen aan de hand van parameters.

7.3 Module Beheer

Over de module 'Beheer' ben ik over het algemeen minder tevreden. De functionaliteit die binnen deze module geboden wordt is goed, maar de presentatie schiet op sommige punten te kort. *Figuur 7.2* laat het uiteindelijke resultaat zien.



7.2 Gebruikersscherm module 'Beheer'

Het scherm dat wordt getoond in *figuur 7.2* is opgedeeld in diverse delen. Bovenaan het scherm ziet u een vlak waarin zichtbaar wordt gemaakt welke gebruiker is ingelogd. Aan de linkerkant ziet u een menu waar vanuit de gebruiker een systeeminstelling kan selecteren. De

systeeminstellingschermen worden uiteindelijk getoond in het vlak naast het menu. Binnen dit vlak ziet u het scherm waarmee de systeeminstelling gewijzigd kan worden. In dit geval wordt het gebruikersscherm getoond waarin zoals u ziet gegevens omtrent een gebruiker ingevoerd, aangepast of verwijderd worden.

De oorzaak waarom ik niet geheel tevreden ben over de presentatie van deze module is dat ik een zeer strakke planning had voor de ontwikkeling van deze module en daardoor zeer weinig tijd had om de presentatie geheel naar mijn zin te ontwerpen. Binnen deze module vond ik het noodzakelijk dat de integriteit van het systeem niet in gevaar werd gebracht bij gebruik van deze module. Hierdoor kreeg de ontwikkeling van de data –en businesslaag meer prioriteit dan de ontwikkeling van de presentatielaag. Aan de hand van de UNIT-testen heb ik kunnen vaststellen dat de functionaliteit binnen dit systeem correct is doorgevoerd.

Kortom, ik ben redelijk tevreden met het resultaat, doordat de module voorziet in de doelstelling. Door het gebrek aan tijd ben ik hellaas niet in staat geweest de module, met betrekking tot de gebruikersinterface, naar volledige tevredenheid op te leveren.

8 Evaluatie doelstelling en probleemstelling

Het belangrijkste vraagstuk bij de evaluatie is toch wel: is er voldaan aan de doelstelling en in welke mate is het probleem opgelost? Om hierop antwoord te krijgen beantwoord ik zodoende deze vragen.

Heb ik aan de doelstelling voldaan?

De modules zijn werkende producten die aan alle systeemeisen voldoen en ook een goede beoordelingen hebben gekregen bij de gebruikerstesten. Aan de hand hiervan kom ik tot de conclusie dat de doelstelling is voldaan.

Is het probleem opgelost?

De vraag waar het nu allemaal om draait is of achteraf gezien het probleem verholpen is. Het probleem wat binnen de opdrachtomschrijving werd geconstateerd kwam neer op het feit dat er binnen de vroegere versie van het Key-Task systeem te kort werd geschoten bij de ondersteuning van een aantal zeer belangrijke bedrijfsprocessen. Denkend aan het applicatiebeheer en facturatieproces. De facturatiemodule, die tijdens dit project ontwikkeld is, beschikt over functionaliteiten om het facturatieproces volledig geautomatiseerd te laten plaatsvinden. Aan de hand van de gebruikerstesten heb ik kunnen constateren dat het probleem omtrent het factureren is opgelost. Handmatige handelingen worden, met het gebruik van de facturatiemodule, vermeden waardoor het facturatieproces veel vlotter verloopt.

De Beheermodule zorgt ervoor dat de systeeminstellingen gewijzigd kunnen worden. Aan de hand van de testen die zijn uitgevoerd heb ik kunnen beoordelen dat dit op correcte wijze plaatsvindt. De beheermodule vormt nu de hoeksteen van het Key-Task systeem en ik kan daarbij stellen dat het probleem omtrent het beheer van het Key-Task systeem is opgelost.

9 Bronnenlijst

Literatuur

Titel	Auteur	Uitgever
Graphical User Interface Design and Evaluation	Rumbaugh, J	Prentice Hall inc.
Evololutionair ontwikkelen van informatiesystemen	Tolido R.J.H.	Academic Service, 1997
Graphical User Interface Design and Evaluation	Redmond-Pyle, D & Moore	Prentice Hall Europe
Databasesystemen voor de praktijk	Vandenbulcke	Kluwer
Active Server Pages	Divers	Wrox press
JavaScript	Cliff Wooton	Wrox press
Database programming with visual basic 6	Petroutos Evangelos	Sybex

Internet

Titel	Locatie
DHTML Libary	http://msdn.microsoft.com/workshop/author/dhtml/reference/objects.asp

10 Woordenlijst

Definitie	Beschrijving
3-lagen model	Een wijze waarop de software architectuur binnen een systeem wordt toegepast.
ASP	Staat voor Active Server Pages. Het is een server-side script, die gebruikt wordt voor het ontwikkelen van web-applicatie.
Browser	Een programma waarmee je internet pagina's kunt bezoeken.
Businesslaag	Een gedeelte uit de softwarearchitectuur, die wordt gebruikt om de systeemfunctionaliteiten te huisvesten.
Datalaag	Een gedeelte uit de softwarearchitectuur, die wordt gebruikt om de integriteit van het systeem te waarborgen.
DFD	Staat voor Data Flow Diagram. Het is een modelleringtechniek, waarmee bepaalde processen ontworpen en geanalyseerd kunnen worden.
DHTML	Dynamic Hyper Text Marup Language. Het is een combinatie van HTML, Stylesheets en JavaScript. Met behulp van DHTML kunnen uiterlijkheden kenmerken binnen de internet pagina gewijzigd worden.
DLL	Staat voor Dynamic Link Library. Een DLL wordt gebruikt om systeemfunctionaliteit in te huisvesten. Een DLL kan beschikken over meerdere objecten die extern te benaderen zijn. Een ander woord voor DLL is ook wel klassenbibliotheek.
Heuristic Evaluation	Een testmethode waar aan de hand van meerdere experts de usability van de software wordt getest.
HTML	Staat voor Hyper Text Markup Language. Is een standaard opmaaktaal die door internet browsers wordt gebruikt om beeld te visualiseren.
IAD	Staat voor Iterative Application Development. Het is een ontwikkelmethode die zich kenmerkt door de hoge mate van gebruikersparticipatie.
IFRAME	Een HTML-element die gebruikt kan worden als overlappend kader waarin internet pagina's geladen kunnen worden.
Incrementeel ontwikkelen	Een ontwikkelstrategie van IAD, waarbij de fase pilotontwikkeling iteratief wordt ontwikkeld.
JavaScript	Scripttaal (client-side).
Klassenbibliotheek	Een bestand dat beschikt over één of meerdere objecten. Deze objecten kunnen gebruikt worden om van extern referentie naar te leggen. Klassenbibliotheken zijn meestal in de vorm van een DLL (*Dynamic Link Library*)
Metrix 4e	Systeem van Metrix ter ondersteuning van operationele processen
Pilot	Een apart in te voeren bouweenheid van een applicatie.
Presentatielaag	Een gedeelte uit de softwarearchitectuur, die wordt gebruikt voor het huisvesten van de gebruikersinterface.
Pro forma	Een stadium waarin zich een factuur kan bevinden. Wanneer een factuur zich in een pro forma stadium verkeerd, dan wil dat zeggen dat de factuur nog niet is gecontroleerd en verzonden.
Proof of concept	Een experiment.
RAD	Staat voor Rapid Application Development. Het is een ontwikkelmethode.
SQL	Staat voor Structured Query Language. Een taal die wordt die wordt gehanteerd voor het opvragen van gegevens uit de database.
Use Case	Scenario's die kunnen worden gebruikt voor taakanalyse of gebruikerstesten.
Visual Basic	4de generatie programmeertaal.

Addendum

Afstudeerverslag

De ontwikkeling van de modules 'Facturatie' en 'Beheer'

Auteur:	Ruben Maas
Studentnummer:	99008637
Datum:	09-01-2004
In opdracht van:	Metrix B.V.

Inhoudsopgave

Bijlage 1 ***Opdrachtomschrijving***

Bijlage 2 ***Plan van aanpak***

Bijlage 3 ***Definitiestudie***

Bijlage 4 ***Pilotontwikkelplan 'Facturatie'***

Bijlage 5 ***Testrapport 'Facturatie'***

Bijlage 6 ***Pilotontwikkelplan 'Beheer'***

Bijlage 7 ***Testrapport 'Beheer'***

Bijlage 8 ***Handleiding facturatiemodule***

Opdrachtsomschrijving Afstuderen

Student: Ruben Maas 99008637

Titel: De ontwikkeling van de modules ‘Beheer’ en ‘Facturatie’.

Omschrijving: Het ontwikkelen van de modules 'Beheer' en ‘Facturatie’ als toevoeging aan het Key-Task systeem.

1. Inleiding

Metrix B.V. (Metrix Europa) is een volle dochter van Metrix USA. Metrix is een leverancier van Field Service Management software. In de Metrix producten draait het om eProductService oftewel de organisatie van het verlenen van service op verkochte producten. Metrix ProductService wordt ook wel in de markt een CRM-oplossing genoemd, maar dan volledig gericht op productondersteuning. CRM staat voor Customer Relation Management en omvat automatisering, die de mogelijkheid biedt informatie vast te leggen over instellingen (bedrijven), contactpersonen en projecten en deze onderling te koppelen om ze vervolgens eenvoudig te kunnen benaderen.

Verder kijkt Metrix B.V. naar de toekomst en houdt zich bezig met nieuwe technologieën. In het kader daarvan heeft Metrix stageplekken beschikbaar gesteld aan studenten, die zich hebben beziggehouden met o.a het volgende produkt:

Key-Task
Doordat Metrix veelal in aanraking kwam met bedrijven, die zich qua vermogen en bezetting te klein achtten voor Metrix ProductService. Heeft Metrix een CRM pakket laten ontwikkelen voor bedrijven uit het MKB. Hierbij valt Key-Task te vergelijken met Metrix alleen is Key-Task afgericht op de behoeftes van kleine organisaties. Dit maakt Key-Task goedkoop en zeer eenvoudig implementeerbaar.

Key-Task is een Service Management systeem. Service Mangement houdt in: Het beheer van service verlenende activiteiten van een organisatie. Een voorbeeld:
Stel je verkoopt wasmachines en je hebt daarbij 3 reparateurs in dienst, die ervoor moeten zorgdragen dat indien een wasmachine kapot is deze wordt gerepareerd. Een reparateur wordt naar de locatie gestuurd en er volgt uiteindelijk een verrekening, omdat de reparateur bepaalde onderdelen van een wasmachine moest vervangen.

Een service management systeem ondersteunt het bijbehorende proces. Zo moet er bekend zijn waar het probleem zich afspeelt welke reparateurs er beschikbaar zijn en welke kosten er zijn gemaakt.

Het Key-Task systeem houdt daarom de volgende gegevens bij:

1. De Klantgegevens
   Wie hebben er allemaal wasmachines gekocht.
2. Productgegevens
   Welke wasmachines bevinden zich bij welke klant.
3. Activiteiten

Omvat gegevens over het gemelde probleem.

4. Taken
   Bedrijfsmatige opdrachten aan medewerkers, zoals het repareren van een wasmachine.
5. Onkosten
   Kosten die zijn gemaakt bij het uitvoeren van de activiteit (probleem).
6. Agenda
   De werktijden van de medewerker. Wanneer is een reparateur beschikbaar.

2. Probleemstelling

Het Key-Task systeem is vooralsnog nog niet geheel bruikbaar voor organisaties binnen het MKB. Er zijn een aantal problemen die een succesvolle implementatie van het systeem in de weg staan.

Allereerst is het in de huidige situatie niet mogelijk het systeem te beheren. Het systeem beschikt over instellingen, die invloed uitoefenen op de werking van het systeem. In de huidige situatie kunnen die instellingen door de eindgebruiker niet worden aangepast. Tevens is er geen overzicht in de instellingen die het systeem gebruikt. Daarom kan het Key-Task systeem niet volledig worden afgericht op de betreffende organisatie.

Tevens blijkt, uit de mening van potentiele klanten, dat het systeem beperkt is in het gebruik, omdat het verrekeningsproces ontbreekt. Het gevolg hiervan is dat gebruikers in de meeste gevallen moeten terugvallen op het handmatig samenstellen van facturen. Buiten het feit dat dit probleem een dergelijke organisatie veel tijd en geld kost, is er geen overzicht welke kosten er reeds zijn verrekend.

3. Doelstelling

De uitvoerder moet ervoor zorgen dat het Key-Task systeem volledig voorziet in de wensen van de eindgebruiker, zodat het systeem succesvol kan worden geimplementeerd. Hiervoor beschrijft de opdracht de volgende toevoegingen aan het huidige Key-Task systeem:

1. Beheer-module Binnen dit systeemdeel dient de gebruiker de mogelijkheid te hebben om de bestaande systeeminstellingen op te vragen en te wijzigen. Zodoende heeft de gebruiker de mogelijkheid het systeem naar haar organisatie in te richten.

2. Facturatie-module Binnen dit systeemdeel moet de gebruiker de mogelijkheid hebben de gemaakte kosten voor een activiteit aan de klant te verrekenen. Dit verrekeningsproces dient geheel geautomatiseerd plaats te vinden.

Bij de ontwikkeling van de bovenstaande modules dienen de integriteits, interface, performance en operationele-eisen, die aan Key-Task zijn gekoppeld, te worden gehandhaafd. Tevens dienen de usability aspecten van Key-Task te worden voortgezet.

Tenslotte dienen de handleidingen van Key-Task te worden uitgebreid met de handleidingen van de te ontwikkelen modules.

4. Op te leveren producten

De volgende producten moet de uitvoerder van dit project opleveren:

1. Plan van aanpak

2. Defintiestudie

3. Pilotontwikkelplannen (*Voorlopig 2 maal*)

3. Module 'Beheer'

4. Testrapport 'Beheer' *(Minimaal 1)*

5. Module 'Facturatie'

6. Testrapport 'Facturatie' *(Minimaal 1)*

7. Handleiding 'Beheer

7. Handleiding 'Facturatie'

5. Uitgangssituatie

5.1 Benodigde software

Key-Task
Microsoft Internet Information Server
Microsoft Visual Studio 5 of 6
Microsoft Access 2000 of hoger
Microsoft Word
Microsoft SQL Server
Microsoft Visio 2000

5.2 Benodigde hardware

Minimaal 1 P.C vanaf Pentium 2
Internetverbinding

5.3 Beschikbare rapporten

Van de ontwikkeling van Key-Task zijn de volgende rapporten beschikbaar:

Definitiestudie
Pilotontwikkelplannen
Stageverslagen

6. Concrete werkzaamheden:

Binnen deze paragraaf zal zal ik ingaan op de werkzaamheden binnen dit project. Hierbij zal een uitleg gegeven worden over de uit te voeren activiteiten, de te hanteren methodieken en de te gebruiken technieken.

6.1 Uit te voeren activiteiten

Het maken van een plan van aanpak

Hierin zal ik mij gaan bezig houden met het nauwkeurig inplannen van de activiteiten omtrent dit project, waarna we het in de loop van het project als leidraad kunnen gebruiken.

Het maken van een definitiestudie

Binnen deze activiteit zal ik mij gaan bezighouden met het bepalen van de doelen en de beperkingen van de te ontwikkelen programmatuur. Dit wordt bewerkstelligd aan de hand van interviews/workshops en de systeemdocumentatie van Key-Task.

Het maken van pilotontwikkelplannen

Van iedere pilot zal een ontwikkelplan worden geschreven. Dit document geeft inzicht in de ontwikkeling van de betreffende pilot. U kunt hierbij denken aan navigatieschema's en objectmodellen.

Het bouwen van de pilots

Aan de hand van de pilotontwikkelplannen worden de pilots gebouwd door de uitvoerende teams.

Het maken van testplannen en verwerken van resultaten

Aan de hand van een test wordt bepaald of een pilot voldoet aan de eisen en doelen die zijn opgesteldt binnen de fase definitiestudie en of er een volgende iteratie geschied. Hiervoor dient de uitvoerder eerst een testplan te schrijven. Het testplan en de verwerkte resultaten vormen op haar beurt het test-rapport. Er zal dan ook voor iedere iteratie een bijsluitend testrapport worden opgeleverd.

6.2 Te hanteren methodieken

Dit project zal uitgevoerd worden aan de hand van de ontwikkelmethode IAD.

6.3 Te gebruiken technieken

SUMI

Zal worden gebruikt om de pilots te testen op bruikbaarheid en kwaliteit.

Object modelling

Zal worden gebruikt om de huidige en gewenste situatie van het systeem in kaart te brengen.

Interviewen

Zal mogelijk worden gebruikt om informatie te winnen voor de fase definitiestudie en pilotontwikkeling.

Workshops

Zal mogelijk worden gebruikt om informatie te winnen voor de fase definitiestudie en pilotontwikkeling.

Planningstechnieken (time boxing)

Zal worden gebruikt om de diverse activiteiten binnen dit project in te plannen.

Met de hierboven staande technieken worden ondersteunende technieken bedoeld, het uiteindelijke product wordt gerealiseerd met de volgende technieken:

ASP
Microsoft visual basic
HTML editor (notepad)

6. Planning (globaal)

Wk	**Activiteiten**
1	Maken plan van aanpak
2	Maken defenitiestudie
3	Start ontwikkeling module 'Facturatie
6	Afsluiting module 'Facturatie'
7	Start ontwikkeling module 'Beheer'
14	Afsluiting module 'Beheer'
19	Uitloop

8. Resultaten voor de opdrachtgever (op te leveren producten):

Het Key-Task systeem inclusief de te ontwikkelen modules.
Een definitiestudie
De pilotontwikkelplannen

9. Relatie blok 6 en 7 (verantwoording afstudeerproject):

Dit project sluit aan wat we bij MI-33 hebben geleerd. Bij dit vak hebben wij geleerd hoe een gebruikerstest voor te bereiden en uit te voeren.

Plan van aanpak

Naam document:	Plan van aanpak
Bestandsnaam:	Pva_v2.doc
Datum oplevering:	05/09/2003
Omschrijving van de inhoud:	Dit document beschrijft de uit te voeren opdracht en de planning
In opdracht van:	Metrix B.V.
Versie:	1.00
Auteur:	Ruben Maas

Voorwoord

Dit rapport is geschreven in het kader het afstuderen. Dit document dient als houvast voor de afstudeerder en geeft de opdrachtgever en de overige betrokken partijen inzicht in de uit te voeren werkzaamheden alsmede een planning wanneer de werkzaamheden uit moeten worden gevoerd.

Inhoudsopgave

Plan van aanpak **1**
1. Inleiding **4**
2. Bedrijfs- en opdrachtsachtergrond **5**
2.1 Het bedrijf 5
3. De Opdracht **7**
3.1 De probleemstelling 7
3.2 De doelstelling 7
3.4 De op te leveren producten 8
3.4.1 Plan van aanpak 8
3.4.2 Definitiestudie 8
3.4.3 Pilotontwikkelplannen 8
3.4.5 Module 'Facturatie' 9
3.4.5 Module 'Beheer' 9
3.4.6 Testrapporten 9
3.4.7 Systeemdocumentatie 9
4. De betrokkenheid van participerende partijen **10**
4.1 De opdrachtgever 10
4.2 De projectbegeleider 10
4.3 De gebruikers 10
5. De risicoanalyse **11**
5.1 onkunde met de programmeertaal 11
5.2 Risico haalbaarheid project 11
5.3 Inschattingfouten 11
5.4 Verandering van de doelstelling 11
6. De ontwikkelmethode en technieken **12**
6.1 De ontwikkelmethode 12
6.2 De technieken 12
7. Beheersaspecten **14**
7.1 De factor geld 14
7.2 De factor organisatie 14
7.3 De factor tijd 14
7.4 De factor informatie 14
7.5 De factor kwaliteit 14
8. De planning **16**
8.1 De globale planning tekstueel 16
8.2 De globale planning visueel 16
8.3 De gedetailleerde planning visueel 18

1. Inleiding

Dit document is bedoeld om een globale indruk te geven van wat hoe en wanneer er iets gedaan moet worden binnen het afstudeerproject. Aangezien dit document aan het begin van het project geschreven is zullen een aantal zaken later aangepast worden. Dit document dient tevens als input voor de definitiestudie.
Allereerst wordt de achtergrond van het bedrijf behandeld, vervolgens de doelstelling en het resultaat van de opdracht, vervolgens de op te leveren producten. Dan berokken personen, de risico's, de ontwikkelmethoden en de benodigde faciliteiten van dit project. Ook bevat dit document nog een globale en een gedetailleerde planning en komt de benodigde hard- en software aan bod.

2. Bedrijfs- en opdrachtsachtergrond

In dit hoofdstuk wordt de achtergrond van de opdracht en het bedrijf beschreven.

2.1 Het bedrijf

Metrix B.V. (Metrix Europa) is een volle dochter van Metrix USA. Metrix is een leverancier van Field Service Management software. In de Metrix producten draait het om eProductService oftewel de organisatie van het verlenen van service op verkochte producten. Metrix ProductService wordt ook wel in de markt een CRM-oplossing genoemd, maar dan volledig gericht op productondersteuning. CRM staat voor Customer Relation Management en omvat automatisering, die de mogelijkheid biedt informatie vast te leggen over instellingen (bedrijven), contactpersonen en projecten en deze onderling te koppelen om ze vervolgens eenvoudig te kunnen benaderen.

Metrix eProductService is een applicatie in het topsegment van aanbieders op service management. Klanten van Metrix zijn voornamelijk multi-nationals zoals Ericsson, Dade-Behring, Delft Instruments, Xerox, Oce, Cisco, De LaRue en Vivendi Water.

Verder kijkt Metrix B.V. naar de toekomst en houdt zich bezig met nieuwe technologieën. In het kader daarvan heeft Metrix stageplekken beschikbaar gesteld aan studenten, die zich hebben beziggehouden met de volgende producten:

Key-Task
Key-Task is een Service Management systeem. Service Management houdt in: Het beheer van service verlenende activiteiten van een organisatie. Een voorbeeld:
Stel je verkoopt wasmachines en je hebt daarbij 3 reparateurs in dienst, die ervoor moeten zorgdragen dat indien een wasmachine kapot is deze wordt gerepareerd. Een reparateur wordt naar de locatie gestuurd en er volgt uiteindelijk een verrekening, omdat de reparateur bepaalde onderdelen van een wasmachine moest vervangen.

Een service management systeem ondersteunt het bijbehorende proces Zo moet er bekend zijn waar het probleem zich afspeelt welke reparateurs er beschikbaar zijn en welke kosten er zijn gemaakt.

Het Key-Task systeem houdt daarom de volgende gegevens bij:

1. De Klantgegevens
   Wie hebben er allemaal wasmachines gekocht.
2. Productgegevens
   Welke wasmachines bevinden zich bij welke klant.
3. Activiteiten
   Omvat gegevens over het gemelde probleem.
4. Taken
   Bedrijfsmatige opdrachten aan medewerkers, zoals het repareren van een wasmachine.
5. Onkosten

Kosten die zijn gemaakt bij het uitvoeren van de activiteit (probleem).

6. Agenda
   De werktijden van de medewerker. Wanneer is een reparateur niet beschikbaar om ingezet te worden.

Binnen Key-Task is er nog geen mogelijkheid om een overzicht op te vragen van de taken die het personeel moet uitvoeren. De Key-Task Taskmanager moet ervoor zorgen dat het inplannen van de taken wordt geautomatiseerd. De Key-Task Taskmanager moet u dan ook zien als uitbreiding op een al bestaand systeem.

3. De Opdracht

In dit hoofdstuk wordt de opdracht omschreven zoals die gedurende het afstudeer project zal worden uitgevoerd.

3.1 De probleemstelling

Het Key-Task systeem is vooralsnog nog niet geheel bruikbaar voor organisaties binnen het MKB. Er zijn een aantal problemen die een succesvolle implementatie van het systeem in de weg staan.

Allereerst is het binnen de huidige situatie niet mogelijk het systeem te beheren. Het systeem beschikt over instellingen, die invloed uitoefenen op de werking van het systeem. Binnen de huidige situatie kunnen die instellingen door de eindgebruiker niet worden aangepast. Tevens is er geen overzicht in de instellingen die het systeem gebruikt. Gezien dit feit kan het Key-Task systeem niet volledig worden afgericht op de betreffende organisatie.

Tevens blijkt, uit de mening van mogelijke klanten uit de doelgroep, dat het systeem beperkt is in het gebruik, omdat het verrekeningsproces ontbreekt. Het gevolg hiervan is dat gebruikers in de meeste gevallen moeten terugvallen op het handmatig samenstellen van facturen. Buiten het feit dat dit probleem een dergelijke organisatie veel tijd en geld kost, is er geen overzicht welke kosten er reeds zijn verrekend.

3.2 De doelstelling

De uitvoerder moet ervoor zorgen dat het Key-Task systeem succesvol kan worden geïmplementeerd binnen organisaties van het MKB. Hiervoor dienen de volgende toevoegingen te worden gedaan aan het systeem:

1. Beheer-module Binnen dit systeemdeel moet de systeeminstellingen kunnen worden gepresenteerd en worden gewijzigd door de gebruiker.

2. Facturatie-module Binnen dit systeemdeel moet de gebruiker de mogelijkheid hebben de gemaakte kosten voor een activiteit aan de klant te verrekenen.

Bij de ontwikkeling van de bovenstaande modules dienen de integriteits, interface, performance en operationele-eisen, die aan Key-Task zijn gekoppeld, te worden gehandhaafd. Tevens dienen de usability aspecten van Key-Task te worden voortgezet.

Tenslotte dienen de handleidingen van Key-Task te worden uitgebreid met de handleidingen van de te ontwikkelen modules.

3.4 De op te leveren producten

De volgende opdrachten dienen te worden opgeleverd:

1. Plan van aanpak
2. Definitiestudie
3. Pilotontwikkelplannen (*Voorlopig 2 maal*)
4. Module 'Beheer'
5. Module 'Facturatie'
6. Testrapporten (*Voorlopig minimaal 2*)
7. Handleiding 'Beheer'
7. Handleiding 'Facturatie'

3.4.1 Plan van aanpak.

De volgende inhoudelijke eisen worden hieraan gesteld:

1. Het bevat een goedgekeurde opdrachtomschrijving
2. Beschrijving te gebruiken Methoden en technieken
3. Beschrijving uit te voeren activiteiten.
4. Risico-analyse
5. Een planning van de uit te voeren werkzaamheden.
6. Beheersaspecten

3.4.2 Definitiestudie

De volgende inhoudelijke eisen worden hieraan gesteld:

1. Beschrijving van het ontwikkelscenario en de context van het ontwikkeltraject.
2. Systeemeisen
3. Systeemconcept
4. Beschrijving Technische structuur
5. Organisatorische inrichting
6. Het pilotplan

3.4.3 Pilotontwikkelplannen

De pilotontwikkelplannen moeten de volgende zaken bevatten:

1. Plan van aanpak
2. Pilotontwerp-workshop
3. Globaal-functionele structuur
4. Organisatorische inrichting
5. Ontwerp software-bouweenheden

3.4.5 Module 'Facturatie'

De module 'Facturatie' wordt opgeleverd alszijnde een object type library (DLL-bestand) en een set van ASP-pagina's.

3.4.5 Module 'Beheer'

De module 'Beheer' wordt opgeleverd alszijnde een object type library (DLL-bestand) en een set van ASP-pagina's.

3.4.6 Testrapporten

Aan het eind van iedere iteratie uit de fase pilotontwikkeling worden gebruikerstesten uitgevoerd om te kunnen bepalen of de pilot aan voldoende functionaliteit beschikt om succesvol ingevoerd te kunnen worden. In dien dit niet het geval is dan zal er mogelijk een nieuwe iteratie gestart kunnen worden waarin aanpassing aan de pilot uitgevoerd kunnen worden.

3.4.7 Systeemdocumentatie

Dit document bevat cruciale informatie ten behoeve van systeemontwikkelaars die zich bezighouden met het verbeteren, aanpassen of uitbreiden van delen van het informatie systeem. De volgende punten zullen worden moeten terug gevonden in dit document:

1. Technische beschrijving
2. Systeemarchitectuur
3. Object Model
4. Overzicht Constanten
5. Functioneel Model
6. Beschrijving Procedures

4. De betrokkenheid van participerende partijen

Bij dit project zijn een aantal participerende partijen betrokken, te weten;

- Metrix BV (de opdrachtgever)
- Dhr. D. Stoikof (Contact persoon en Technical Consultant)
- Dhr. R Maas (Projectuitvoerder)
- Darumath (Gebruiker)

4.1 De opdrachtgever

De opdrachtgever heeft de taak zich te betrekken bij het project. Hij heeft de taak controles uit te voeren op het ontwikkelde ten aanzien van de juistheid ervan, de afstudeerder te voorzien van de benodigde informatie ten behoeve van de opdracht.

4.2 De projectbegeleider

De projectbegeleider heeft de taak de afstudeerder te ondersteunen in de taken die hij moet uitvoeren om de opdracht te kunnen voltooien. Eveneens hebben zij als functie om de voortgang van het project te bewaken en kwaliteitscontrole van het project.

4.3 De gebruikers

Voor dit project is gezocht naar een aantal gebruikers die gedurende de uitvoer van dit project onderdeel kunnen uitmaken van het algehele ontwikkelproces. Hiervoor hebben we benaderd Darumath. Dit bedrijf verzorgt grafische opdrukken op allerlei materialen.

5. De risicoanalyse

In dit hoofdstuk worden de risico's die tijdens de uitvoering van de opdracht worden gelopen geanalyseerd en wordt er aan oplossingen gedacht om dit risico te vermijden anders te minimaliseren.

5.1 onkunde met de programmeertaal

Het kan zo zijn dat er voor een bepaalde functionaliteit specifieke kennis van de programmeertaal vereist is waar de kunde van de ontwikkelaar niet toereikend is. Deze onkunde zal dan weggewerkt moeten worden door middel van bijscholing, al dan niet het raadplegen van internet of studiemateriaal. Dit soort problemen kunnen echter leiden tot vertraging van het project.

5.2 Risico haalbaarheid project

De haalbaarheid van de koppeling van huidige Key-Task systeem en de modules is een risico.

5.3 Inschattingfouten

Fouten maken is menselijk. De ontwikkelaar kan bijvoorbeeld fouten hebben gemaakt in de planning zodat de opdracht niet in zijn oorspronkelijke staat volbracht kan worden. Er is altijd de mogelijkheid om de planning indien nodig aan te passen.

5.4 Verandering van de doelstelling

Hoewel aan het begin van de opdracht de doelstelling duidelijk is geformuleerd kan het zijn dat er omstandigheden zijn waardoor deze doelstellingen veranderen. Hierdoor zou ook de opdracht kunnen veranderen.

Wanner zaken als doelstellingen drastisch worden veranderd zal er contact opgenomen worden met de betrokkenen die gezamenlijk een oplossing zoeken. In het uiterste geval zal de opdrachtomschrijving opnieuw worden geschreven en gekeurd waardoor het proces (helaas) opnieuw moet beginnen.

6. De ontwikkelmethode en technieken

In dit hoofdstuk wordt een uiteenzetting gedaan over de gebruiken ontwikkelmethode en technieken.

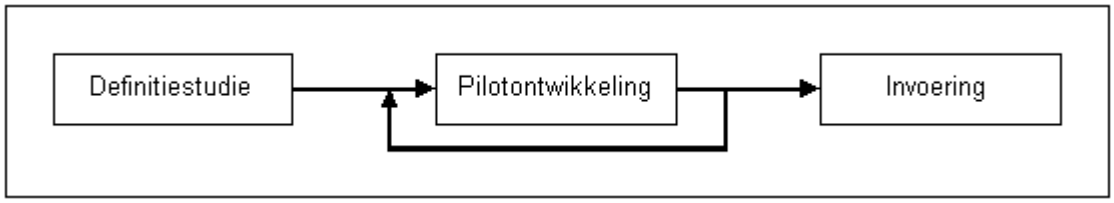
6.1 De ontwikkelmethode

Er zijn tal van methodes beschikbaar om de te ontwikkelen modules te ontwikkelen. Ik zal gedurende dit project gebruik maken van de ontwikkelmethode IAD.
Dit is een ontwikkelmethode die is gebonden aan de eisen die de opleiding Vormgeving en Ontwerp van Interactie van de sector informatica op de Haagse Hogeschool stelt aan het ontwerpen en realiseren van producten.
Kijkend naar het feit dat dit project een korte looptijd heeft is de keuze voor IAD berust op de ervaring die wij reeds hebben opgedaan.

Het gehele ontwikkeltraject zal daarom een iteratief karakter hebben en het systeem zal in nauwe samenwerking met de toekomstige gebruikers van het systeem gedefinieerd en ontworpen moeten worden. Als ontwikkelmethodiek is gekozen voor IAD om de gebruikersparticipatie te optimaliseren.

Binnen IAD zijn er een aantal strategieën beschikbaar die beschrijven op welke wijze een systeem wordt ontwikkeld. Dit is afhankelijk van vele factoren zoals: Soort systeem, Team samenstelling, Geld, Tijd etc…

Gedurende dit project zal ik gebruik maken van de ontwikkelstrategie 'Incrementeel Ontwikkelen'. Kort gezegd worden aan de hand van deze strategie eerst de systeemeisen volledig in kaart gebracht en afgebeeld op een systeemconcept. Daarna wordt het gehele systeem via een aantal iteraties van een en dezelfde fase pilotontwikkeling ontwikkeld. De invoering vindt in een keer plaats. Ik zal dit verduidelijken aan de hand van figuur 6.1.



6.2 De technieken

Tijdens het uitvoeren van dit project zal ervan een groot aantal technieken gebruik worden gemaakt waarvan er een aantal op het moment van schrijven nog niet bekend zijn. Zeker zal er gebruik worden gemaakt van de volgende:

- Object Modelling Techniek
- Observatietechnieken
- Interviewtechnieken
- Planningstechnieken (time boxing)
- Evaluatietechnieken

- Objectorientatie
- Navigatieschema
- Taakanalyse
- Relationeel databasemodel
- Functioneel databasemodel

Wanneer blijkt dat er additionele technieken benodigd zijn dan zullen deze worden toegepast, uiteraard zal dit worden vermeld.

7. Beheersaspecten

In dit hoofdstuk zullen de zogenaamde beheersaspecten ook wel GOTIK factoren worden besproken. Dit is om een beter inzicht te krijgen in de aspecten zoals, **G**eld, **O**rganisatie, **T**ijd, **I**nformatie en **K**waliteit.

7.1 De factor geld

Naar aanleiding van dit project zal geen enkele vergoeding, in welke vorm dan ook, aan de projectleden worden uitgekeerd.

7.2 De factor organisatie

Organisatie van dit project is in grote mate afhankelijk van een gedegen en gedetailleerde planning, te vinden in dit plan van aanpak. De belangrijkste zaken die verder in de gaten moeten worden gehouden zijn

- Op tijd leveren van zaken die de onderwijsinstelling wil zien (voortgangsverslagen en eindverslag)
- Aanwezigheid bij afspraken met de projectbegeleider.
- Producten die de opdrachtgever wil zien (te vinden bij het kopje op te leveren producten)

7.3 De factor tijd

Het project beslaat 10 weken in de op de volgende pagina afgedrukte planning staan deze 10 weken met de globale activiteiten in.

7.4 De factor informatie

Informatie voor de opdracht wordt verkregen uit/van

1. Eigen literatuur
2. Literatuur van het bedrijf
3. Internet
4. Feedback van Metrix BV en haar medewerkers
5. Eigen inbreng
6. Systeem documentatie Key-Task

7.5 De factor kwaliteit

De op te leveren producten worden door de projectgroep zelf op kwaliteit gecontroleerd. Zo is er een sjabloon beschikbaar voor documenten. Hierdoor zullen alle documenten dezelfde layout hebben en alle lettertypen hetzelfde zijn.

Door een goede taakverdeling binnen de projectgroep zal ook de kwaliteit hoog blijven. Zo is er een projectmanager die alles in goede banen leidt. Een planner om

de planning te waarborgen en een documentbeheerder om de documenten overzichtelijk te beheren.
De projectgroep werkt met verschillende methoden en technieken zoals eerder genoemd in dit document. Deze zullen mede bijdragen aan een goede kwaliteit.

8. De planning

in dit hoofdstuk wordt een uiteenzetting van de activiteiten in de tijd weergegeven. Dit zal gebeuren met een globale plannig en met een gedetaileerde plannnig.

8.1 De globale planning tekstueel

Globale Planning Tekstueel

Week	Activiteit
1	Opleveren Plan van aanpak / Start Defenitiestudie
3	Opleveren Definitiestudie
4	Start Pilotontwikkelplan 'Facturatie'
6	Opleveren Pilotontwikkelplan 'Facturatie'
7	Start bouw pilot 'Facturatie'
11	Opleveren module ‘Facturatie’
12	Opleveren Handleiding ‘Facturatie’
13	Start Pilotontwikkelplan ‘Beheer’
14	Opleveren Pilotontwikkelplan 'Beheer'
15	Start bouw pilot ‘Beheer’
17	Opleveren module ‘Beheer’
18	Opleveren Handleiding ‘Beheer’

8.2 De globale planning visueel

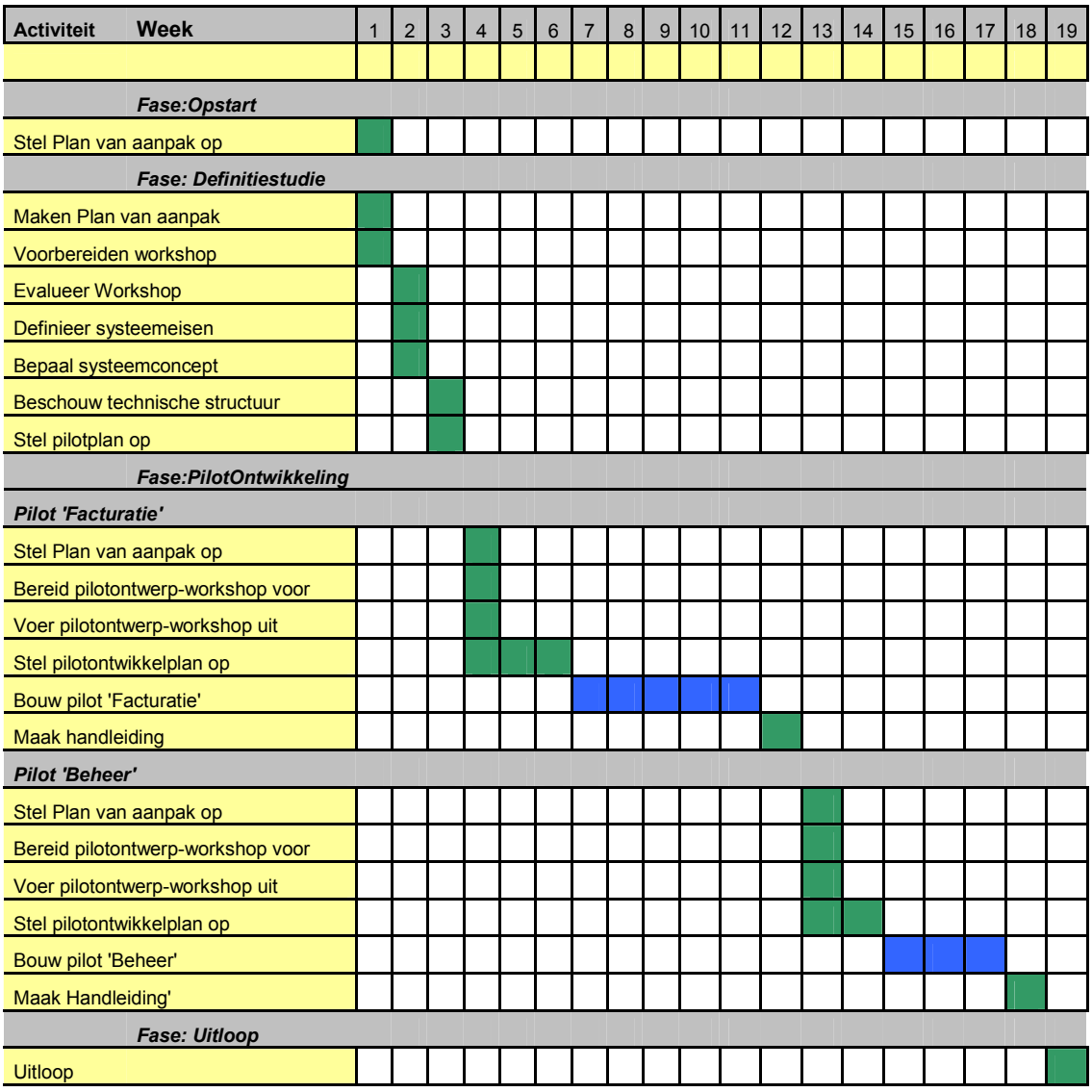
Activiteiten \ Weken	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Maken Plan van aanpak																			
Maken Defenitiestudie																			
Maken Pilotontwikkelplan 'Facturatie'																			
Bouwen pilot 'Facturatie'																			
Maken Handleiding 'Facturatie'																			
Maken Pilotontwikkelplan 'Beheer'																			
Bouwen pilot 'Beheer'																			
Maken Handleiding ‘Beheer’																			
Uitloop																			

Kijkend naar de bovenstaande planning ziet u dat er in week 1 wordt begonnen met zowel het Plan van aanpak als de definitiestudie. De tijdsperiode voor deze twee fases binnen het ontwikkelingsproces zijn geminimaliseerd, zodat er grotere tijdsperiodes beschikbaar zijn voor de pilotontwikkeling. Gezien het feit dat ik de ontwikkelstrategie ‘Incrementeel ontwikkelen’ zal gaan gebruiken wordt er vanuit dat

oogpunt meer aandacht besteedt aan de pilotontwikkeling dan aan de definitiestudie. De definitiestudie zal daarbij de systeemeisen in kaart brengen voor wat betreft alle op te leveren pilots en de plaats van de modules binnen het totale pakket beschrijven.

Verder valt in de planning af te leiden dat de bouwfase van de pilot 'Vertaal' 14 weken bestrijkt, waarbij de pilotontwikkeling van de betreffende pilot ½ week bevat. Dit is gedaan omdat de bouwfase van de pilot zeer veel tijd in beslag zal nemen. Deze tijd wordt verdeeld over de hele bouw periode van dit project. U kunt hierbij denken dat er dagelijks een uurtje wordt besteedt aan de bouwfase van pilot. Hierdoor voorkom je dat er langdurig eentonig werk wordt verricht.

8.3 De gedetailleerde planning visueel



Definitiestudie

Naam document:	Definitiestudie
Bestandsnaam:	Definitiestudie_v1.doc
Project:	De ontwikkeling van de modules ‘Facturatie’ en ‘Beheer’
Datum oplevering:	19/09/2003
Omschrijving van de inhoud:	Dit document beschrijft de doelen en beperkingen van de modules ‘Facturatie’ en ‘Beheer’
In opdracht van:	Metrix B.V.
Auteur:	Ruben Maas

Voorwoord

In de fase definitiestudie worden de doelen van het systeem geanalyseerd, evenals de beperkingen ervan. Dit document is opgesteld in nauwe samenwerking met de opdrachtgever. Tijdens workshopsessies worden de systeemeisen geïnventariseerd en geprioriteerd. Vervolgens wordt een systeemconcept opgesteld.

Doelstellingen van deze fase en dit document:

- Gezamenlijke overeenstemming bereiken over het te volgen ontwikkelscenario
- De systeemeisen bepalen
- De eisen prioriteren
- Een globale oplossing in functionele termen: een systeemconcept
- De technische en organisatorische aspecten analyseren
- Het systeemconcept onderverdelen in pilots
- De fase pilotontwikkeling plannen

Inhoudsopgave

1. Inleiding 5
2. Het plan van aanpak 6
2.1 De opdracht 6
2.1.1 De probleemstelling 6
2.1.2 De doelstelling 6
2.2 Planning fase definitiestudie 6
2.2.1 Planning van de activiteiten in de fase definitiestudie per dag 7
2.2.2 De activiteiten met de uit te voeren werkzaamheden 7
3. Het ontwikkelscenario 9
3.1 De Projectkenmerken 9
3.2 De impact van het project 9
3.3 De cruciale succesfactoren 9
3.3.1 De factor geld 9
3.3.2 De factor organisatie 9
3.3.3 De factor tijd 10
3.3.4 De factor informatie 10
3.3.5 De factor kwaliteit 10
3.4 De globale pilotstrategie 10
3.4.1 De te ontwikkelen onderdelen 10
3.4.2 De prioriteit van de te ontwikkelen onderdelen 10
3.6 De globale teststrategie 10
4.Interview & Workshop 11
4.1 Doelen 11
4.1.1 Interview 11
4.1.2 Workshop 11
4.2 gewenste personen 11
4.2.1 Interview 11
4.2.2 Workshop 11
4.3 Uitgangspunten 11
4.3.1 Interview 11
4.3.2 Workshop 11
5. De Huidige situatie 12
5.1 Wat is Key-Task 12
5.2 Key-Task binnen de organisatie 13
5.2.1 Huidige situatie facturatie 13
5.2.2 Huidige situatie 'Beheren systeem-instellingen' 15
5.3 De Software architectuur van Key-Task 15
6. Systeemeisen 22
6.1 Systeemeisen module 'Facturatie' 22
6.1.1 Basissysteem-eisen 22
6.1.2 Interface-eisen 22
6.1.3 Integriteits-eisen 22
6.1.4 Performance-eisen 23
6.1.5 Operationele-eisen 23
6.2 Systeemeisen module 'Beheer' 23
6.1.1 Basissysteem-eisen 23
6.1.2 Interface-eisen 24
6.1.3 Integriteits-eisen 24
6.1.4 Performance-eisen 24
6.1.5 Operationele-eisen 24
6.3 Prioritering systeemeisen 25
7. Gewenste Situatie 26
7.1 Module 'Facturatie' 26
7.1.1 Factuur aanmaken 26

7.1.2 Factuur zoeken 27
7.1.3 Factuur Verwijderen 28
7.1.4 Factuur Verzenden 28
7.1 Module 'Beheer' 28
8. Technische Structuur 29
8.1 De technische Architectuur van de hardware 29
8.2 De technische Architectuur van de software 30
8.3 De haalbaarheid van de technische aspecten 32
8.4 De herbruikbare componenten 32
9. Organisatorische inrichting 33
9. Organisatorische inrichting 33
9.1 De werkstroom 33
9.2 De organisatorische gevolgen 33
9.3 De vereiste gebruikersdocumentatie 33
10. Pilotplan 34
10.1 De uit te voeren Pilots 34
10.1.1 Pilot Facturatie 34
10.1.2 Pilot Beheer 34
10.2 Ontwikkelscenario 34
10.2.1 De prioriteit van de te ontwikkelen onderdelen 34
10.2.2 Pilottesten 35
10.2.3 Planning 35

1. Inleiding

Binnen dit document worden de doelen en de beperkingen van de te ontwikkelen systeem geanalyseerd. Dit is volbracht in combinatie met de ontwikkelaar, opdrachtgever en gebruikers.

Na het lezen van dit document is voor de lezer mede duidelijk hoe de ontwikkeling van de Key-Task modules worden aangepakt en aan welke eisen het systeem moet voldoen. Op basis van gesprekken, die ik zowel met de gebruiker als met de opdrachtgever heb gehad, heb ik een lijst van systeemeisen kunnen samenstellen en de huidige situatie kunnen beschrijven. Op basis hiervan heb ik een systeemconcept en een technische architectuur geschreven.

Tenslotte bevat dit document een pilotplan waarbij er verder invulling wordt gegeven en het verdere verloop van het project.

2. Het plan van aanpak

Het doel van dit hoofdstuk is duidelijk vast te leggen over de inhoud en de omvang van de op te leveren producten en de activiteiten. De meeste informatie uit dit hoofdstuk is

Allereerst is het binnen de huidige situatie niet mogelijk het systeem te beheren. Het systeem beschikt over instellingen, die invloed uitoefenen op de werking van het systeem. Binnen de huidige situatie kunnen die instellingen door de eindgebruiker niet worden aangepast. Tevens is er geen overzicht in de instellingen die het systeem gebruikt. Gezien dit feit kan het Key-Task systeem niet volledig worden afgericht op de betreffende organisatie.

afkomstig van het globale plan van aanpak, enkele zaken zijn hierin echter gedetailleerder uitgewerkt.

2.1 De opdracht

2.1.1 De probleemstelling

Het Key-Task systeem is vooralsnog nog niet geheel bruikbaar voor organisaties binnen het MKB. Er zijn een aantal problemen die een succesvolle implementatie van het systeem in de weg staan.

Tevens blijkt, uit de mening van mogelijke klanten uit de doelgroep, dat het systeem beperkt is in het gebruik, omdat het verrekeningsproces ontbreekt. Het gevolg hiervan is dat gebruikers in de meeste gevallen moeten terugvallen op het handmatig samenstellen van facturen. Buiten het feit dat dit probleem een dergelijke organisatie veel tijd en geld kost, is er geen overzicht welke kosten er reeds zijn verrekend.

2.1.2 De doelstelling

De uitvoerder moet ervoor zorgen dat het Key-Task systeem succesvol kan worden geïmplementeerd binnen organisaties van het MKB. Hiervoor dienen de volgende toevoegingen te worden gedaan aan het systeem:

1. ***Beheer-module*** Binnen dit systeemdeel moet de systeeminstellingen kunnen worden gepresenteerd en worden gewijzigd door de gebruiker.

2. ***Facturatie-module*** Binnen dit systeemdeel moet de gebruiker de mogelijkheid hebben de gemaakte kosten voor een activiteit aan de klant te verrekenen.

Bij de ontwikkeling van de bovenstaande modules dienen de integriteits, interface, performance en operationele-eisen, die aan Key-Task zijn gekoppeld, te worden gehandhaafd. Tevens dienen de usability aspecten van Key-Task te worden voortgezet.

2.2 Planning fase definitiestudie

De fase definitiestudie bestaat uit ongeveer 2,5 werkweken (12 werkdagen), zoals beschreven in het globale plan van aanpak en bestaat uit een aantal activiteiten die uitvloeien in resultaten. In deze paragraaf vindt men een opsomming van deze zaken.

2.2.1 Planning van de activiteiten in de fase definitiestudie per dag

Activiteit per dag	1	2	3	4	5	6	7	8	9	10	11	12
Stel plan van aanpak op												
Bereid interview & workshop voor												
Interview & workshop houden												
Evalueer interview & workshop												
Beschouw ontwikkelscenario												
Huidige situatie in kaart brengen												
Defineer systeemeisen												
Bepaal systeemconcept												
Beschouw technische structuur												
Beschouw organisatorische inrichting												
Stel pilotplan op												
Uitloop												

2.2.2 De activiteiten met de uit te voeren werkzaamheden

Plan van aanpak opstellen
- planning maken

interview & workshop voorbereiden
- Opzetten accommodatie/administratie
- Uitnodigen deelnemers
- Evaluatierapport opstellen

interview & workshop houden
- het houden van het interview

interview & workshop evalueren
- Evaluatie van het interview en de workshop

Huidige Situatie in kaart brengen
- Interview antwoorden onderzoeken
- Verschillende overzichtsfiguren maken van de organisatie & software

Systeemeisen definiëren
- Reikwijdte bevestigen
- Basissysteemeisen definiëren
- Interface eisen definiëren
- Integriteitseisen definiëren
- Performance eisen definiëren
- Operationele eisen definiëren

Systeemconcept bepalen
- Globale actoren specificeren
- Globale events specificeren
- Globaal procesmodel specificeren
- Operationele kenmerken beschrijven

Technische structuur beschouwen
- Technische architectuur beschouwen
- Technologische basis beschouwen
- Systeemconfiguratie beschouwen
- Hebruikbare componenten vaststellen

Organisatorische structuur beschouwen

- Onderzoek doen naar de organisatorische inrichting

Pilotplan opstellen

- Pilotstructuur definiëren
- Pilotontwikkeling structureren
- Pilots priortieren

3. Het ontwikkelscenario

Het hoofddoel van deze activiteit is de reikwijdte en de context van het ontwikkeltraject vast te leggen of te bevestigen en ervoor te zorgen dat de iteratieve ontwikkelstrategie op de juiste wijze gebruikt wordt.

3.1 De Projectkenmerken

In deze paragraaf vinden we een uiteenzetting van de reikwijdte en de beperkingen van het project.

In een periode van totaal 19 weken worden een tweetal modules ontwikkeld met de volgende kenmerken:

1. Het project beschikt over een uitgebreide systeemdocumentatie van het huidige Key-Task systeem.
2. De pilots kunnen beschikken over meerdere iteraties. Na elke iteratie wordt de pilot onderworpen aan een gebruikerstest. Indien er veranderingen of aanpassingen gewenst zijn dan zullen deze worden verwerkt binnen de daaropvolgende iteratie.

De beperkingen in dit project zijn:

1. Het project heeft in totaal een looptijd van 19 weken.
2. Voor dit project is geen budget beschikbaar. Er kan dus geen gebruik worden gemaakt van externe componenten of arbeidskrachten.

3.2 De impact van het project

Dit project heeft een aantal invloeden op een organisatie. Indien een organisatie gebruik zal maken van de Key-Task applicatie zullen ze zelf in staat zijn het systeem naar hun verlangen in te stellen. Ze hoeven daarom niet meer contact te maken met het distribuerende bedrijf. Dit heeft voor beide partijen een reducerende werking op zowel de kosten als de tijd wanneer de aanpassing geschiedt.

Tevens moet de organisatie die gebruik maakt van de Key-Task applicatie zich inpassen aan de wijze waarop de facturatie binnen Key-Task geschiedt. Binnen het Key-Task zijn systeeminstellingen opgenomen waarmee het facturatieproces kan worden aangepast aan de betreffende organisatie. Indien dit niet het geval kan zijn dan dient de klant zijn facturatieproces aan te passen.

3.3 De cruciale succesfactoren

In deze paragraaf zullen de cruciale succesfactoren ook wel **GOTIK** factoren worden besproken. Dit is om een beter inzicht te krijgen in de aspecten zoals, **G**eld, **O**rganisatie, **T**ijd, **I**nformatie en **K**waliteit.

3.3.1 De factor geld

Voor dit project is tot dusver geen budget beschikbaar.

3.3.2 De factor organisatie

Organisatie van dit project is in grote mate afhankelijk van een goede planning, te vinden in dit plan van aanpak. De belangrijkste zaken die verder in de gaten moeten worden gehouden zijn:

- Op tijd leveren van zaken die de onderwijsinstelling wil zien
- Producten voor de opdrachtgever wil zien

3.3.3 De factor tijd

Het project beslaat 25 werkdagen in de op de volgende pagina afgedrukte planning staan deze 50 werkdagen met de globale activiteiten in. Terugkomdagen van school worden volgens de regels van deze instelling als werkdag.

3.3.4 De factor informatie

Informatie voor de opdracht wordt verkregen uit/van

1. Eigen literatuur
2. Literatuur van het bedrijf
3. Internet
4. Feedback van Gebruikers, Metrix BV en haar medewerkers
5. Eigen inbreng

3.3.5 De factor kwaliteit

Voor de op te leveren software is de kwaliteit voldoende indien deze aan de gestelde eisen voldoet, waarbij voor sommige eisen gebruikerstesten noodzakelijk zijn om de betreffende eisen te verifiëren.

3.4 De globale pilotstrategie

In deze paragraaf wordt omschreven hoe de pilots te ontwikkelen.

3.4.1 De te ontwikkelen onderdelen

Bij de ontwikkeling van de modules zijn de volgende twee pilots te onderscheiden:

- Pilot 'Facturatie'
- Pilot 'Beheer'

3.4.2 De prioriteit van de te ontwikkelen onderdelen

Voor wat betreft de prioriteit van de te ontwikkelen modules geef ik aan de pilot 'Facturatie' een hoge prioriteit en aan de pilot 'Beheer' een lage prioriteit.

Ik heb hiertoe besloten omat de pilot 'Facturatie' een essentieel onderdeel is van het systeem. Doch de pilot beheer ook een essentiële functie kunnen we in dat geval kiezen voor eenvoudige snelle manieren om de instellingen aan het systeem mee te geven.

3.6 De globale teststrategie

De pilots worden getest totdat het gewenste resultaat is geboekt. Na elke bouwfase zal een test plaats vinden aan de hand van een gebruikerstest. Eventuele gebreken of fouten worden in een iteratie hersteld. Hierna zal de pilot wederom worden getest waarna zich het proces herhaald.

4.Interview & Workshop

Het houden van interviews/workshops dienen ervoor om- met een groep betrokkenen een zo hoog mogelijk input van informatie te genereren die nodig is om de applicatie te maken. In de workshop kan men aangeven wat men belangrijk en minder belangrijk vind in het te bouwen systeem. De workshops zijn voor het opstellen van de huidige situatie, systeemeisen, het beschouwen van en de organisatieaspecten. Het interview daarentegen wordt gebruikt om specifieke systeemeisen te achterhalen.

4.1 Doelen

4.1.1 Interview

Informatie vergaren met betrekking tot het opstellen van de huidige situatie, systeemeisen en de organisatieaspecten.

4.1.2 Workshop

Informatie vergaren met betrekking tot het opstellen van de systeemeisen en de technische aspecten.

4.2 gewenste personen

4.2.1 Interview

Opdrachtgever

4.2.2 Workshop

Vertegenwoordigers van Darumath.

4.3 Uitgangspunten

4.3.1 Interview

De geïnterviewde wordt onderworpen aan de gestelde vragen van de interviewer. Het interview zal voornamelijk bestaan uit openvragen.

4.3.2 Workshop

De betrokkenen krijgen een uitnodiging en inzicht in de tot nu toe opgeleverde documentatie/producten om een goed beeld te krijgen van de situatie tot nu toe. De workshop wordt geleid door de uitvoerder van dit project, maar laat altijd ruimte aan de deelnemers van de workshop om vragen te stellen en om te discussereen.

5. De Huidige situatie

Om een verandering door te kunnen voeren is het belangrijk om eerst de huidige situatie in kaart te brengen. Dit doe ik op twee verschillende vlakken. Allereerst geef ik een algemeen inzicht op een organisatie uit het midden -en kleinbedrijf, die ter ondersteuning van hun service activiteiten gebruik maakt van Key-Task.Ik richt me hier op een willekeurig bedrijf die binnen de doelgroep van het Key-Task systeem valt. Dit bedrijf ,genaamd Darumath, heb ik benaderd met het verzoek het Key-Task systeem in gebruik te nemen om zo de behoeftes van de gebruiker te definiëren en eventueel te integreren in het systeem.

Tevens neem ik van de huidige situatie gebruik om u een inzicht te verschaffen in het huidige objectmodel van de Key-Task applicatie. Doordat de te ontwikkelen modules onderdeel uitmaakt van de Key-Task applicatie is het van uiterst belang te weten hoe de huidige Klassenbibliotheek in elkaar steekt.

5.1 Wat is Key-Task

Key-Task is een service management systeem voor het midden- en kleinbedrijf. Een service management systeem geeft ondersteuning aan de service activiteiten van een organisatie doormiddel van de Informatie Technologie.

Het Key-Task Systeem springt in op de volgende punten

- **Klantbeheer**
  Heeft als doel alle klanten van een organisatie te beheren waaronder de locaties van een klant, de contactpersonen en de producten die de klant in zijn bezit heeft.
- **Taakbeheer**
  Heeft als doel de middelen van een organisatie, deze zijn veelal in de vorm van bedrijfsmedewerkers beschikbaar, te koppelen aan de taken die moeten worden uitgevoerd. Taken kunnen worden gekoppeld aan een klant of ze kunnen worden gekoppeld aan de betreffende organisatie.
- **Onkostenbeheer**
  Heeft als doel de gemaakte kosten van een organisatie te beheren zodat ze kunnen worden verantwoord en doorberekend. Deze onkosten ontstaan op het moment dat er activiteiten worden uitgevoerd. Voorbeelden van onkosten zijn: reiskosten, materiaalkosten etc…

Doormiddel van de bovenstaande elementen kunnen de service activiteiten van een organisatie worden beheerd. Het gaat hierbij om dat de activiteiten van een organisatie kunnen worden verantwoord. Dat wil zeggen dat het systeem kan vertellen waar het probleem zich opspeelt, wie zich daar mee bezig houdt, wanneer dit gaat gebeuren en welke kosten hieraan gerelateerd zijn.

Key-Task is binnen de huidige situatie een webapplicatie, dat wil zeggen dat het systeem zijn presentatie vindt in een web-browser. In de loop van dit hoofdstuk wordt nog uitgebreid ingegaan op de technische aspecten van Key-Task.

Wilt u precies weten hoe Key-Task eruit ziet verzoek ik u om de volgende url te bezoeken:

http://213.84.99.144/keytask
Domein: General-Demo
Username: admin
Password: admin

5.2 Key-Task binnen de organisatie

Het probleem zoals deze staat beschreven in de probleemstelling, waarvoor ik u moet verwijzen naar de opdrachtomschrijving, heeft te maken met het feit dat het systeem op een tweetal plaatsen tekort schiet.

Facturatie blijkt een onmisbaar onderdeel te zijn voor een dergelijk systeem. Het is mogelijk voor iedere activiteit de gemaakte kosten te laten bijhouden. Uiteindelijk zal een organisatie een rekening sturen naar de betreffende klant om zodoende de gemaakte kosten te verrekenen. Binnen de huidige situatie zouden ze dit laatste moeten oplossen met een externe tool. Wat wil zeggen dat in veel gevallen arbeidsintensief werk aan te pas komt. De secundaire kosten van een dergelijk systeem kunnen hierdoor erg oplopen.

Het tweede punt waarop het huidige systeem te kort schiet is het niet kunnen beheren van de systeeminstelling. Het systeem wordt beïnvloed door allerlei instellingen. Deze instellingen worden bewaard in een database. Voor het wijzigen van deze gegevens dient de applicatiebeheerder vele handelingen te doen door de database handmatig te openen en de betreffende tabel te openen om zo eventuele bewerkingen uit te voeren. Buiten het feit dat dit niet geheel een gebruiksvriendelijke oplossing is, speelt het probleem dat de gegevens niet worden gecontroleerd of ze geldig zijn of dat ze in de juiste grootheden worden ingevoerd. Ongeldige systeeminstellingen kunnen leiden tot uitvallen of vertragen van het systeem.

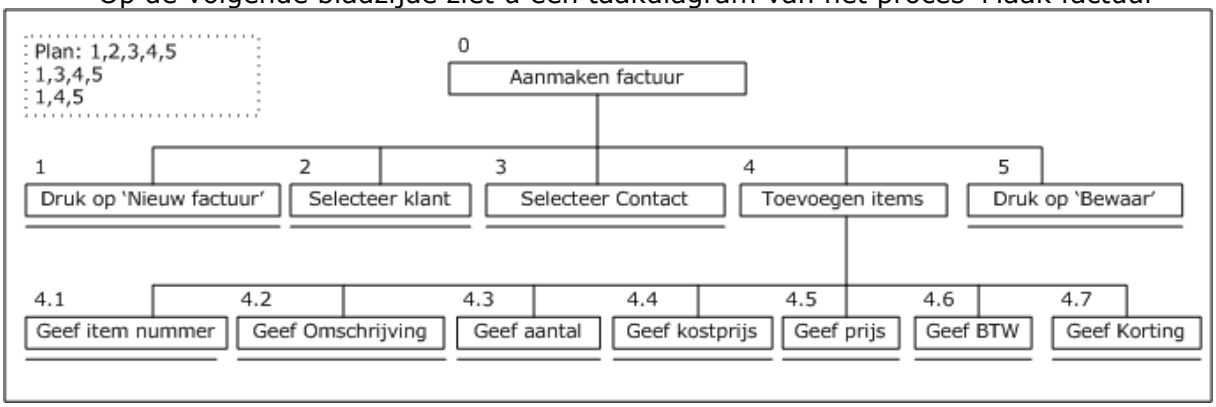
Ik zal deze twee beschreven problemen nader beschrijven binnen dit hoofdstuk. Allereerst maak ik van zowel de facturatie als zijnde het beheren van de systeeminstellingen een schets maken hoe het binnen de huidige situatie verloopt.
Tenslotte zal ik van de mogelijkheid gebruik maken om het huidige systeem te beschrijven aan de hand van een objectmodel. Mede aan hand van de gegevens binnen dit hoofdstuk zal ik in staat zijn een systeemconcept te beschouwen.

5.2.1 Huidige situatie facturatie

Binnen deze paragraaf zal nader worden beschreven hoe de facturatie binnen de huidige situatie verloopt. We bekijken een tweetal situaties, namelijk:

- Factuur aanmaken *Welke stappen dienen er worden uitgevoerd voor het maken van een factuur.*
- Factuur zoeken *Welke stappen dient de gebruiker te verrichten om achteraf, aan de hand van een activiteitnr, klantnr, taaknr of onkostennr een factuur te bekijken.*

Op de volgende bladzijde ziet u een taakdiagram van het proces 'Maak factuur'



Figuur 5.1:Proces aanmaken factuur

Indien een gebruiker een factuur wil maken van een bepaalde activiteit moet hij er eerst achter zien te komen welke kosten daar aan zijn gekoppeld. Als kosten beschouw ik de tijd die een medewerker heeft besteedt en de onkosten, zoals materialen, die ervoor zijn gebruikt. Hiervoor dient de gebruiker de betreffende activiteit op te zoeken aan de hand van een aantal voor de hand liggende criteria's:

- Klantnummer
- Activiteitsnr
- Uitvoerder
- Team
- Taaknr
- Onkostennr

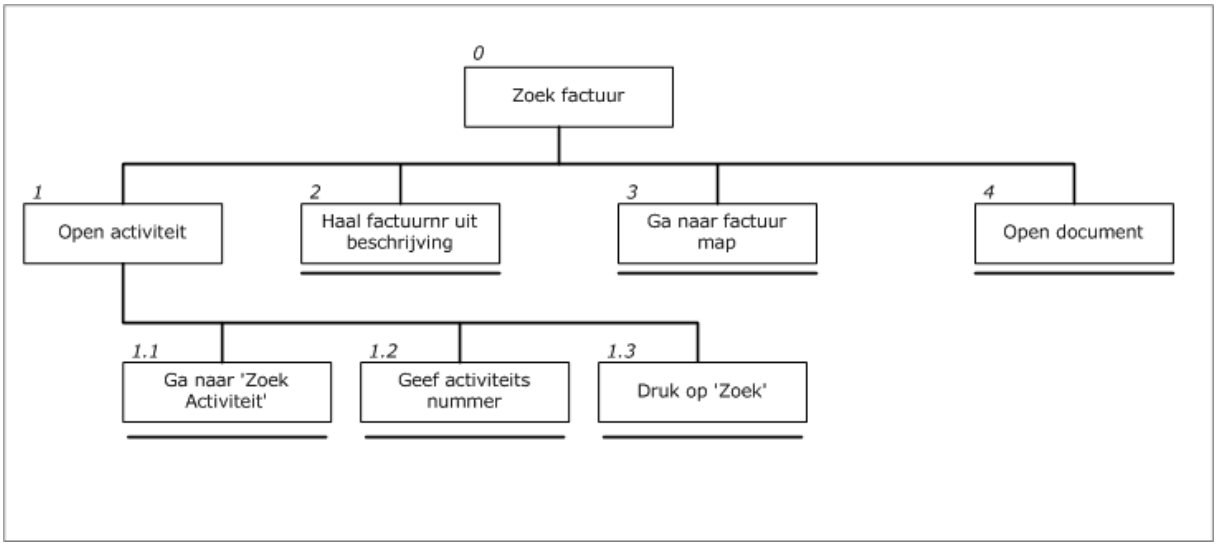
Er zijn echter meerdere criteria maar die zijn buiten beschouwing gelaten, omdat deze verder niet relevant zijn.

Wanneer de gebruiker de juiste activiteit heeft gevonden kan hij/zij specifiekere informatie aanschouwen door op de activiteit te klikken. Eenmaal in het activiteitsscherm is allerlei informatie beschikbaar over de activiteit zelf, maar ook ziet u een opsomming van de kosten die daar aan zijn gerelateerd.

Het is aan de gebruiker de taak de kosten te analyseren om te bepalen of ze in rekening moeten worden gebracht al dan niet met een marge.

De volgende stap voor de gebruiker is een Microsoft word template te openen, waarna hij de betreffende items kan bijvoegen in het document. Per item dient te worden meegegeven het itemnr, beschrijving, aantal, BTW percentage, korting, en prijs.

Indien de benodigde items zijn beschreven binnen de factuur kan deze vervolgens worden uitgeprint.
De volgende figuur laat de taakdiagram 'Zoek factuur' zien. Dit proces wordt ingezet op het moment dat er na tijd van facturering informatie opgevraagd wordt over een factuur aan de hand van een activiteit.



Figuur 5.2:Proces zoek factuur

Allereerst dient de gebruiker de betreffende activiteit op te zoeken aan de hand van een aantal criteria. Als hij de juiste activiteit heeft gevonden dient hij op die activiteit te klikken om zo gedetailleerde gegevens van de activiteit te bekijken. Zo ook de beschrijving. Hieruit kan worden herleid, indien deze gefactureerd is, welk factuurnr hieraan gekoppeld is. Het factuurnummer staat tevens voor het

word document. Deze kan geopend worden en de gebruiker ziet de betreffende factuur op het scherm.

Geconcludeerd kan worden na het beschrijven van deze twee processen dat er voor de gebruiker handmatige arbeid aan te pas komt welke in staat is geautomatiseerd te worden, zodat de processen vlotter worden uitgevoerd en de arbeidskosten worden gereduceerd.

5.2.2 Huidige situatie 'Beheren systeem-instellingen'

Binnen de huidige situatie beschouw ik een tweetal processen met betrekking tot het beheer van de systeeminstellingen. Allereerst het proces 'Bewerk systeeminstellingen' en als tweede 'Bekijk systeeminstellingen'. Beide processen worden niet nauwelijks ondersteund. Hierdoor is het irrelevant deze processen aan de hand van een taakdiagram te beschrijven. Ik zal dan ook van de gelegenheid gebruik maken om de twee processen tekstueel te beschrijven.

Binnen de huidige situatie van Key-Task onderscheid ik vier soorten systeeminstellingen, namelijk:

- META-gegevens
- Systeemvariabelen
- Foutmeldingen
- Gebruikers

META-gegevens houdt in: aanvullende informatie omtrent een object die door de applicatiebeheerder te bewerken en toe te voegen zijn. U kunt hierbij denken aan projectstatussen, btw, lijncodes, taakstatussen, klanttypes, etc… In de meeste gevallen worden deze gegevens afgebeeld in een dropdown listbox zodat ze door de gebruiker geselecteerd kunnen worden.

Systeemvariabelen zijn waardes die werking van het systeem direct beïnvloeden. U kunt hierbij denken aan:

- Sessietimeout een variabele die aangeeft in secondes wanneer een sessie afloopt.
- Maxresults Een variabele die aangeeft hoeveel resultaten een zoekprocedure mag teruggeven

Dergelijke systeeminstellingen hebben kunnen tevens invloed uitoefenen op de performance van het systeem.

Foutmeldingen zijn binnen het Key-Task systeem enkel te bewerken. Dit betekent dat er geen nieuwe foutmeldingen kunnen worden toegevoegd. Een foutmelding bestaat over een code, beschrijving en een oplossing. Wanneer het systeem een fout oproept dan kan de beschrijving en oplossing door de applicatiebeheerder worden aangepast.

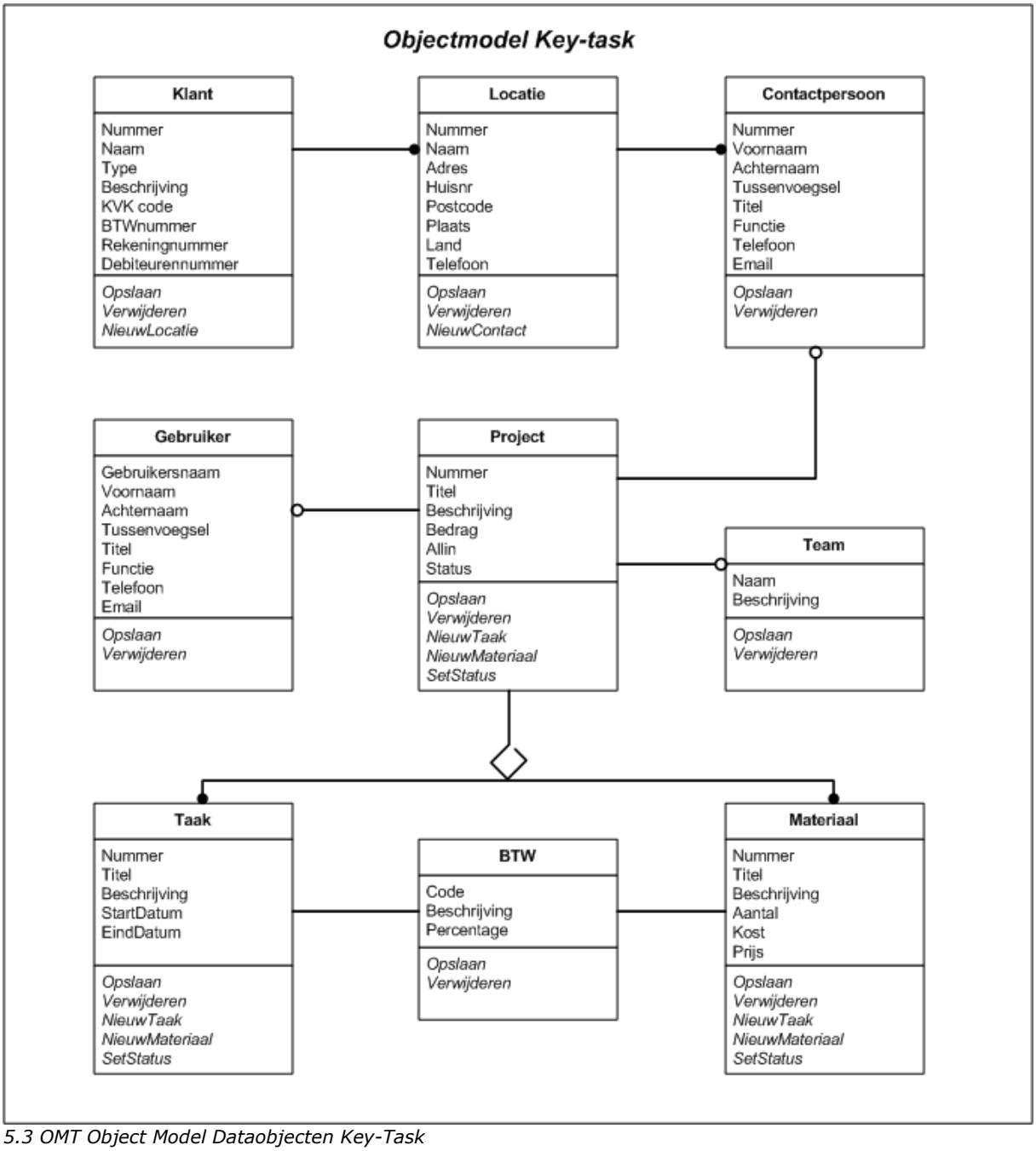
Tenslotte moeten de gebruikers kunnen worden aangemaakt. Doch dit al reeds staat verwerkt in de huidige klassenbibliotheek van Key-Task is hiervoor tot op heden geen interface voor zodat de gebruiker hier nog geen gebruik van kan maken.

5.3 De Software architectuur van Key-Task

Deze paragraaf heeft als doel een beeld te scheppen over hoe de software, van Key-Task, in elkaar steekt. Dit is van groot belang omdat we op deze wijze goed kunnen bepalen waar we de koppeling moeten bouwen.

Binnen deze paragraaf zullen we ingaan op de beschikbare objecten van Key-Task en bekijken we hoe Key-Task is opgebouwd. De objecten die in relatie staan met ons

probleem worden zonodig specifieker uitgewerkt. Allereerst zal ik u een beeld schaffen over de beschikbare dataobjecten van Key-Task.



5.3 OMT Object Model Dataobjecten Key-Task

De bovenstaande figuur laat het Key-Task OMT object model zien. In dit model zitten de dataobjecten van Key-Task. Binnen Key-Task wordt onderscheidt gemaakt tussen data- en business objecten. De dataobjecten representeren de gegevens uit de database. De business objecten daarentegen worden gebruikt om de dataobjecten te benaderen. Voor wat betreft de functionele objecten moet ik u door verwijzen naar verderop deze paragraaf.

In Figuur 5.3 ziet u dat een Customer bestaat uit minimaal een Address object. Een Address bevat informatie over de vestiging van een klant. Een klant kan dus meerdere vestigingen bevatten. Een vestiging bevat op haar beurt weer uit contactpersonen, activiteiten en producten.

Contactpersonen zijn mensen die werkzaam zijn bij de betreffende klant. De contactpersonen worden bijgehouden om de zakelijke relaties te beheren.

Activiteiten zijn gebeurtenissen die zijn ontstaan bij een klant en waaraan een taak of materiaal gekoppeld moet worden. Een activiteit is eigenlijk een verzameling van allerlei taken en materialen. Een activiteit bestaat op haar beurt weer uit:

- Een Medewerker Degene die de activiteit beheert.
- Een Taak Uit te voeren werkzaamheden door de medewerker
- Een Materiaal Kosten die in relatie staan bij een activiteit.

Buiten de dataobject bevat Key-Task ook een aantal Functionele objecten. Zoals we reeds hebben beschreven dienen deze objecten ter representatie van de data objecten. We zullen dit proberen te verduidelijken aan de hand van figuur 5.4 De figuur verschaft een kijk op de wijze waarop de functionele objecten de data objecten representeert.

Functioneel Object Key-Task		
Procedure name	**Parameters**	**Resultaat**
getProjectByID	ID (Long)	Project object
getTaskByID	ID (Long)	Taak object
getEmployeeByID	ID (String)	Employee Object
getOpenTasks	-	Verzameling Taak objecten
getRecentProject	-	Verzameling Project Objecten

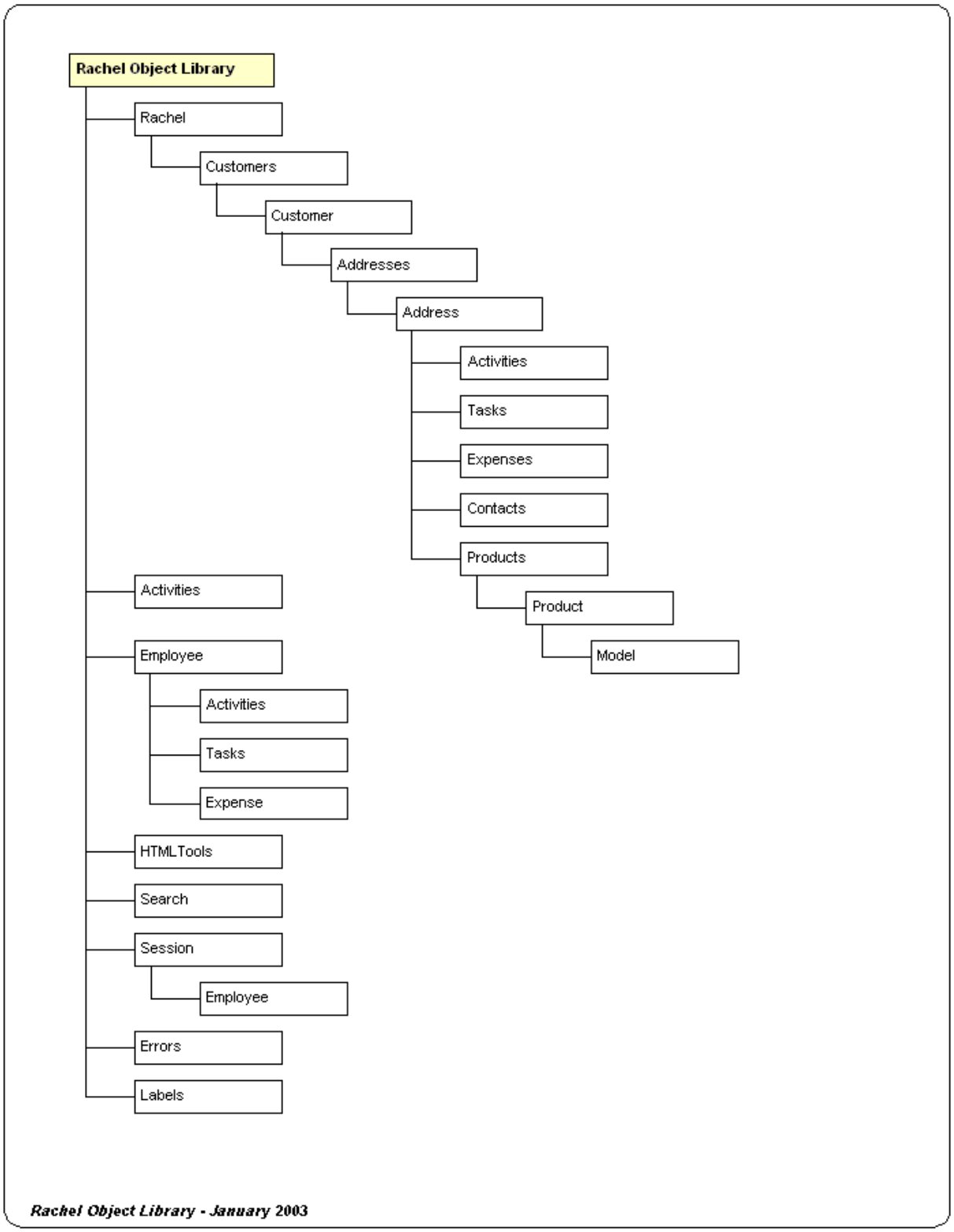
Figuur 5.4

Ter verduidelijking: De bovenstaande figuur bevat niet alle functionele objecten van Key-Task en toont ook niet alle bestaande procedures. De afbeelding is slechts een gedeelte van de beschikbare functionele objecten van Key-Task. Indien u geïnteresseerd bent in het totale object model of systeem specifieke kenmerken, die niet zijn gerelateerd aan het probleem, moeten wij u verwijzen naar de systeem documentatie van Key-Task.

Het Key-Task object bevat een aantal procedures die de dataobjecten kan opvragen. Deze procedures kunnen verzamelingen, ook wel collecties genoemd, en enkelvoudige dataobjecten teruggeven.

U heeft nu een goed beeld kunnen scheppen over de beschikbare data objecten binnen Key-Task. Tevens heeft u daarbij kunnen waarnemen hoe deze vervolgens door de functionele objecten worden geraadpleegd.

Op de volgende pagina ziet u *figuur 5.5* afgebeeld, deze geeft een verdere invulling van het centrale functioneel object 'Key-Task'. Dit object kan doormiddel van eigenschappen en collecties de dataobjecten presenteren aan de bovenliggende laag. De manier van modeleren wordt niet specifiek benaamd als het zijnde een techniek, echter wordt het door Microsoft gebruikt als standaard manier van object modelling. Wij gebruiken echter deze manier van modelleren omdat dit zeer eenvoudig te hanteren en te interpreteren is.



Figuur 5.5: Key-Task Klassenbibliotheek

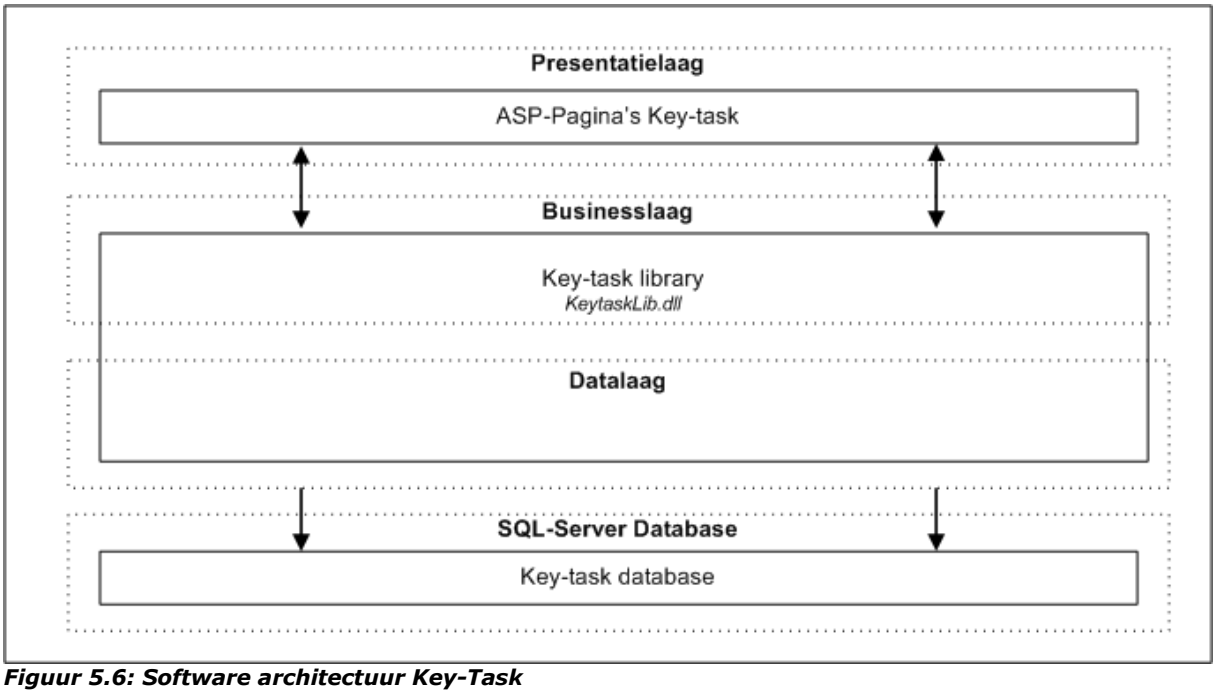
Kijkend naar figuur 5.5 valt de relaties van de objecten van Key-Task te herleiden. Dit is hebben we nodig bij het ontwerp van de Key-Task Task Manager, omdat we op basis van de huidige situatie en de systeemeisen moeten gaan bepalen welke objecten we gaan gebruiken en welke objecten daarmee weer in relatie staan.

Bovenaan ziet u de Key-Task Klassenbibliotheek, dit is de zogenaamde Klasse Bibliotheek. Deze klasse bibliotheek is een verzameling van de modules en klasses die op

hun beurt weer met elkaar kunnen werken. Deze klasse bibliotheek vormt zich in de vorm van een DLL bestand. Vanuit de Windows omgeving is het vervolgens mogelijk referenties te leggen naar de klasse bibliotheek, om vervolgens de functionaliteit daarin te raadplegen.

De klasse Key-Task vormt daarin een belangrijke rol. Deze klasse vormt het aanspreekpunt van de klasse bibliotheek, waarmee de overige klassen kunnen worden aangesproken. Zo bevat de klasse Key-Task het de eigenschap Customers. Dit is een verzameling van alle klanten die de betreffende organisatie heeft. Dit is een voorbeeld van het aanspreken van dataobjecten, echter kan vanuit de klasse Key-Task ook nog functionele objecten worden aangesproken.
Zo bevat de klasse Key-Task op haar beurt weer uit het object HTMLTools. Dit object bevat procedures ter ondersteuning van de opmaak en inhoud van een internet pagina. De procedures binnen dit object geven HTML broncodes terug van onder meer dropdownlist boxes. Stel je wil vanuit een internet pagina alle klanten in een drop down list box hebben dan kun je een van de procedures uit het object HTMLTools aanspreken. Dit maakt het aanzienlijk makkelijker voor de ontwikkelaar van de presentatie.

Tot nog toe hebben we een beeld gegeven van de objecten die Key-Task bezit.We zullen nu ingaan op de architectuur van de software van Key-Task. We brengen in kaart hoe de software momenteel in elkaar steekt. In de volgende afbeelding ziet u een weergave van de software architectuur.



Figuur 5.6: Software architectuur Key-Task

De figuur laat zien dat het Key-Task systeem is opgebouwd uit drie verschillende lagen. Elke laag heeft op haar beurt weer zijn eigen functie. De database beschouwen wij niet als een aparte laag.

De datalaag en de businesslaag vormen de zogehete Klassenbibliotheek (Klassen bibliotheek). Zoals u reeds in dit hoofdstuk heeft kunnen lezen wordt deze gevormd door een DLL-bestand.

We zullen nu alle verschillende lagen afzonderlijk bespreken:

De datalaag

De datalaag heeft als primair doel de integriteit van de database te behouden. U kunt het zien als allerlei procedures die worden uitgevoerd op het moment dat er database transacties worden uitgevoerd. Gezegd kan worden dat de datalaag de functie overneemt van de zogehete stored-procedures van een DBMS (Database Management Systeem). Het voordeel van het gebruik van een datalaag daarentegen is dat je niet afhankelijk bent een specifiek DBMS, maar dat je streeft naar database onafhankelijkheid. Je zou in dit geval gebruik kunnen maken van zowel een Oracle als een MS-SQL database.

De bussinesslaag

Deze laag bevat de eigenlijke functionaliteit van het Key-Task systeem. Hierbij valt te denken aan een tal van klassen die procedures bevatten die enkelvoudige of verzamelingen van de dataobjecten. Vervolgens kan de presentatielaag de inhoud van de objecten presenteren aan de gebruiker. De business laag is het aanspreekpunt van de Key-Task Klassenbibliotheek, waarbij geldt dat de klasse Key-Task als hoofd klasse fungeert. Vanuit deze klasse kunnen de overige objecten benaderd worden. Het moge duidelijk zijn dat de procedures uit de datalaag niet van buitenaf de Klassenbibliotheek aan te spreken zijn. Dit is ten behoeve van de veiligheid en de integriteit van het systeem, omdat indien je toegang heb tot de datalaag je overal toegang tot heb.

De presentatielaag

Deze laag heeft als doel de eigenlijke functionaliteit zichtbaar te maken voor de gebruiker en de inhoud van de dataobjecten op dusdanige wijze te tonen aan de gebruiker, zodat deze op haar beurt de gegevens kan interpreteren en verdere acties kan nemen. De presentatielaag is opgebouwd uit een aantal ASP-pagina's. Deze bestanden bevatten een combinatie van de server-side-script ASP en de opmaak taal HTML. De verdere interactie met de gebruiker wordt verstevigd met JavaScript.

6. Systeemeisen

Het doel van deze activiteit is het opstellen en actualiseren van de geprioriteerde lijst van systeemeisen. Deze definitie van de systeemeisen moet een juiste weergave zijn van de behoeften van de toekomstige gebruikers. De lijst met systeemeisen vormt de basis voor alle ontwerpactiviteiten. Bovendien zal er bij het definiëren van pilots gebruik gemaakt worden. De systeemeisen staan per item geprioriteerd op belangrijkheid. Waarbij de belangrijkste eis bovenaan staat.

6.1 Systeemeisen module 'Facturatie'

6.1.1 Basissysteem-eisen

	Basissysteem-eisen
1.	De module moet een factuur kunnen genereren op basis van een project,taak of materiaal.
2.	De module moet een factuuritem kunnen toevoegen aan een niet voltooide factuur op basis van een gemaakte onkosten of besteedde tijd.
3.	Factuuritems dienen handmatig te kunnen worden ingevoerd.
4.	De gebruiker moet de mogelijkheid hebben voor het toekennen van korting op een factuuritem.
5.	Het BTW percentage dient per factuuritem te worden aangegeven.
6.	De module moet over een zoekscherm beschikken waarvan uit facturen kunnen worden opgezocht.
7.	De module moet beschikken over schermen waarin facturen en haar items kunnen worden beheerd.
8.	Een factuur dient digitaal te kunnen worden verstuurd naar de klant.
9.	Het moet mogelijk zijn te factureren op een reeds aangemaakte factuur.

6.1.2 Interface-eisen

	Interface-eisen
1.	Het systeem moet bestaan uit een hoofdscherm waarvan uit een overzicht wordt verschaft van de beschikbare medewerkers, de openstaande taken en een grafische weergave van de uit te voeren taken.
2.	De schermen moeten passen in een resolutie van 1024X768.
3.	De stijl van Key-Task dient gehandhaafd te worden.

6.1.3 Integriteits-eisen

	Integriteits-eisen

1.	Een activiteit mag pas worden gefactureerd indien deze beschikt over de status 'Voltooid'.
2.	Gemaakte kosten mag pas worden besteed als deze beschikt over status 'Geleverd'.
3.	Besteedde tijd mag pas gefactureerd worden op het moment dat deze beschikt over de status 'Voltooid'.
4.	Een factuur moet beschikken over een adres.
5.	Als een factuur verzonden is naar de klant, dan mogen hier geen veranderingen op plaatsvinden.
6.	Indien de factuur gekoppeld is aan een activiteit dan is het standaard factuuradres het adres waarop de activiteit heeft plaatsgevonden.
7.	Facturen die zijn verzonden mogen niet worden verwijderd.
8.	Facturen die nog niet verzonden zijn mogen niet beschikken over een factuurnummer.
9.	Een taak of materiaal mag alleen worden gefactureerd indien deze behoort tot een all-inclusive project.

6.1.4 Performance-eisen

	Performance-eisen
1.	De laadtijd van een pagina mag niet meer dan 10 seconden in beslag nemen uitgaande van een 64k ISDN verbinding.
2.	Het opzoeken van een factuur mag niet langer dan 1 seconden in beslag nemen.

6.1.5 Operationele-eisen

	Operationele-eisen
1.	De functionaliteit binnen de module moet volledig herbruikbaar zijn.
2.	De module moet zo ontworpen worden dat medewerkers van Metrix aan de hand van de geleverde documentatie het systeem kunnen onderhouden.
3.	De module moeten kunnen worden gehost vanuit Microsoft Internet Information Server.
4.	De module moet kunnen draaien in Internet Explorer 5.5 en hoger.
5.	De operationele-eisen van het huidige Key-Task systeem is ook van toepassing tot deze module

6.2 Systeemeisen module 'Beheer'

6.1.1 Basissysteem-eisen

	Basissysteem-eisen

1.	Een gebruiker moet ingelogd zijn om van de module gebruik te maken.
2.	Systeeminstellingen dienen gecategoriseerd aan de gebruiker gepresenteerd te worden.
3.	Systeemfoutmeldingen dient de gebruiker te kunnen bewerken.
4.	Systeemvariabelen dient de gebruiker te kunnen bewerken.
5.	Systeemgebruikers dient de gebruiker te kunnen bewerken, aanmaken en verwijderen.
6.	De META-gegevens dient de gebruiker te kunnen bewerken, aanmaken en verwijderen.

6.1.2 Interface-eisen

	Interface-eisen
1.	De schermen moeten passen in een resolutie van 1024X768.
2.	Voor de interface geldt dat de style-sheet van het huidige Key-Task systeem te worden aangehouden.
3.	De stijl van Key-Task dient gehandhaafd te worden.

6.1.3 Integriteits-eisen

	Integriteits-eisen
1.	Enkel gebruikers van het type "Administrator" mogen gebruik maken van de module 'Beheer'.

6.1.4 Performance-eisen

	Performance-eisen
1.	De laadtijd van een pagina mag niet meer dan 10 seconden in beslag nemen uitgaande van een 64k ISDN verbinding.
2.	De uitvoer van procedures mogen niet langer dan 1 seconden in beslag nemen.

6.1.5 Operationele-eisen

	Operationele-eisen
1.	De functionaliteit binnen de module moet volledig herbruikbaar zijn.
2.	De module moet zo ontworpen worden dat medewerkers van Metrix aan de hand van de geleverde documentatie het systeem kunnen onderhouden.
3.	De module moeten kunnen worden gehost vanuit Microsoft Internet Information Server.

4.	De module moet kunnen draaien in Internet Explorer 5.5 en hoger.
5.	De operationele-eisen van het huidige Key-Task systeem is ook van toepassing tot deze module

6.3 Prioritering systeemeisen

Ik heb een top 10 samengesteld van belangrijkste eisen.

	Eisen Top 10
1.	De module moet een factuur kunnen genereren op basis van een project,taak of materiaal.
2.	De module moet beschikken over schermen waarin facturen en haar items kunnen worden beheerd.
3.	Systeemvariabelen dient de gebruiker te kunnen bewerken.
4.	De META-gegevens dient de gebruiker te kunnen bewerken, aanmaken en verwijderen.
5.	Als een factuur verzonden is naar de klant, dan mogen hier geen veranderingen op plaatsvinden.
6.	Facturen die nog niet verzonden zijn mogen niet beschikken over een factuurnummer.
7.	Factuuritems dienen handmatig te kunnen worden ingevoerd.
8.	Een factuur dient digitaal te kunnen worden verstuurd naar de klant.
9.	Een gebruiker moet ingelogd zijn om van de module gebruik te maken.
10.	Systeemgebruikers dient de gebruiker te kunnen bewerken, aanmaken en verwijderen.

7. Gewenste Situatie

Terwijl de systeemeisen, zie hoofdstuk 6, staan voor een specificatie van de probleemstelling in 'bedrijfsterminologie', is het systeemconcept een voornamelijk beschrijvende weergave op globaal niveau van de oplossing. In het systeemconcept ligt een functionele architectuur besloten die de belangrijkste functionele componenten en hun afhankelijkheden betreft.

7.1 Module 'Facturatie'

Voor wat betreft de module 'Facturatie' wordt het facturatieproces geautomatiseerd.Dit doen op basis van een aantal taken. Al deze taken worden uitgevoerd door de actor: Medewerker

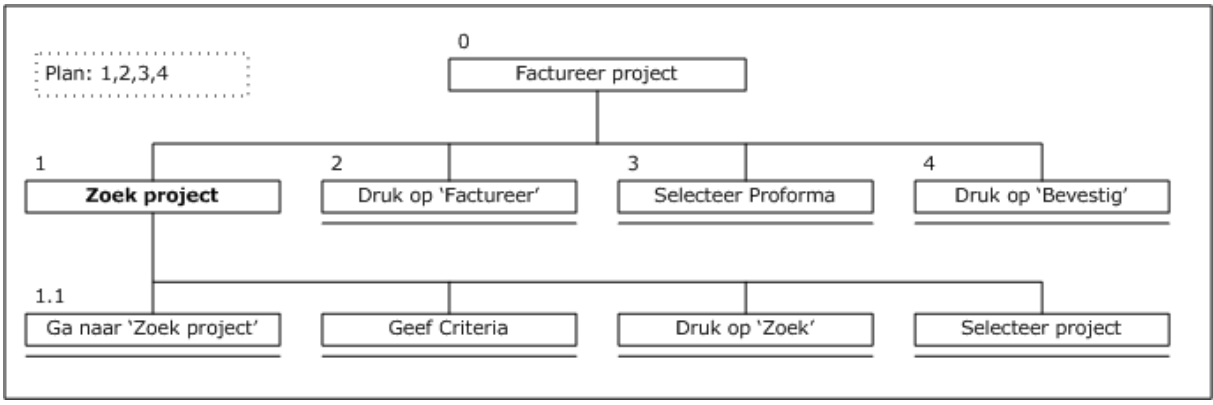
Hieronder vind u een beschrijving van taken die er worden geautomatiseerd.

7.1.1 Factuur aanmaken

Het aanmaken van facturen kan op drie verschillende manieren:

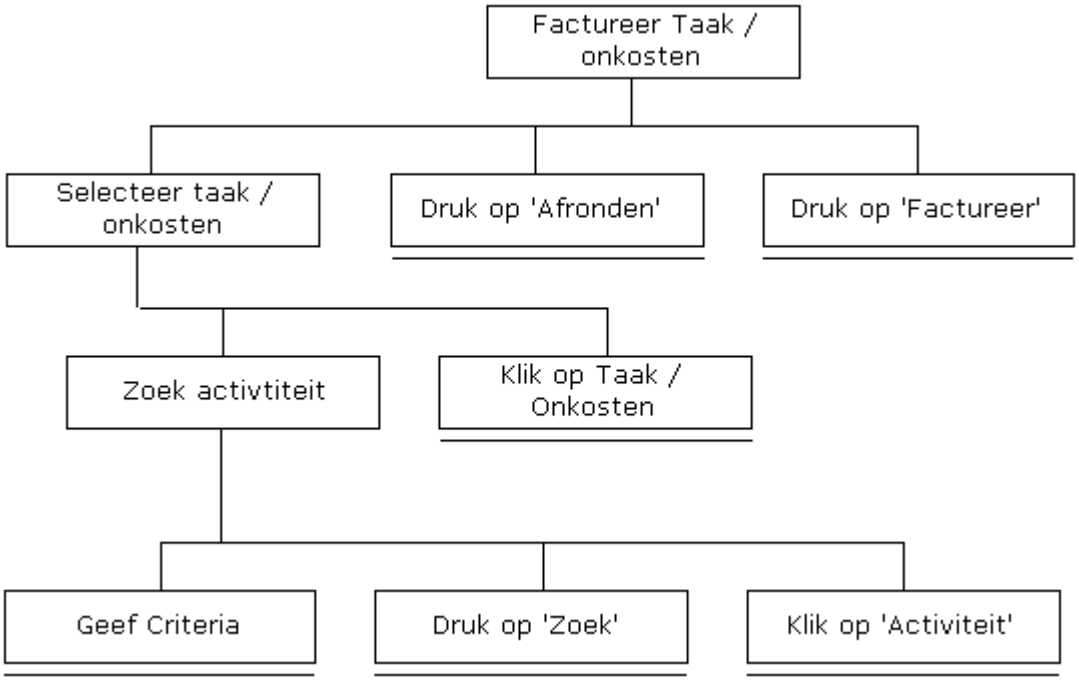
1. Factuur maken van een project

Hierbij worden alle taken en onkosten van een activiteit overgezet naar een factuur



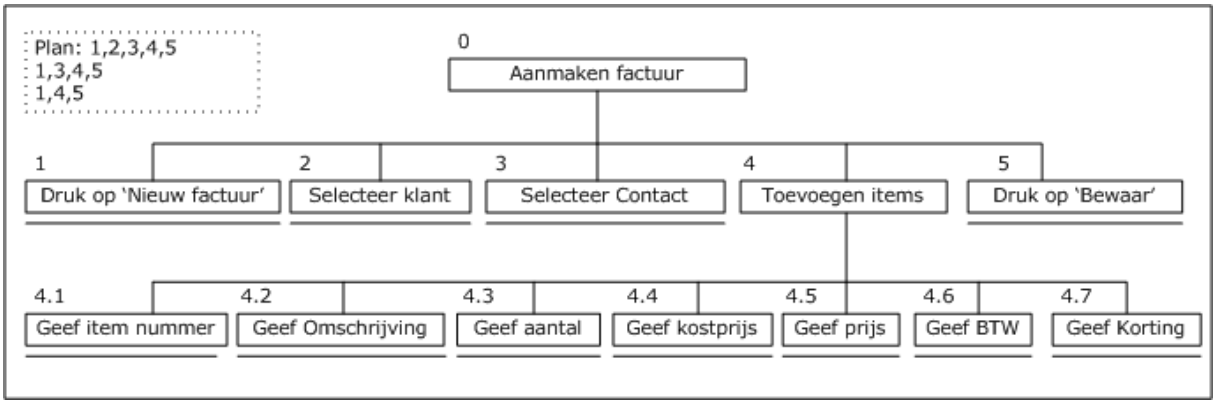
2. Factuur maken op basis van een taak of onkosten

Hierbij wordt een enkele taak of onkosten naar een factuur geplaatst.

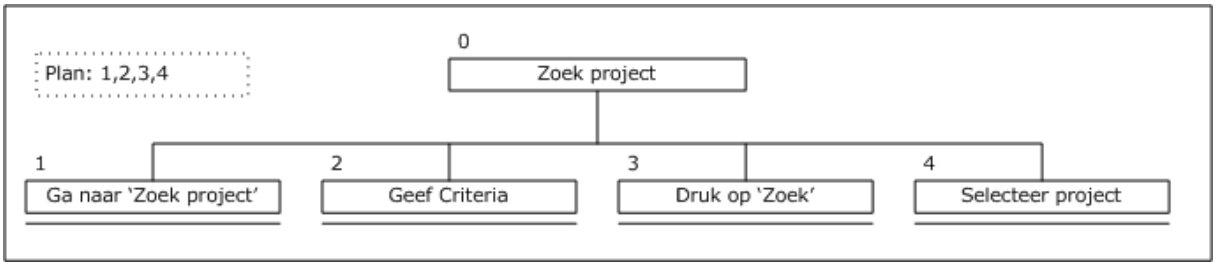


3. Handmatige factuur aanmaken

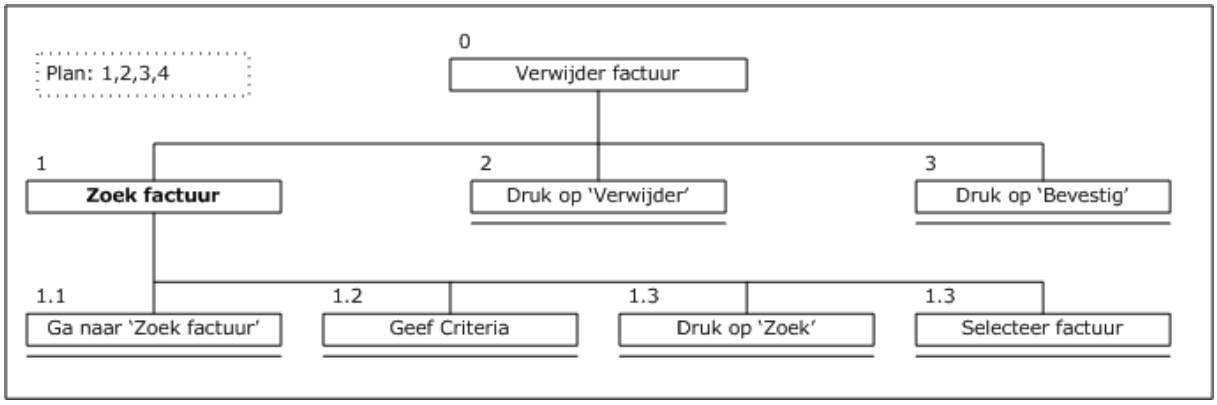
Het handmatig samenstellen van een factuur



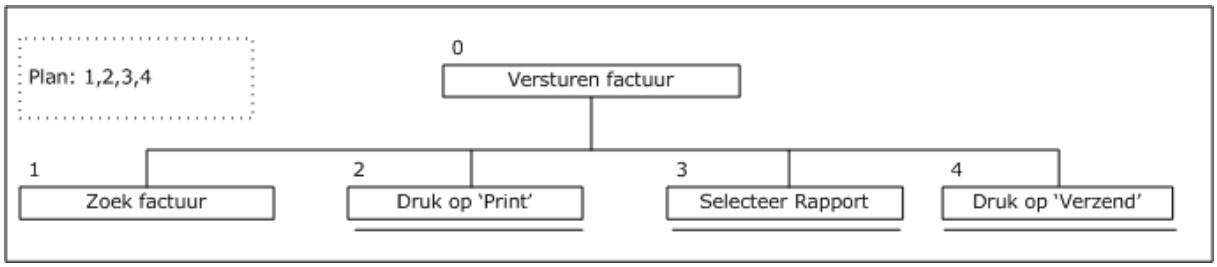
7.1.2 Factuur zoeken



7.1.3 Factuur Verwijderen



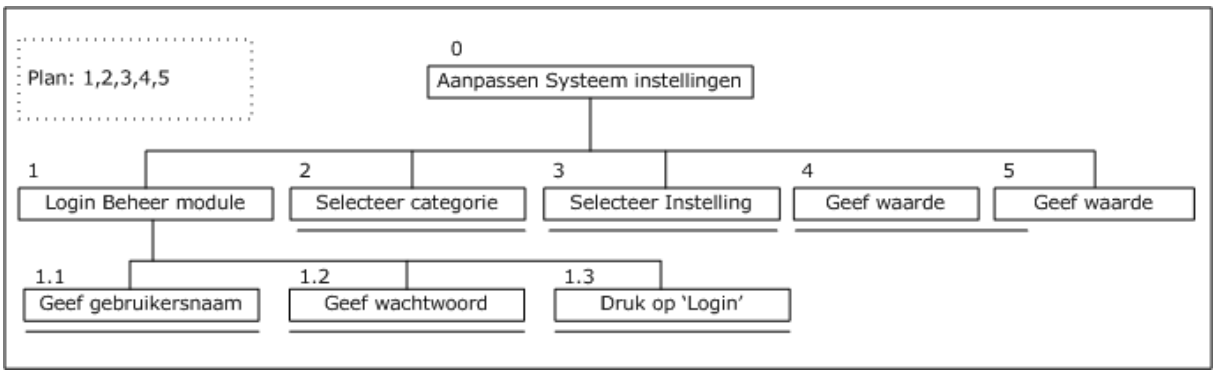
7.1.4 Factuur Verzenden



7.1 Module 'Beheer'

Voor wat betreft de module 'Facturatie' wordt het facturatieproces geautomatiseerd.Dit doen op basis van een aantal taken. Al deze taken worden uitgevoerd door de actor: Administrator

Hieronder vind u een beschrijving van taken die er worden geautomatiseerd.



8. Technische Structuur

De technische uitgangspunten worden verondersteld gedefinieerd te zijn in de initiële reikwijdte van het project; dit geldt voor de hardware (zowel te behoeve van de ontwikkeling als voor operationele doeleinden) en voor de software. Deze activiteit maakt deel uit van het systeemconcept en zal een inzicht geven over de plaats van het systeem binnen de infrastructurele mogelijkheden van een organisatie op het gebied van de Informatie Technologie. Tevens wordt er beschreven hoe de toekomstige structuur van het de te ontwikkelen modules eruit ziet en hoe deze gekoppeld wordt aan Key-Task.

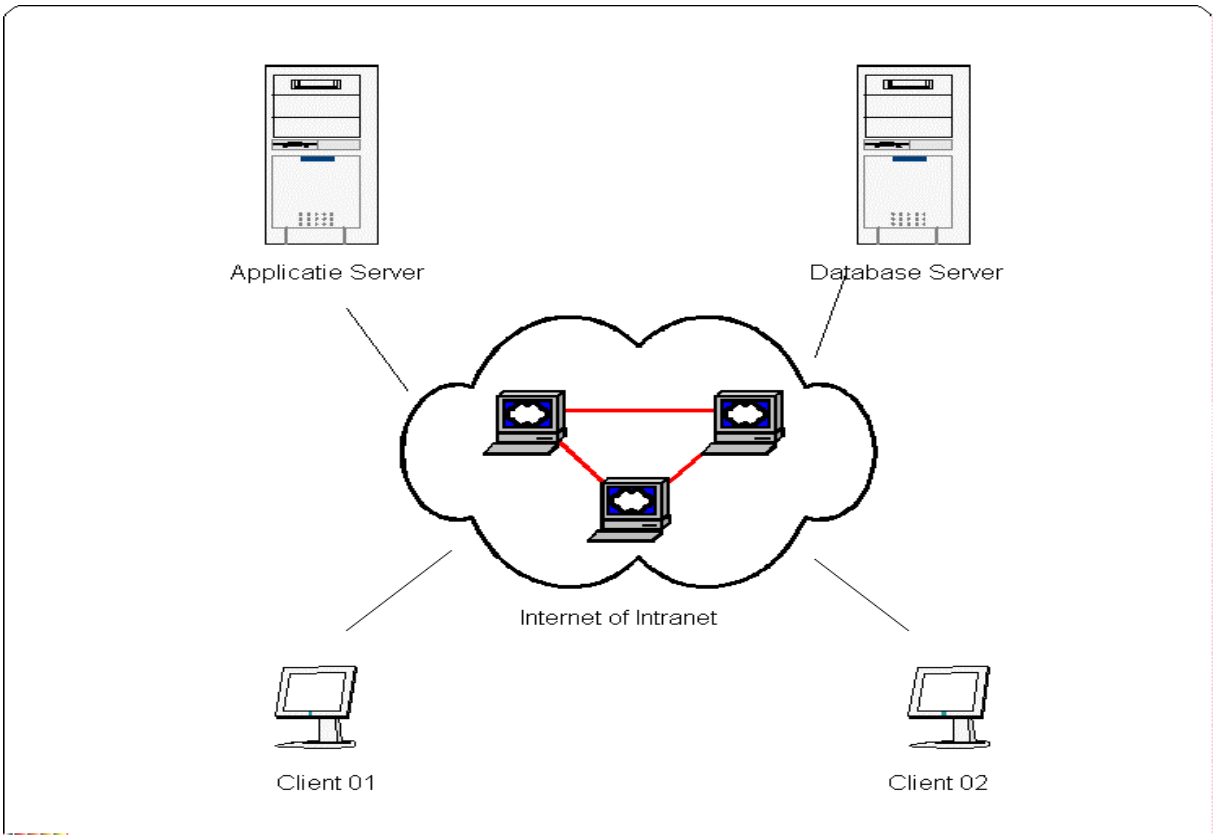
8.1 De technische Architectuur van de hardware

Binnen deze paragraaf worden alle elementen besproken die van invloed zijn op het succesvol hosten van het systeem. Dit omvat zowel hardwarematige als softwarematige elementen.

Zoals binnen de systeemeisen staat beschreven moet de te ontwikkelen modules gebruik maken van de hardware en software-eisen van Key-Task. Voor het gemak zullen we dan ook deze eisen binnen dit hoofdstuk bespreken.

De inhoud van paragraaf 8.1 is dan ook afkomstig uit de definitiestudie van Key-Task.

Figuur 8.1 laat een grafische weergave zien van al deze elementen.



Figuur 8.1: Technische omgeving Key-Task Systeem

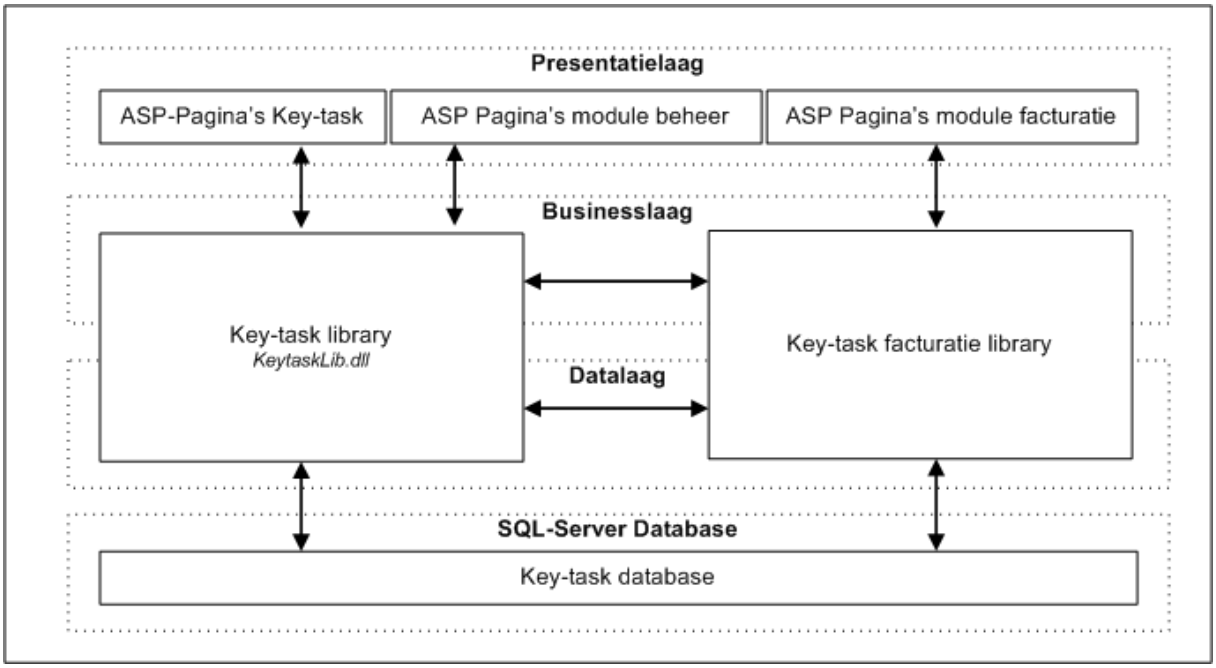
De te ontwikkelen modules dienen te worden gehost door een computer met daarop Microsoft Internet Information Server. De kenmerken van de machine van invloed op de performance van het systeem. Gesteld kan worden dat bij een machine met minimaal een 700 Mhz Pentium 3 processor en 128 MB intern geheugen de performance ter allertijd toelaatbaar is. Dit is binnen de ontwikkeling van Key-Task uitgebreid getest. Indien de machine hogere systeemeisen heeft, wordt de performance in positieve zin beïnvloed.
Voor de Applicatie Server is het noodzakelijk dat er Microsoft ActivieX Data Object 2.5 of hoger staat geïnstalleerd. Deze componenten verzorgen de connectie met de database. U kunt deze vinden onder de naam MDAC(Microsoft Database Access Components) en is vanaf www.microsoft.com te downloaden.

Afhankelijk van het type DBMS wat wordt gebruikt moet er ook een database-server zijn. Het is geheel aan de systeembeheerder om te bepalen hoe hij dit wil inrichten. Microsoft SQL server en oracle zijn de meest gebruikte oplossingen.

Verder dient er beschikking te zijn over een netwerk in de vorm van internet of een intranet. Het is tevens weer aan de systeem beheerder hoe hij dit wil inrichten.

8.2 De technische Architectuur van de software

Binnen dit gedeelte gaan we in op de architectuur van de te ontwikkelen software. De software dient nauwkeurig aan te sluiten op het huidige Key-Task systeem. Een goede architectuur is onmisbaar. Figuur 8.2 geeft een goed inzicht in de manier waarop het Key-Task systeem en haar te ontwikkelen modules in elkaar zit.



Figuur 8.2: Software Architectuur vernieuwde Key-Task systeem

Ik heb gekozen voor een 3-lagen model. Ik heb hiervoor gekozen doordat met behulp van dit model het systeem schaalbaar is en met het oog op de toekomst goed aanpasbaar en herbruikbaar.

Allereerst worden de drie verschillende lagen, die terug komen binnen onze architectuur, besproken om een goed beeld te geven over de functie van een specifieke laag.

De Datalaag
Het voornamelijk doel van de datalaag is het behouden van de integriteit van de database gegevens. U moet hierbij denken indien er transacties op de database worden uitgevoerd, de datalaag de gegevens controleert op fouten. Om een voorbeeld te geven: Wanneer een record wordt toegevoegd met een telefoonnummer controleert de datalaag of het telefoonnummer wel een correct formaat heeft.
Tevens controleert de database bij het verwijderen van records of er geen relaties worden geschonden. Kortom de datalaag valt te vergelijken met het principe 'Stored Procedures' van het DBMS (Database Management System). In dit geval ben ik zodoende onafhankelijk van een DBMS, wat opgevat kan worden als een zeer groot voordeel.

De procedures binnen deze laag communiceren met ADO (ActiveX Data Objects) met de betreffende datasource. Deze technologie biedt de mogelijkheid connecties te maken met verschillende datasources. Vanwege de datasource onafhankelijkheid worden stored procedures vermeden.

De Businesslaag
De businesslaag bevat de functionaliteit van het systeem en bestaat uit allerlei objecten. Indien met betrekking tot een bepaalde functionaliteit bepaalde database transacties moeten worden uitgevoerd, doet het betreffende object een aanvraag bij een van de procedures uit de datalaag.
Enkel vanuit de businesslaag kunnen de procedures uit de datalaag worden aangesproken.

De Presentatielaag
De presentatielaag omvat datgene wat voor de gebruiker zichtbaar is. In het geval van de te ontwikkelen modules zijn dat de op te leveren ASP-pagina's, die uiteindelijk een resultaat vormt van de onderliggende lagen.

Zoals u in figuur 8.2 kunt zien zal de koppeling tussen Key-Task en de te ontwikkelen modules plaatsvinden op de businesslaag. De ontwikkeling van de modules zal een uitbreiding zijn op het Key-Task systeem. Het is noodzakelijk om zo veel mogelijk te herbruiken. We zullen dus zeer veel inspringen op de reeds ontwikkelde functionaliteit van Key-Task.

Door de gekozen opbouw is de applicatie erg schaalbaar. Er is een concessie gedaan en dat is om de business logic laag niet te programmeren voor MTS (Microsoft Transaction Server) omgeving aangezien dit de complexiteit op de korte termijn vergroot. Tot ongeveer 100 gebruikers zal MTS geen grote performance verbetering geven.

8.3 De haalbaarheid van de technische aspecten

Te verwachten is dat de geschetste situatie haalbaar is. Dit baseren wij op doormiddel van eigen ervaringen en de resultaten uit de definitiestudie.

8.4 De herbruikbare componenten

De businesslaag van de te ontwikkelen modules zijn volledig te herbruiken. Vanuit de windows omgeving kunnen referenties worden gelegd naar de klasse bibliotheek. Dit maakt dat de zogenaamde Basissysteem-eisen te herbruiken zijn. De presentatie daarentegen wordt niet ontwikkeld als herbruikbaar component, omdat de technieken die wij hiervoor gebruiken niet toereikend zijn voor dergelijke doeleinden. Voor het gebruiken van de juiste technieken dienen er aanpassingen worden gedaan aan de hardware en software-eisen. Zoals u binnen het hoofdstuk systeemeisen heeft kunnen lezen dient te worden voldaan aan de hardware en software-eisen van Key-Task. Hierdoor kunnen we geen gebruik maken van technieken die het mogelijk maakt de presentatie optimaal herbruikbaar maakt.

9. Organisatorische inrichting

Het doel van dit hoofdstuk is vast te stellen wat de impact van het project is.

9.1 De werkstroom

Binnen de huidige situatie wordt de werkstroom binnen de organisatie gekenmerkt door veel formulieren die in een ongedigitaliseerde vorm door de organisatie en haar netwerken circuleren. Doormiddel van het automatiseren van de bedrijfsprocessen die gepaard zijn bij Service Management wordt de werkstroom vereenvoudigd en worden de kosten beperkt. Gegevens omtrent een bepaald formulier zijn voor meerdere personen beschikbaar en sneller benaderbaar.

9.2 De organisatorische gevolgen

Dit project heeft vooral zijn invloed op het facturatieproces van een organisatie. Wij gaan ervan uit dat het factureren vooralsnog handmatig gebeurt. De gevolgen op de organisatie zijn van die aard dat de menselijke handelingen worden beperkt doordat dit proces wordt geautomatiseerd.

9.3 De vereiste gebruikersdocumentatie

Er zal aanvullende documentatie worden aangeleverd om het systeem onderhoudbaar te houden en om duidelijkheid te verschaffen over de werkzaamheden. Metrix kan op deze manier het systeem makkelijk onderhouden en uitbreiden.

10. Pilotplan

In dit hoofdstuk wordt ingegaan op de vraag: "Welke pilots worden uitgevoerd?". Tevens wordt binnen dit hoofdstuk beschreven hoe en wanneer dit wordt bewerkstelligd. Dit alles is afhankelijk van de prioriteit van de pilot.

10.1 De uit te voeren Pilots

De volgende pilots worden uitgevoerd:

1. Facturatie
2. Beheer

10.1.1 Pilot Facturatie

Binnen de pilot 'Facturatie' wordt er gewerkt aan de realisatie van de module 'Facturatie'. Deze module is een uitbreiding op het bestaande Key-Task systeem. Bij de ontwikkeling van de module zal rekening gehouden worden met het Technische Architectuur van het huidige systeem. Dit omdat de diverse systeemonderdelen in nauwe samenwerking dienen te functioneren.

10.1.2 Pilot Beheer

De pilot 'Beheer' heeft tot doel de module 'Beheer' te verwezenlijken. Binnen de module 'Beheer' heeft de gebruiker de mogelijkheid om de systeeminstellingen te bekijken en zonodig te bewerken. Voor de uitvoer van deze pilot dient eerst in kaart worden gebracht welke instellingen worden meegenomen binnen deze module.

De indeling van de pilots is een logisch gevolg van de opdrachtomschrijving. Er is gekozen voor zoweinig mogelijk pilots dit om de overzichtelijkheid van het project te waarborgen.

10.2 Ontwikkelscenario

Het hoofddoel van deze activiteit is de reikwijdte en de context van het ontwikkeltraject vast te leggen of te bevestigen en ervoor te zorgen dat de iteratieve ontwikkelstrategie op de juiste wijze gebruikt wordt.

10.2.1 De prioriteit van de te ontwikkelen onderdelen

Binnen deze paragraaf wordt per module de prioriteit aangegeven. Dit om eventuele risico's op te vangen.

Prioriteit:

1. Pilot 'Facturatie'
2. Pilot 'Beheer'

De pilot 'Facturatie' heeft de hoogste prioriteit, dit op aandringen van de gebruiker die meer waarde hecht aan de voltooiing van het algemene proces waarvan facturatie onderdeel uitmaakt. Tevens is het binnen een noodsituatie mogelijk tot tijdsverkortende alternatieven voor wat betreft de pilot 'Beheer'.

Uit deze prioritering kan worden opgemaakt, dat pas aan de ontwikkeling van de pilot 'Beheer' kan worden begonnen op het moment dat de pilot 'Facturatie' is goedgekeurd door zowel de opdrachtgever als zijnde de eindgebruikers. Voor wat

betreft de eindgebruikers zal dit worden bewerkstelligd aan de hand van een gebruikerstest.

10.2.2 Pilottesten

Na iedere iteratie van de fase pilotontwikkeling zal de pilot worden onderworpen aan een gebruikerstest. Deze test moet ervoor zorgen dat er eventuele fouten of gebreken worden verholpen voordat de pilot daadwerkelijk wordt ingevoerd. De volgende testmethodes worden uitgevoerd op de volgende pilots:

- Pilot 'Facturatie' - Use-Case test
- Pilot 'Beheer' - Heuristic Evaluation

Voor elke pilot wordt er een apart testrapport geschreven. Dit rapport zal worden hergebruikt indien zich nieuwe iteraties voordoen.

10.2.3 Planning

Pilot 'Facturatie'	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Stel Plan van aanpak op															
Bereid workshop voor															
Voer pilotontwerp-workshop															
Stel pilotontwikkelplan op															
Bouw pilot 'Facturatie'															
Maak handleiding															
Pilot 'Beheer'															
Stel Plan van aanpak op															
Bereid workshop voor															
Voer pilotontwerp-workshop															
Stel pilotontwikkelplan op															
Bouw pilot 'Beheer'															
Maak Handleiding'															

Pilotontwikkelplan Facturatie

Naam document:	Pilotontwikkelplan 'Facturatie'
Bestandsnaam:	Pilot_facturatie_v2.doc
Project:	De ontwikkeling van de modules 'Facturatie' en 'Beheer'
Datum oplevering:	15/10/2003
Omschrijving van de inhoud:	Dit document beschrijft het ontwerp van de pilot 'Facturatie'.
In opdracht van:	Metrix B.V.
Versie:	2.00
Auteur:	Ruben Maas

Inhoudsopgave

1 Inleiding 3

2 Plan van Aanpak 4

3 Functionele structuur 6

4 Ontwerp Presentatielaag 7

5 Ontwerp Businesslaag 11

6 Ontwerp Datalaag 14

7 Ontwerp database 21

8 Aanpassingen externe componenten 22

9 Bouweenheden 23

10 Invoering 24

1 Inleiding

Binnen dit document wordt beschreven hoe het systeem zal worden ontworpen. Allereerst kunt u in hoofdstuk 2 het plan van aanpak lezen.

In hoofdstuk 3 ziet u hoe ik zowel het interview met de opdrachtgever als de workshop met de gebruikers heb aangepakt.

In hoofdstuk 4 zal ik de functionele-structuur beschrijven hierin zal de benodigde functionaliteit worden blootgelegd.

Vervolgens zijn hoofdstuk 4,5 en 6 besteedt aan het beschrijven van de technische-structuur.

2 Plan van aanpak

Doelstelling
Voor deze pilot heb ik een doelstelling geformuleerd, die luidt als volgt:
De pilot 'Facturatie' moet ervoor zorgen dat de menselijke handelingen, binnen het huidige facturatieproces, worden geautomatiseerd.

Op te leveren producten
De pilot 'Facturatie' voorziet in de volgende producten:
- Het document pilotontwikkelplan 'Facturatie'
- De module 'Facturatie'
- Handleiding 'Module facturatie'
- Minimaal 1 testrapport (*Afhankelijk van het aantal iteraties*)

Uit te voeren activiteiten
De volgende activiteiten worden in deze fase uitgevoerd:
- Beschrijven functionele structuur
- Beschrijven technische structuur
- Beschrijven database
- Beschrijven bouwdelen

Functionele structuur
Deze activiteit gebruiken ik om de functionaliteit van deze pilot in kaart te brengen.

Technische structuur
Binnen deze activiteit wordt beschreven hoe het systeem zal worden opgebouwd.
De technische structuur is opgedeeld is drie delen, namelijk:
- Ontwerp Presentatielaag
- Ontwerp Businesslaag
- Ontwerp Datalaag

Beschrijven database
Aan de hand van de bovenstaande activiteiten kan ik bepalen welke toevoegingen er verricht moeten worden aan de Key-Task database

Beschrijven bouwdelen
Mede aan de hand van de technische structuur zal ik de pilot in verschillende bouweenheden opdelen.

Pilot 'Facturatie'									
Stel Plan van aanpak op									
Bereid pilotontwerp-workshop voor									
Voer pilotontwerp-workshop uit									
Beschrijven Functionele structuur									
Beschrijven ontwerp presentatielaag									
Beschrijven ontwerp businesslaag									
Beschrijven ontwerp datalaag									
Beschrijven ontwerp database									
Beschrijven Bouweenheden									
Beschrijven testprocedure									
Beschrijven aanpassingen externe componenten									
Bouw pilot									
Maak handleiding									

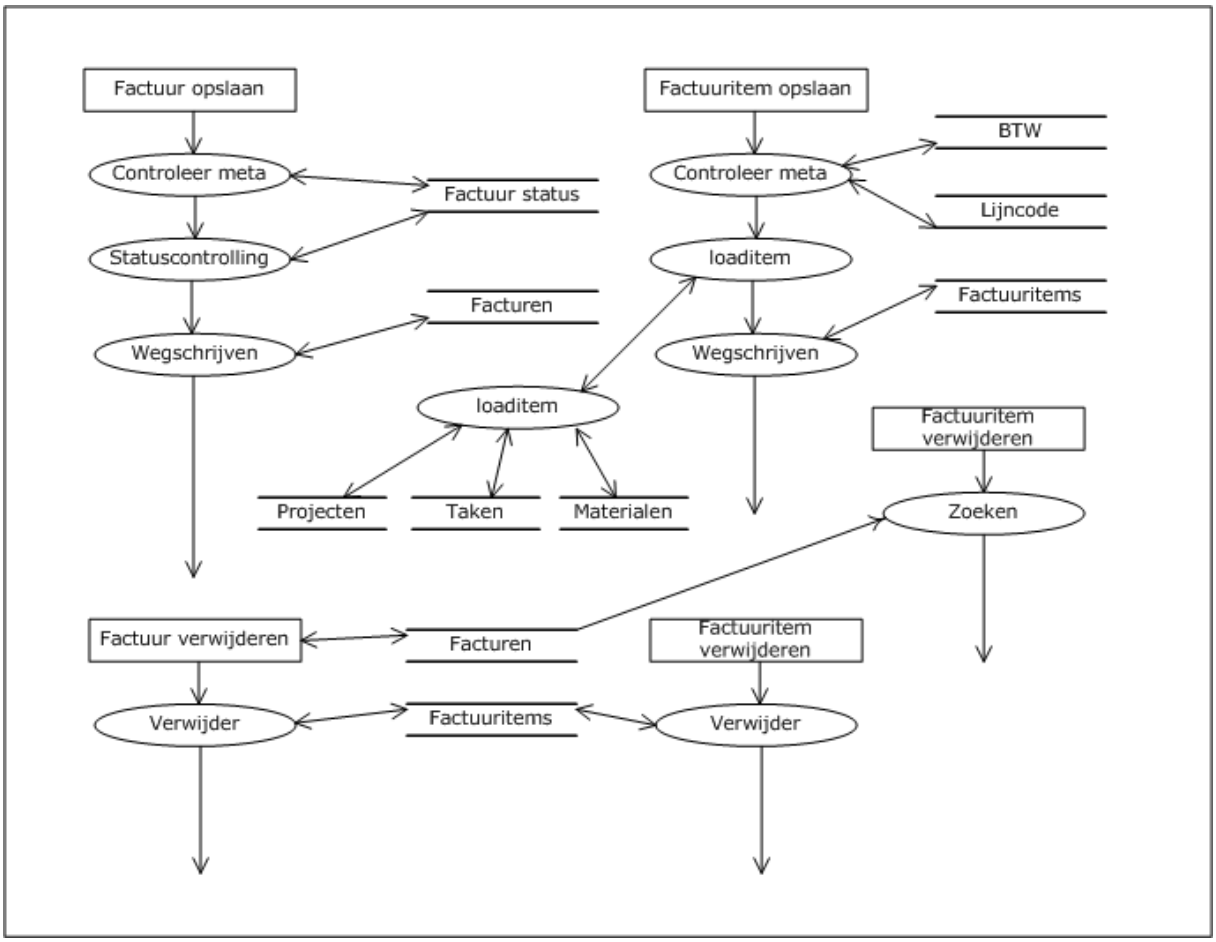
2.1 De activiteiten met de uit te voeren werkzaamheden

3 Functionele structuur

Binnen dit hoofdstuk zal vastgelegd worden over welke functionaliteiten deze pilot moet beschikken. De functionaliteiten worden bepaald aan de hand van de systeemeisen die staan beschreven in de definitiestudie.

Over de volgende functionaliteiten moet de pilot facturatie beschikken:

- Factuur aanmaken
- Factuuritem aanmaken
- Factuur verwijderen
- Factuuritem verwijderen
- Project, taak, materiaal factureren
- Factuur zoeken
- Factuur verzenden



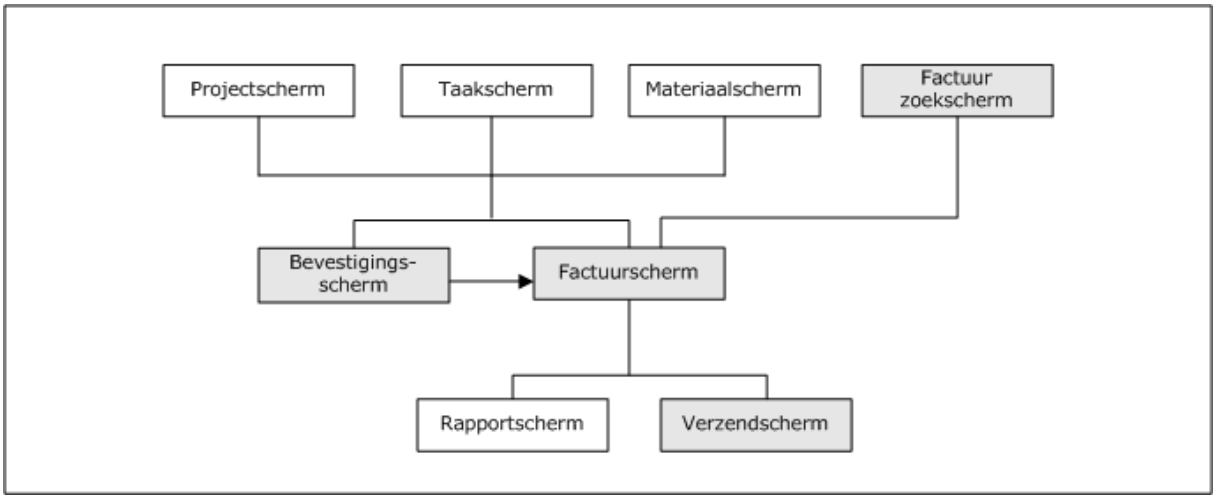
Figuur 3.1 DFD pilot 'Facturatie'

4 Ontwerp presentatielaag

Binnen dit hoofdstuk wordt beschreven hoe de presentatielaag eruit komt te zien. Dit wordt bewerkstelligd aan de hand van een navigatieschema, schermontwerp en de te hanteren iconen en kleuren.

4.1 Navigatieschema

Om te kunnen bepalen welke schermen er dienen te worden ontwikkeld, heb ik een navigatieschema opgesteld.



Figuur 4.1 navigatieschema

Bij het navigatieschema geldt dat de entiteiten die niet grijnsgetint zijn al reeds zijn ontwikkeld en onderdeel uitmaken van het huidige Key-Task systeem. De volgende pagina's moeten dus worden ontwikkeld:

- Factuur zoekscherm
- Bevestigingsscherm
- Factuurscherm
- Verzendscherm

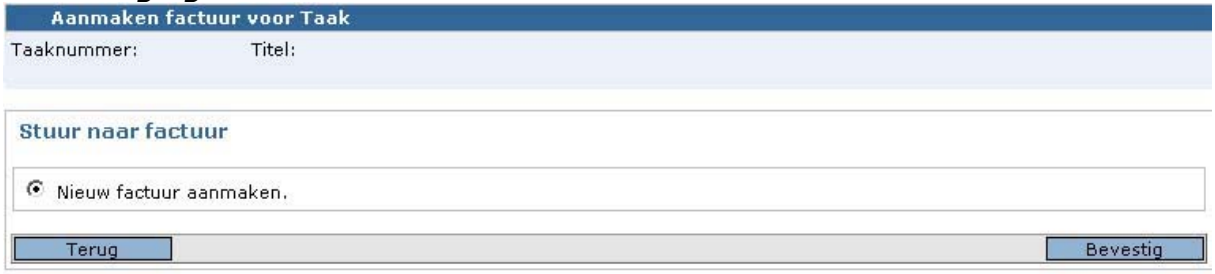
In hoofdstuk 4.2 zullen de te ontwerpen schermen worden beschreven.

4.2 Schermontwerp

Binnen dit hoofdstukken wordt het schermontwerp besproken. Het schermontwerp bestaat uit een statische HTML-pagina van de te ontwikkelen pagina.

Het bevestigingsscherm heeft als doel de gebruiker voor de keuze te stellen of hij/zij het te factureren item op een nieuwe factuur of een bestaande factuur wil plaatsen. De bestaande factuur moet dan wel beschikken over de status 'Pro forma'.

Bevestigingsscherm

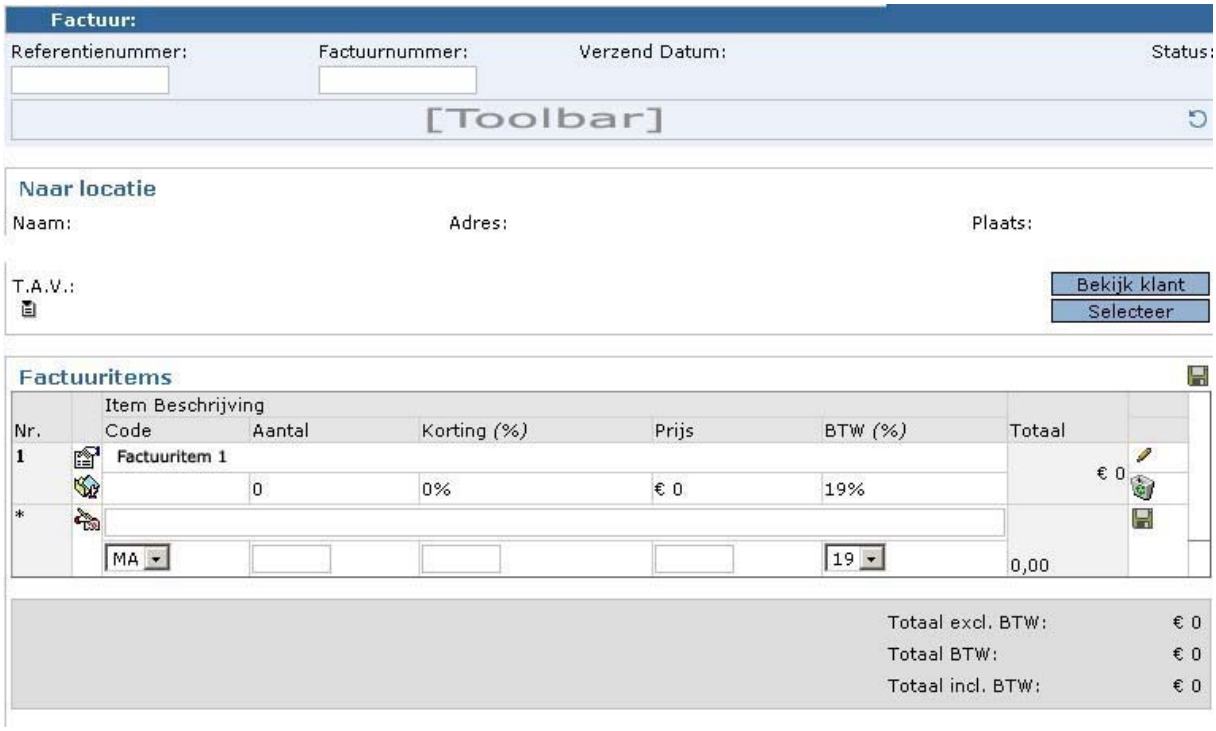


Figuur 4.2 Bevestigingsscherm

Factuurscherm
Het factuurscherm heeft als doel alle gegevens omtrent een factuur aan de gebruiker te tonen. Deze gegevens zijn onder te verdelen in een tweetal categorieeen, namelijk header -en detailinformatie.

Headerinformatie:
- Factuurnummer
- Factuurid
- Verzenddatum
- Status
- Klant
- Contactpersoon

Detailinformatie:
- Factuuritems
  - Itemnummer
  - Aantal
  - Beschrijving
  - Korting
  - Lijncode
  - BTW



Figuur 4.3 Factuurscherm

Binnen het facturatiescherm is ruimte gelaten voor een zogenaamde toolbar. Deze toolbar heeft als doel de benodigde gerelateerde functionaliteiten te huisvesten. Dit wordt bewerkstelligd aan de hand van een aantal iconen. Binnen het factuurscherm is gekozen voor de volgende iconen:

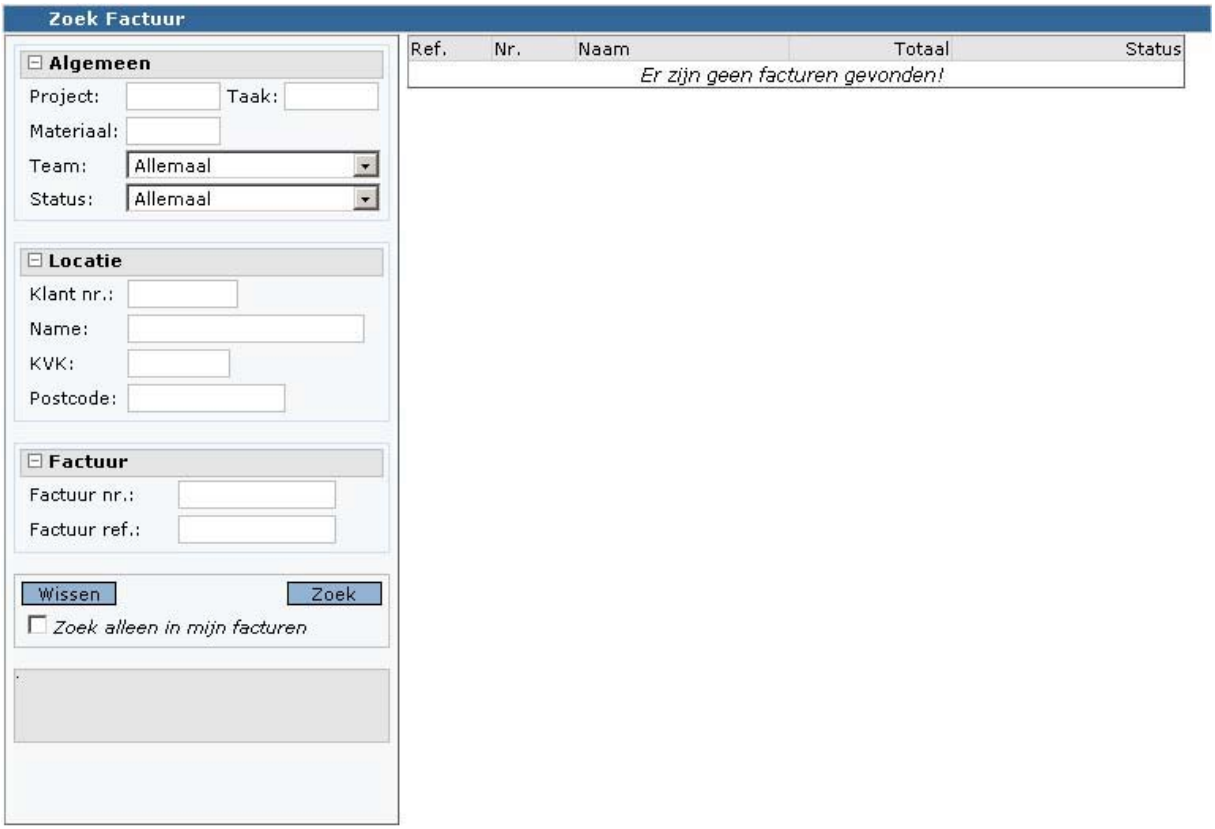
	Nieuw factuur aanmaken
	Factuur verzenden
	Hiermee wordt aangegeven dat het factuuritem handmatig is aangemaakt
	Factuur of factuuritem opslaan
	Factuur of factuuritem verwijderen
	Factuur printen

	Naar taak, project of materiaal

Zoekscherm

Het zoekscherm heeft als doel facturen te kunnen opzoeken aan de hand van de volgende criteria:

- Projectnummer
- Materiaalnummer
- Taaknummer
- Team
- Status
- Klantnummer
- Klantnaam
- KVK code
- Postcode klant
- Factuurnummer
- Factuurid



Figuur 4.4 Zoekscherm

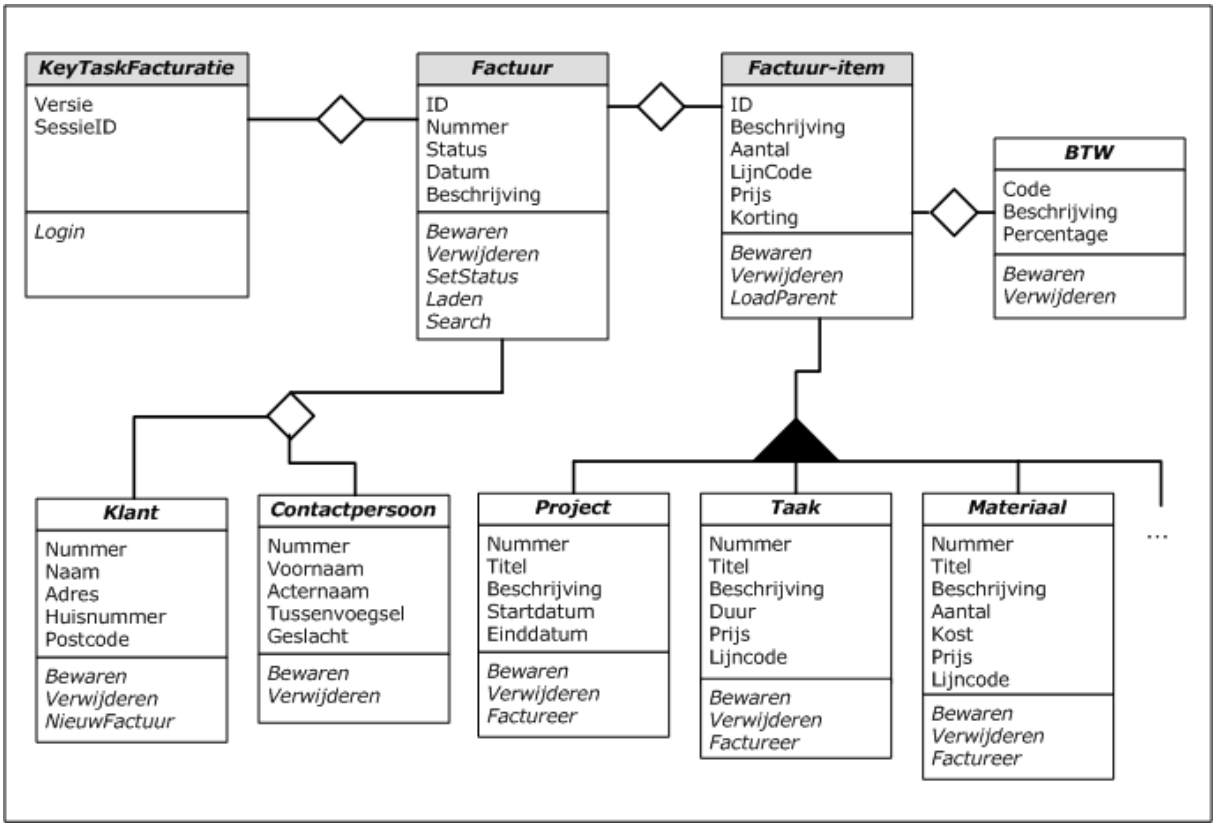
Verzendscherm

5 Ontwerp Businesslaag

Binnen dit hoofdstuk wordt de businesslaag van de pilot facturatie beschreven.

5.1 OMT model

Het onderstaande OMT model laat de te ontwikkelen objecten zien. Ter verduidelijking: De grijs getinte objecten dienen ontwikkeld te worden. De rest is reeds ontwikkeld.



De 3 grijs getinte objecten vormen de klassenbibliotheek KeyTaskFacturatieLib. Centraal aanspreekpunt hierbij is het object KeyTaskFacturatie. Om toegang te krijgen tot de Key-Task database dient de gebruiker eerst in te loggen. Dit wordt bewerkstelligd doormiddel van de procedure 'Login'. Deze procedure verwijst naar de KeyTaskSessionLib Library die het inloggen verder behandeld. De procedure geeft een Session object terug waaruit de zogehete datasource uit gehaald kan worden.

Het 'Factuur' Object

De volgende eigenschappen zijn van toepassing tot dit object:

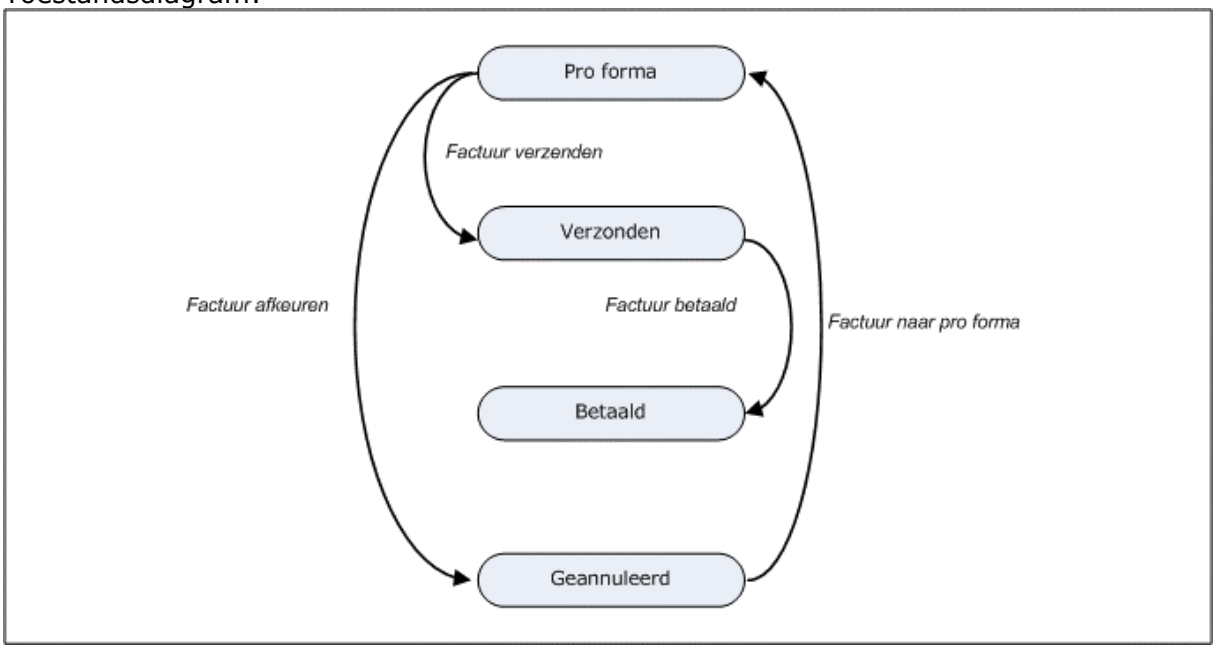
Eigenschap	Datatype	Beschrijving
ID	Numeriek	Dit is de primaire sleutel. Aan de hand van deze eigenschap kun je een specifiek factuur object opvragen.
Beschrijving	Tekst	Bevat een beschrijving over de uit te voeren activiteit
Aangemaakt	Datum/Tijd	Bevat de datum waarop het object is gecreeerd
AangemaaktDoor	Employee	Bevat een employee object van degene die de activiteit heeft aangemaakt
Items	Collectie	Bevat nul of meerdere task objecten
Locatie	Klant	Bevat nul of meerdere expense objecten
Contactpersoon	Contactpersoon	Bevat een employee object van degene die verantwoordelijk is voor de uitvoer van de activiteit
Status	Numeriek	Bevat een numerieke code, die een bepaalde status weergeeft. De volgende statussen zijn van toepassing: - 100 Proforma - 200 Verzonden - 300 Betaald - 400 Geannuleerd

Nummer	Numeriek	Bevat het factuurnummer. Dit factuurnummer wordt gecreëerd indien de factuur naar de status verzonden wordt gezet.

	Deze eigenschappen kun je alleen lezen

Procedures	**Resultaat**	**Beschrijving**
Bewaren	Boolean	Zorgt ervoor dat de waardes omtrent dit object worden weggeschreven naar de database. Deze procedure geeft true terug indien geslaagd.
Verwijderen	Boolean	Zorgt ervoor dat het betreffende record verwijderd wordt uit de database
SetStatus	Boolean	Veranderd de status van het betreffende factuur object en schrijft de gewijzigde attributen weg naar de database.
Laden	Boolean	Aan de hand van een factuur referentienummer kan de betreffende factuur in het object worden geladen.
Search	Collectie	Aan de hand van de attributen geeft deze procedure een collectie van facturen terug die daar aan overeenkomen.

Toestandsdiagram:



Het 'Factuuritem' Object

De volgende eigenschappen zijn van toepassing tot dit object:

Eigenschap	Datatype	Beschrijving
ID	Numeriek	Dit is de primaire sleutel. Aan de hand van deze eigenschap kun je een specifiek factuuritem object opvragen.
Beschrijving	Tekst	Bevat een beschrijving over over het betreffende item
Aangemaakt	Datum/Tijd	Bevat de datum waarop het object is gecreëerd
AangemaaktDoor	Employee	Bevat een medewerker object van degene die het factuuritem heeft aangemaakt
Itemnummer	Numeriek	Sequentieel itemnummer
Aantal	Numeriek	Geeft aan hoeveel van het betreffende item in rekening wordt gebracht
Prijs	Valuta	De prijs wat in rekening wordt gebracht
Korting	Numeriek	Geeft het kortingspercentage aan
BTW code	Tekst	Bevat de code van de BTW die gehanteerd dient te worden
Lijn code	Tekst	Bevat de code van de lijncode die gehanteerd wordt.

	Deze eigenschappen kun je alleen lezen

Procedures	Resultaat	Beschrijving
Bewaren	Boolean	Zorgt ervoor dat de waardes omtrent dit object worden weggeschreven naar de database. Deze procedure geeft true terug indien geslaagd.
Verwijderen	Boolean	Zorgt ervoor dat het betreffende record verwijderd wordt uit de database
loadParent	Boolean	Koppeld een taak, project of materiaal aan het betreffende item

6 Ontwerp Datalaag

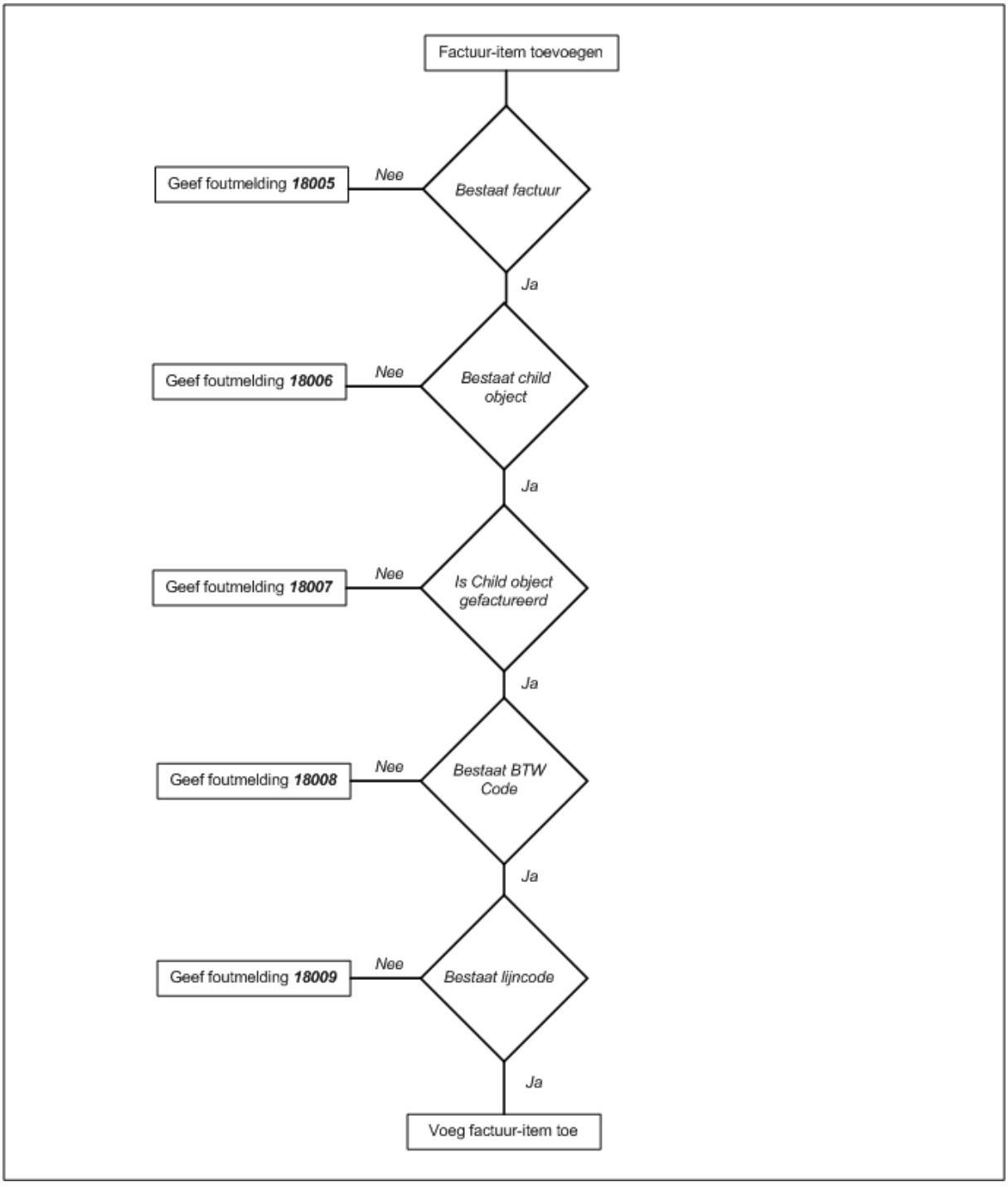
Binnen dit hoofdstuk zal de datalaag worden ontworpen. De datalaag bestaat uit tal van procedures die de integriteit van de database bewaken. De procedures worden in hoofdstuk 5.1 beschreven aan de hand van beslissingsmodellen.

6.1 Beslissingsmodellen

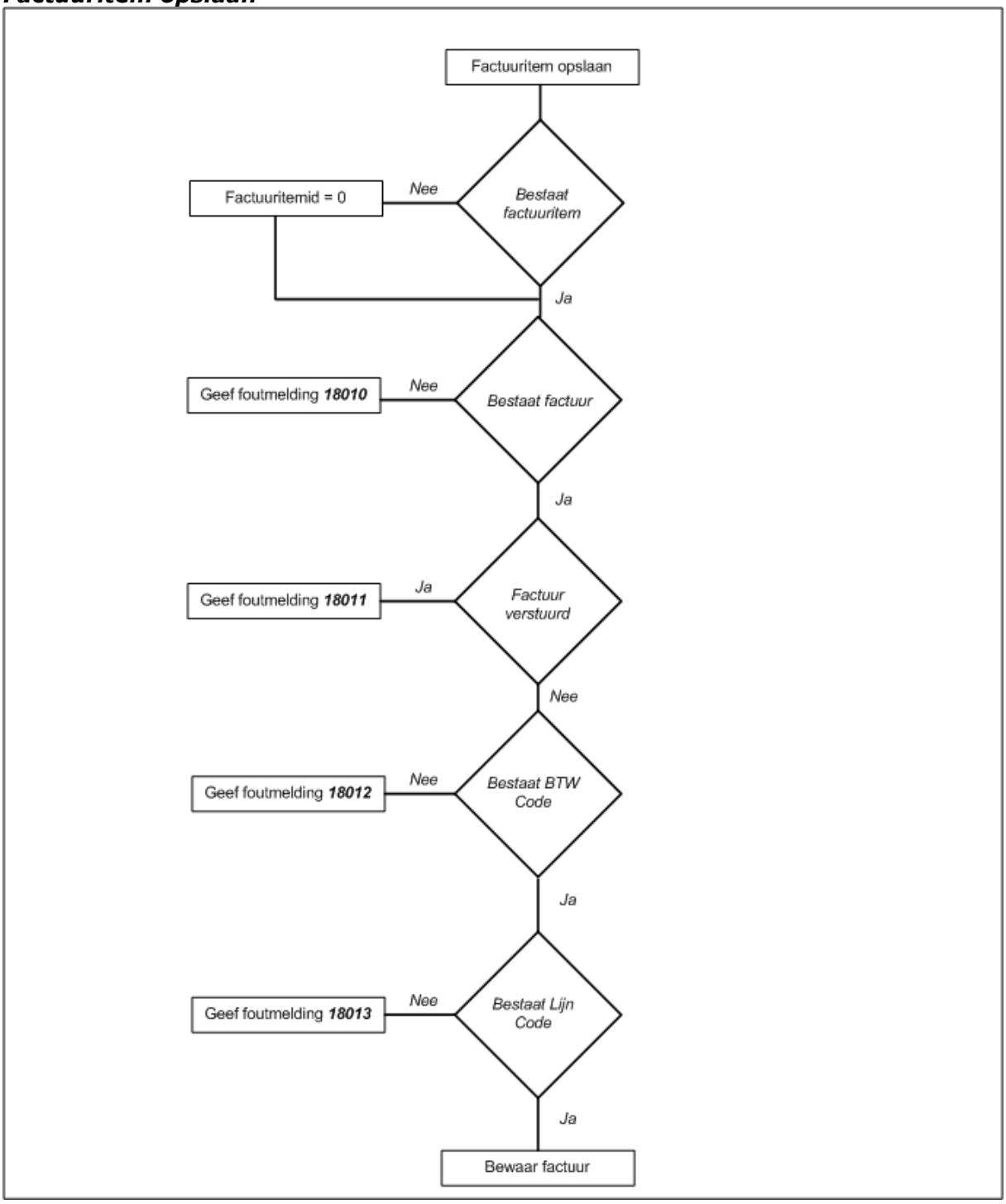
Nu de businesslaag is ontworpen kan ik bepalen welke integriteitcontroles er uitgevoerd dienen te worden. De volgende controles moeten worden uitgevoerd:

- Factuuritem toevoegen
- Factuuritem opslaan
- Factuur Opslaan
- Factuur setstatus
- Factuur verwijderen
- Factuuritem verwijderen

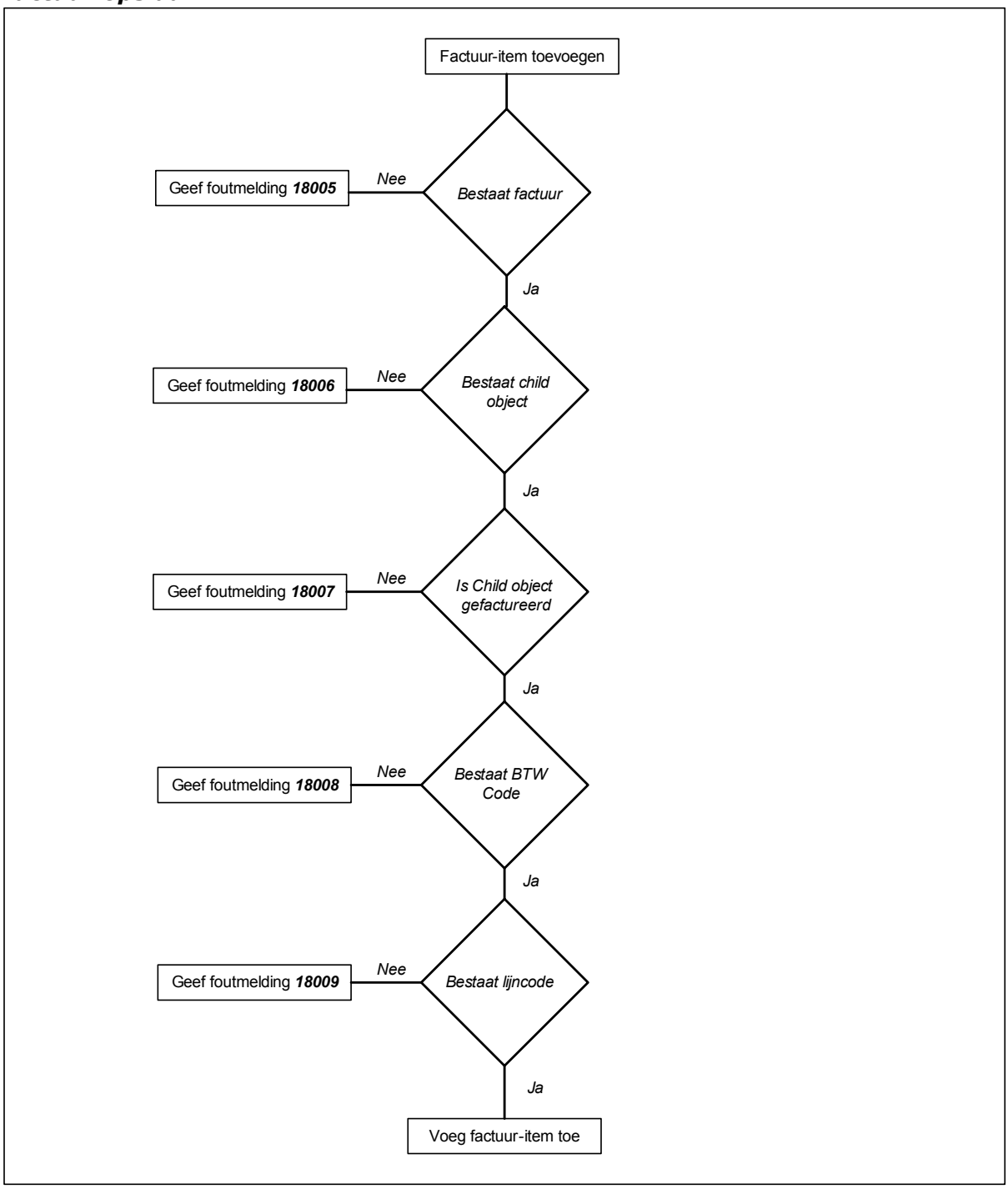
Factuuritem toevoegen



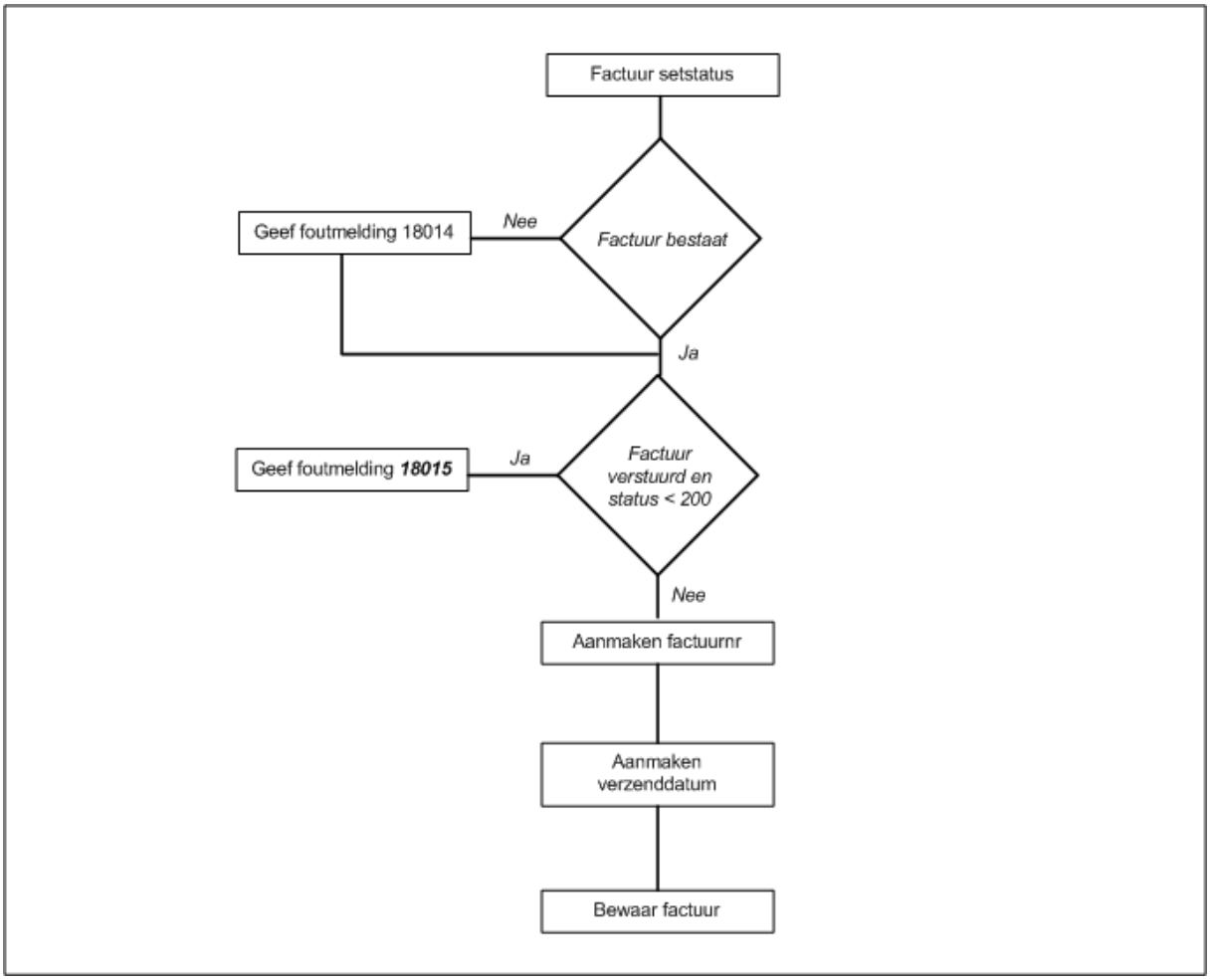
Factuuritem opslaan



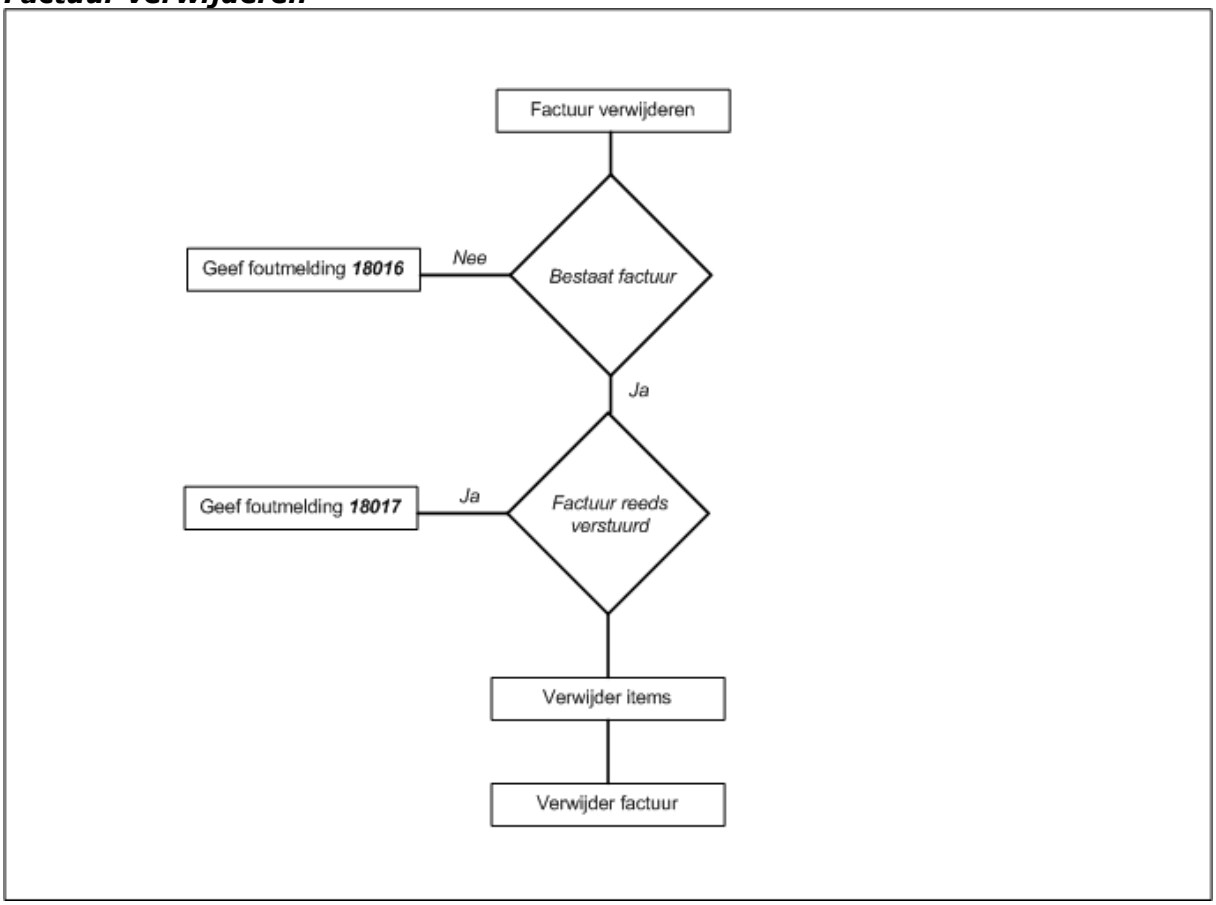
Factuur opslaan



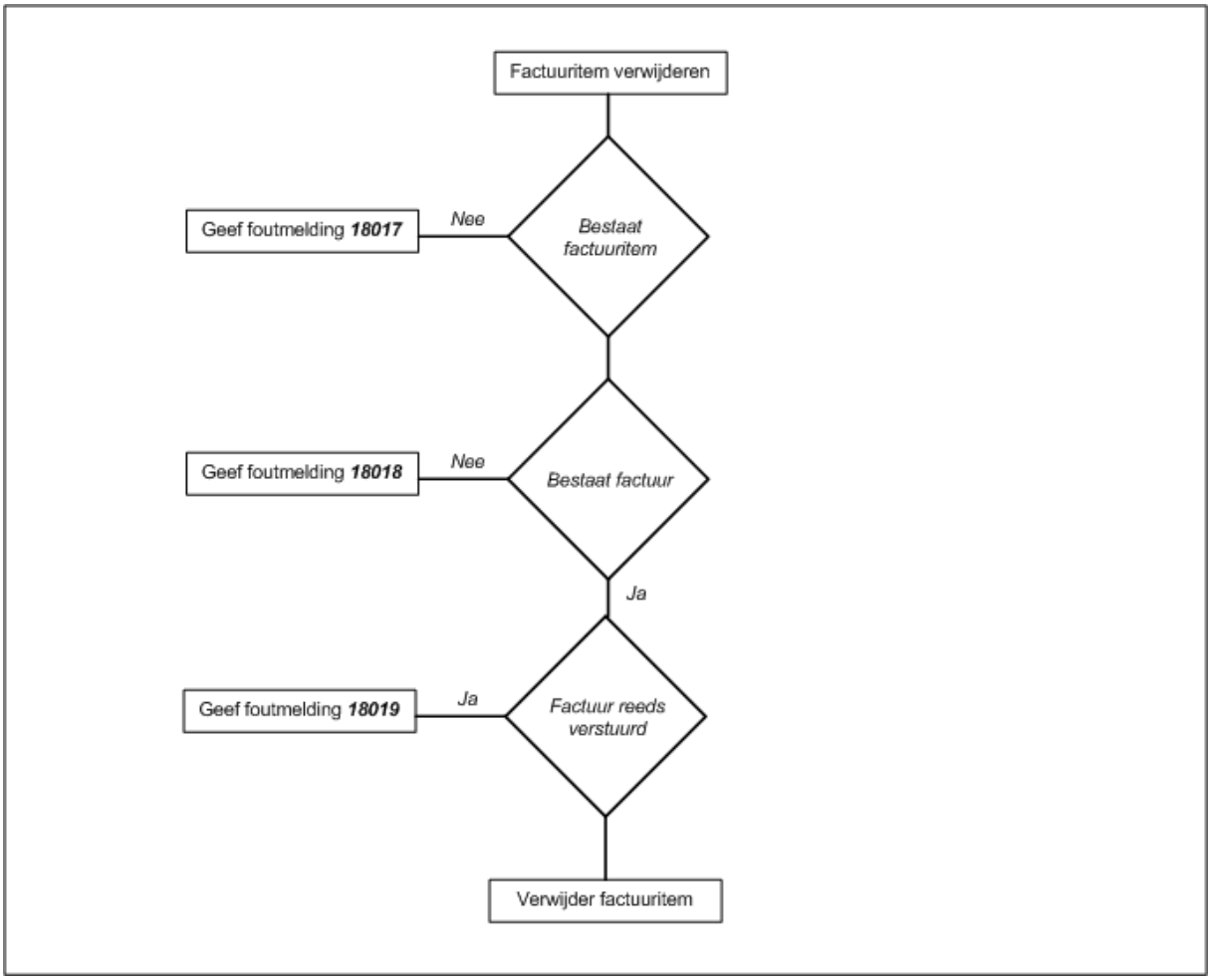
Factuur setstatus



Factuur verwijderen



Factuuritem verwijderen



7 Ontwerp database

Factuur (factuur_id, factuur_nummer, factuur_status, factuur_datum, factuur_adres, factuur_contact, factuur_opmerking)
Foreign key factuur_adres references "adres_id" in adres. Nulls not allowed
Foreign key factuur_contact references "contact_id" in contact. Nulls not allowed

Factuuritem (factuur_item_id, factuur_item_factuur, factuur_item_prijs, factuur_item_aantal, factuur_item_korting, factuur_item_btw, factuur_item_beschrijvi
factuur_item_lijn_code, factuur_item_parent_id, factuur_item_parent_type)
Foreign factuur_item_factuur references "factuur_id" in factuur. Nulls not allowed
Foreign factuur_item_btw references "btw_code" in btw. Nulls not allowed
Foreign factuur_item_lijn_code references "lijncode_code" in lijncode. Nulls not all

Factuurstatus (factuur_status_code, factuur_status_beschrijving)

8 Aanpassingen externe componenten

Binnen dit hoofdstuk zal ingegaan worden in de aanpassingen die verricht moeten worden aan externe componenten van deze module/pilot. U kunt hierbij denken aan het toevoegen van knoppen in diversen bestaande schermen van Key-Task.

De volgende aanpassingen dienen verricht te worden:

Bron	**Beschrijving**
Activitydetail.asp	Knop toevoegen 'factureer project'
Timedetail.asp	Knop toevoegen 'factureer taak'
Materialdetail.asp	Knop toevoegen 'factureer taak'
Timedetail.asp	Indicatie of de taak is gefactureerd
materialdetail.asp	Indicatiie of materiaal is gefactureerd
Actvitydetail.asp	Indicatie wat er is gefactureerd
Activitydetail.asp	Per item aangeven wat er is gefactureerd is.
Taskbar.inc	Snelzoek functie zoek factuur toevoegen.

9 Bouweenheden

Binnen dit hoofdstuk staan de bouweenheden beschreven en staan deze ingepland.

Nummer	Naam	Code	Prioriteit
1.	Realisatie datalaag	BE01	Hoog
2.	Realisatie businesslaag	BE02	Hoog
3.	Realisatie factuurscherm	BE03	Hoog
4.	Realisatie zoekscherm	BE04	Gemiddeld
5.	Realisatie bevestigingsscherm	BE05	Gemiddeld
6.	Realisatie verzendscherm	BE06	Laag
7.	Realisatie aanpassingen externe componenten	BE07	Laag

Code	Geschatte tijd	Werkelijke tijd
BE01	3 dagen	4 dagen
BE02	3 dagen	3,5 dag
BE03	2 dagen	2 dagen
BE04	3 dagen	3 dagen
BE05	2 dagen	2 dagen
BE06	2 dagen	1,5 dag

10 Invoering

De procedures zijn overwogen ten aanzien van de volgende punten.

- Hardware
- Software
- Opleiding
- Handleidingen

In de volgende paragrafen zal worden beschreven hoe de verantwoordelijkheden zijn per procedure en hoe de uiteindelijke procedure eruit ziet.
Verantwoordelijkheden invoering
Hieronder zullen per punt de verantwoordelijkheden van elke invoeringsprocedure beschreven worden.
De betrokkenen bij de invoering zijn:

- Ontwikkelaars (het verkopende bedrijf)
- Ondersteuners (helpdesk en dergelijke)
- Eindgebruikers
- Opdrachtgever

1.1.1 Hardware

Voor de hardware ligt de verantwoording bij de eindgebruikers. De hardware zal moeten voldoen aan de eisen zoals deze omschreven zijn in Hoofdstuk 7.1 De technische Architectuur van de hardware in de definitiestudie.
De Ontwikkelaars kunnen wel adviseren over de aan te schaffen hardware.

1.1.2 Software

Voor de software ligt de verantwoording ook bij de opdrachtgever. De software zal moeten voldoen aan de eisen zoals deze omschreven zijn in Hoofdstuk 7.1 De technische Architectuur van de hardware in de definitiestudie. In dit hoofdstuk staat tevens genoemd aan welke eisen de software moet voldoen.

1.1.3 Opleiding

De verantwoording voor de opleiding van de verschillende gebruikers ligt in eerste instantie bij de ondersteuners. De ondersteuners worden opgeleid door een consultant die van het verkopende bedrijf uitgaat. Daarnaast zullen de ontwikkelaar een handleiding schrijven dat beschrijft hoe het product werkt. Van de ondersteuners wordt verwacht dat zij deze handleiding goed kennen.

1.1.4 Handleidingen

De handleidingen zijn de verantwoording van de ontwikkelaars. Zij zullen mede in het kader van de opleiding een handleiding schrijven die de werking van het product omschrijft. De ondersteuners binnen het bedrijf kunnen zelf toevoegingen op deze handleiding schrijven. Echter zullen deze toevoegingen ook de bouwers moeten toekomen om deze extra informatie toe te kunnen voegen aan de handleidingen zoals deze zijn aangeleverd om zo een betere handleiding in de toekomst te kunnen uitleveren.

Testrapport pilot 'Facturatie'

Naam document:	Testrapport pilot facturatie
Bestandsnaam:	Testrapport_pilot_facturatie_it_1.doc
Datum oplevering:	06/11/2003
Iteratie:	0 & 1
Omschrijving van de inhoud:	Dit document beschrijft de aanpak en de testscenario's met betrekking tot de gebruikerstest iteratie 0 pilot 'Facturatie'
In opdracht van:	Metrix B.V.
Auteur:	Ruben Maas

Inhoudsopgave

1 Inleiding **4**

2 Plan van aanpak **5**

2.1 Doel 5
2.2 De testmethode 5
2.3 Beschrijving 5
2.4 Toepasbaarheid 5
2.5 Voorwaarden 5
2.6 Verwerking resultaten 5
2.7 Deelnemers 6
2.8 Werkwijze 6
2.9 Hulpmiddelen 6

Testscenario's **8**

3 Resultaten iteratie 0 ..... **13**

3 Resultaten iteratie 1 ..... Fout! Bladwijzer niet gedefinieerd.

Inleiding

Dit document beschrijft de aanpak van het Heuristic Evaluation testen in het kader van de pilot 'Facturatie'. In hoofdstuk 1 zal ik beschrijven hoe ik het testen ga aanpakken. Tenslotte zal ik hoofdstuk 2 wijden aan het beschrijven van de resultaten.

1 Plan van aanpak

1.1 Doel

Het vroegtijdig onderkennen van problemen van zowel de gebruikersinterface als de functionaliteit van de pilot ‘Facturatie’

1.2 De testmethode

De testmethode die gehanteerd zal gaan worden is Use-case testen. Deze testmethode test aan de hand van zogenaamde testscenario’s. Deze scenario’s worden uitgevoerd door meerderen testmethodes.

1.3 Beschrijving

Een groep gebruikers (4-6) voeren onafhankelijk van elkaar de testscenario’s uit. Binnen de testscenario’s zal elke functionaliteit binnen de pilot ’Facturatie’ getest worden. De testpersoon kan beoordelen of het resultaat juist is. Tevens kan de testpersoon beoordelen of de functionaliteit goed te benaderen is.

1.4 Toepasbaarheid

Deze test is toepasbaar indien de fase pilotontwikkeling doorlopen is. Deze test kan meerdere keren worden uitgevoerd, doordat de fase pilotontwikkeling meerdere iteraties kan hebben.

1.5 Voorwaarden

Er zijn tussen de 4 en 6 testpersonen aanwezig.
De scenario’s staan uitgebreid beschreven.

1.6 Verwerking resultaten

De deelnemers doen schriftelijk verslag van de test en deze worden vervolgens besproken met de ontwikkelaar. De ontwikkelaar staat vervolgens voor de keuze om aanpassingen te verrichten op de pilot ‘Facturatie’. Mochten er één of meerdere aanpassingen verricht worden dan komt de fase pilotontwikkeling in een nieuwe iteratie. Dit betekent dat er ter aller tijde alsnog een nieuwe test wordt uitgevoerd.

1.7 Deelnemers

- Ruben Maas
- Ab de Kwant
- Karel van Stee
- Michon v/d Berg
- Peter Jol

1.8 Werkwijze

1. Deelnemers doorlopen elk testscenario van elk ca 5 min
2. Bevindingen verzamelen en ordenen.
3. Bevindingen overdragen en bespreken.
4. Aandragen van verbeteringen.

1.9 Hulpmiddelen

Er wordt gebruik gemaakt van tien usability "heuristics" (controle punten):

1. Eenvoudige en natuurlijke dialoog (spreek de taal van de gebruiker) het systeem moet de taal van de gebruiker spreken, met woorden, zinnen en concepten waar de gebruiker mee vertrouwd is
   - voorkom systeem georiënteerde termen
   - presenteer informatie in een natuurlijke en logische volgorde
2. minimaliseer druk op geheugen van gebruiker
   - herkennen is beter dan herinneren
   - zorg dat men zo weinig mogelijk hoeft te onthouden
3. wees consistent
   - gebruikers moeten zich niet hoeven afvragen of verschillende woorden, situaties of acties hetzelfde betekenen
   - gebruik standaards
4. geef feedback
   - het systeem moet altijd aangeven wat er gebeurt of wat er is gebeurd
5. geef duidelijk uitgangen aan
   - gebruikers kiezen vaak verkeerde functies en hebben dan een duidelijke "nooduitgang" nodig
   - maak gebruik van "undo & redo"
6. wees flexibel en efficiënt
   - breng short cuts aan met name voor ervaren gebruikers
   - maak het mogelijk om veel gebruikte acties vast te leggen (voorbeeld rechter muisknop)
7. ontwerp goede foutboodschappen
   - help gebruikers bij het herstellen van fouten
8. voorkom fouten
   - nog beter dan een goed foutboodschap is het voorkomen van fouten

- vraag je bij elke foutboodschap af: had deze fout vermeden kunnen worden?

9. esthetisch en minimale vormgeving
   - "less is more"
   - stem vormgeving af op de doelgroep
   - iedere extra informatie-eenheid op het scherm doet de relatieve zichtbaarheid afnemen
10. help en documentatie
    - liever niet nodig
    - overzichtelijk
    - stappenplan

niet te veel informatie

2 Testscenario's

Scenario 1: Taak factureren (A)	
Beginsituatie:	Gebruiker is ingelogd Taakscherm geopend Taak afkomstig van niet all-inclusive project Taak heeft willekeurige status Taak niet gefactureerd
Uitvoer:	Gebruiker drukt op de factureer icoon Gebruiker selecteert factuur Gebruiker drukt op bevestigen
Resultaat:	Het factuurscherm wordt geopend met daarin het betreffende item aangemaakt.

Scenario 2: Taak factureren (B)	
Beginsituatie:	Gebruiker is ingelogd Taakscherm geopend Taak afkomstig van een all-inclusive project Taak heeft willekeurige status Taak niet gefactureerd
Uitvoer:	Gebruiker drukt op de factureer icoon
Resultaat:	Foutmelding: Kan taak niet factureren, omdat het afkomstig is van een all-in project

Scenario 3: Materiaal factureren (A)	
Beginsituatie:	Gebruiker is ingelogd Materiaalscherm geopend Materiaalafkomstig van niet all-inclusive project Materiaalheeft willekeurige status Materiaal niet gefactureerd
Uitvoer:	Gebruiker drukt op de factureer icoon Gebruiker selecteert factuur Gebruiker drukt op bevestigen
Resultaat:	Het factuurscherm wordt geopend met daarin het betreffende item aangemaakt.

Scenario 4: Materiaal factureren (B)	
Beginsituatie:	Gebruiker is ingelogd Materiaalscherm geopend Materiaalafkomstig van all-inclusive project Materiaalheeft willekeurige status Materiaal niet gefactureerd
Uitvoer:	Gebruiker drukt op de factureer icoon Gebruiker selecteert factuur Gebruiker drukt op bevestigen
Resultaat:	Foutmelding: Kan taak niet factureren, omdat het afkomstig is van een all-in project

Scenario 5: Project factureren (A)	
Beginsituatie:	Gebruiker is ingelogd Projectscherm geopend Project afkomstig van niet all-inclusive project Project heeft willekeurige status Project niet geheel gefactureerd Project heeft ongefactureerde items
Uitvoer:	Gebruiker drukt op de factureer icoon Gebruiker selecteert factuur Gebruiker drukt op bevestigen
Resultaat:	Het factuurscherm wordt geopend met daarin het betreffende item aangemaakt.

Scenario 6: Project factureren (B)	
Beginsituatie:	Gebruiker is ingelogd Projectscherm geopend Project is all-in Project heeft willekeurige status Project niet geheel gefactureerd Project heeft ongefactureerde items
Uitvoer:	Gebruiker drukt op de factureer icoon Gebruiker selecteert factuur Gebruiker drukt op bevestigen
Resultaat:	Het factuurscherm wordt geopend met daarin het betreffende item aangemaakt.

Scenario 7: Factuur verwijderen (A)

Beginsituatie:	Gebruiker is ingelogd Factuurscherm geopend Factuur niet verzonden
Uitvoer:	Gebruiker drukt op de verwijder icoon Gebruiker drukt op 'Ja'
Resultaat:	Factuur verwijderd uit systeem

Scenario 8: Factuur verwijderen (B)	
Beginsituatie:	Gebruiker is ingelogd Factuurscherm geopend Factuur is verzonden
Uitvoer:	Gebruiker drukt op de verwijder icoon Gebruiker drukt op 'Ja'
Resultaat:	Foutmelding: kan de factuur niet verwijderen! Factuur is reeds verzonden.

Scenario 9: Factuuritem verwijderen (A)	
Beginsituatie:	Gebruiker is ingelogd Factuuritemscherm geopend Factuur niet verzonden
Uitvoer:	Gebruiker drukt op de verwijder icoon Gebruiker drukt op 'Ja'
Resultaat:	Factuuritem verwijderd!

Scenario 10: Factuuritem verwijderen (B)	
Beginsituatie:	Gebruiker is ingelogd Factuuritemscherm geopend Factuur verzonden

Uitvoer:	Gebruiker drukt op de verwijder icoon Gebruiker drukt op 'Ja'
Resultaat:	Foutmelding: Kan het factuuritem niet verwijderen! Factuur is reeds verzonden.

3 Resultaten 1e test

Nummer	Beschrijving
1.	Verzenden factuur lukt niet! Foutmelding geconstateerd: Generatie van factuurnummer niet geslaagd.
2.	Consistentie gebruik iconen geschonden.  wordt gebruikt voor diverse functionaliteiten.
3.	 Dit icoon brengt een verkeerde associatie met zich mee. Doet denken aan de refresh funcionaliteit en niet aan terug.
4.	Het verzenden van facturen werkt nog niet.
5.	Facturen kunnen niet worden gezocht aan de hand van een meegegeven periode, dus start -en einddatum.
6.	

Resultaten 2e test

Alle functionaliteiten waren naar wens van de testpersonen

Pilotontwikkelplan Beheer

Naam document:	Pilotontwikkelplan 'Beheer'
Bestandsnaam:	Pilot_beheer_v1.doc
Project:	De ontwikkeling van de modules 'Facturatie' en 'Beheer'
Datum oplevering:	04/12/2003
Omschrijving van de inhoud:	Dit document beschrijft het ontwerp van de pilot 'Beheer'.
In opdracht van:	Metrix B.V.
Versie:	1.00
Auteur:	Ruben Maas

Voorwoord

In dit rapport wordt het ontwerp beschreven die wordt gebruikt bij de ontwikkeling van de pilot ‘Beheer’. Dit document is opgesteld in nauwe samenwerking met de opdrachtgever en eindgebruikers.

Inhoudsopgave

1	**Inleiding**	**4**
2	**Plan van aanpak**	**5**
3	**Functionele-Structuur**	**7**
4	**Ontwerp presentatielaag**	**9**
5	**Ontwerp Businesslaag**	**14**
6	**Ontwerp Datalaag**	**16**
7	**Aanpassingen externe componenten**	**21**
8	**Bouweenheden**	**22**
9	**Invoering**	**23**

1 Inleiding

In dit document staat de fase pilotontwikkeling beschreven voor wat betreft de pilot 'Facturatie'. In hoofdstuk 2 zal het plan van aanpak worden beschreven, waarin de activiteiten zullen worden beschreven en ingepland.

Hoofdstuk 3 wijd ik aan het beschrijven van de functionele structuur waarin de functionaliteit op ondubbelzinnige wijze staat beschreven. Vervolgens volgt de technische structuur onder verdeelt in een drietal onderdelen namelijk:

- Ontwerp Presentatielaag
- Ontwerp Businesslaag
- Ontwerp Datalaag

In hoofdstuk 7 staan de eventuele aanpassingen aan externe componenten beschreven en tenslotte in hoofdstuk 8 staat een beschrijving en planning van de bouweenheden.

2 Plan van Aanpak

2.1 Doelstelling

Voor deze pilot heb ik een doelstelling geformuleerd, die luidt als volgt:
De pilot 'Beheer' moet ervoor zorgdragen dat de gebruiker de mogelijkheid heeft systeeminstellingen vanuit het Key-Task systeem te bewerken.

2.2 Op te leveren producten

De pilot 'Beheer' voorziet in de volgende producten:

- Het document pilotontwikkelplan 'Beheer'
- De module 'Beheer'
- Handleiding 'Module Beheer'
- Minimaal 1 testrapport (*Afhankelijk van het aantal iteraties*)

2.3 Uit te voeren activiteiten

De volgende activiteiten worden in deze fase uitgevoerd:

- Beschrijven functionele structuur
- Beschrijven technische structuur
- Beschrijven database
- Beschrijven bouwdelen

Functionele structuur
Deze activiteit gebruiken ik om de functionaliteit van deze pilot in kaart te brengen.

Technische structuur
Binnen deze activiteit wordt beschreven hoe het systeem zal worden opgebouwd.
De technische structuur is opgedeeld is drie delen, namelijk:

- Ontwerp Presentatielaag
- Ontwerp Businesslaag
- Ontwerp Datalaag

Beschrijven database
Aan de hand van de bovenstaande activiteiten kan ik bepalen welke toevoegingen er verricht moeten worden aan de Key-Task database

Beschrijven bouwdelen
Mede aan de hand van de technische structuur zal ik de pilot in verschillende bouwdelen opdelen.

2.4 Planning

De planning bestaat uit in totaal negen weken (45 werkdagen) waarin de diverse activiteiten, die IAD voorschrijft, staan ingepland. De eerste drie weken staan ingepland voor het maken van het pilotontwikkelplan. Vijf weken staan er ingepland voor de realisatie van de pilot en tenslotte één week voor her maken van de handleiding.
Het testen en de invoering zijn meegenomen in de bouwfase.

Pilot 'Beheer'									
Stel Plan van aanpak op									
Bereid pilotontwerp-workshop voor									
Voer pilotontwerp-workshop uit									
Beschrijven Functionele structuur									
Beschrijven ontwerp presentatielaag									
Beschrijven ontwerp businesslaag									
Beschrijven ontwerp datalaag									
Beschrijven ontwerp database									
Beschrijven Bouweenheden									
Beschrijven testprocedure									
Beschrijven aanpassingen externe componenten									
Bouw pilot									
Maak handleiding									

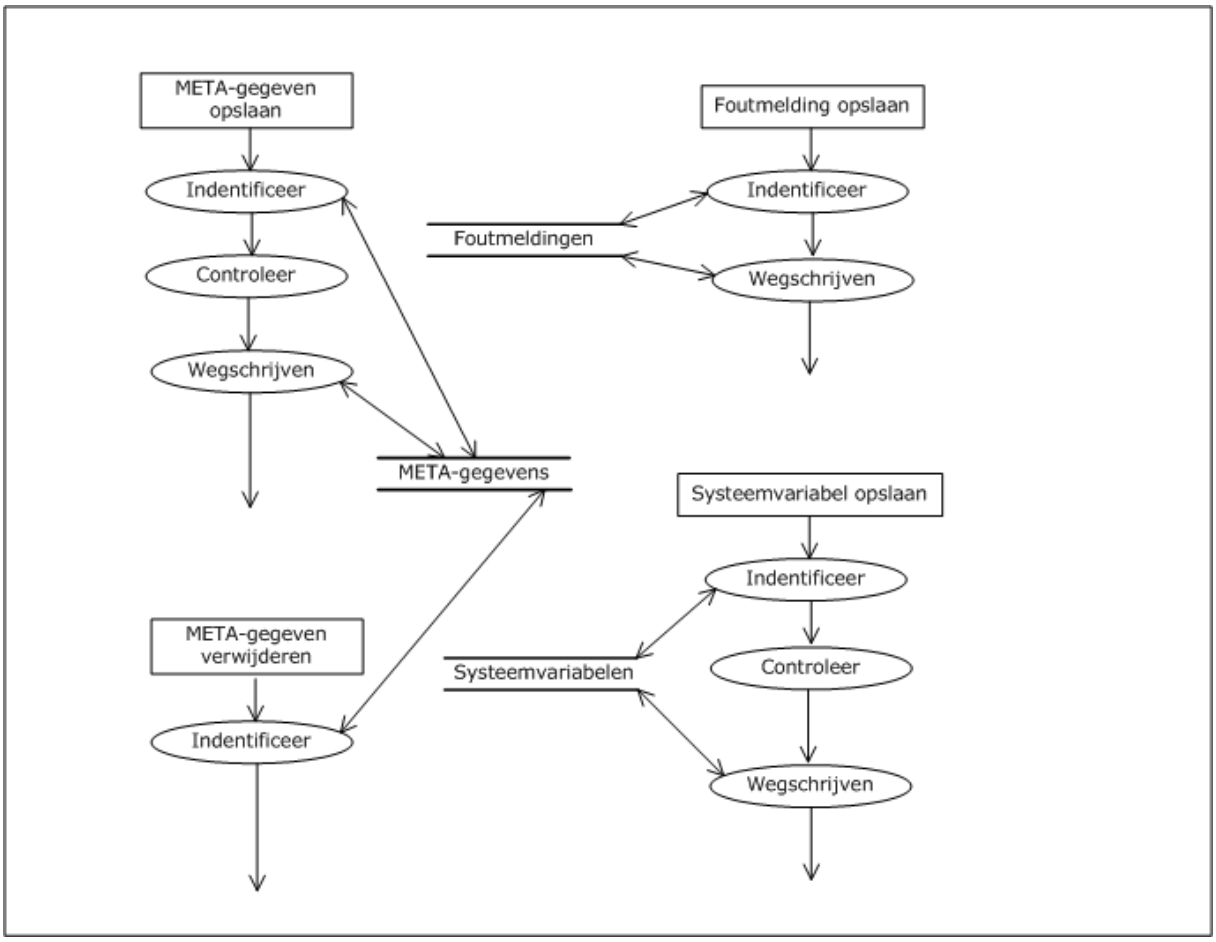
2.1 Planning pilot 'Beheer'

3 Functionele structuur

Binnen dit hoofdstuk zal vastgelegd worden over welke functionaliteiten deze pilot moet beschikken. De functionaliteiten worden bepaald aan de hand van de systeemeisen die staan beschreven in de definitiestudie.

Over de volgende functionaliteiten moet de pilot 'Beheer' beschikken:

- META-gegeven aanmaken en bewerken
- Gebruiker aanmaken en bewerken
- Foutmelding aanmaken en verwerken
- Systeemvariabelen bewerken



Figuur 3.1 DFD: functionele structuur pilot 'Beheer'

Om duidelijkheid te scheppen in de vraag welke instellingen er nou precies horen bij welke categorie volgt er nu een overzicht:

META-gegevens	
Naam:	Bron:
Project status	Tbl: Action_status_codes
Taak status	Tbl: action_time_status_codes
Material status	Tbl: action_expense_status_codes
Project type	Tbl: action_type
Klant type	Tbl: company_address_type
Contact type	Tbl: company_contact_type
Prioriteit	Tbl: priority
Landen	Tbl: countries
BTW codes	Tbl: tax
Lijncodes	Tbl: line_codes

Medewerkers type	Tbl: employee_type

Systeemvariabelen	
Naam:	Bron:
MAX_RESULTS	
SESSION_TIME_OUT	
MAX_USERS	
MAX_USER_SESSIONS	
FILE_LOCATION	
SYSTEM_LOGO	
OBJECT_LIBRARY	
SERVER_LOCATION	
ORGANISATION_NAME	
ORGANISATION_LOGO	
ORGANISATION_SMTP	
ORGANISATION_EMAIL	
DEF_PROJECT_SUBJECT	
DEF_TASK_SUBJECT	
DEF_MATERIAL_SUBJECT	
DEF_CALENDAR	
DEF_TEAM	
SERVER_DATE	
SERVER_TIME	

De foutmelding staan opgeslagen in tabel: **system_error_messages**

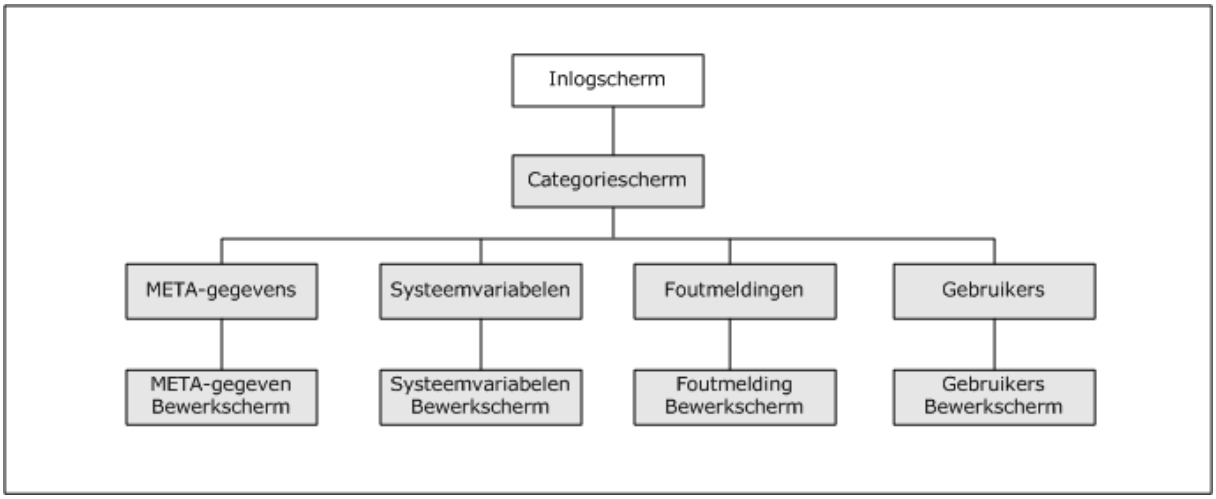
De gebruikerssinstellingen staan in tabel: **employees**

4 Ontwerp presentatielaag

Binnen dit hoofdstuk wordt beschreven hoe de presentatielaag eruit komt te zien. Dit wordt bewerkstelligd aan de hand van een navigatieschema, schermontwerp en de te hanteren iconen en kleuren.

4.1 Navigatieschema

Om te kunnen bepalen welke schermen er dienen te worden ontwikkeld, heb ik een navigatieschema opgesteld.



Figuur 2.4 navigatieschema

Bij het navigatieschema geldt dat de entiteiten die niet grijs getint zijn al reeds zijn ontwikkeld en onderdeel uitmaken van het huidige Key-Task systeem. De volgende pagina's moeten dus worden ontwikkeld:

- Hoofdscherm
- META-Gegevens instantiescherm
- META-Gegevens bewerkscherm
- Systeemvariabelen instantiescherm
- Systeemvariabelen bewerkscherm
- Foutmeldingscherm
- Foutmelding bewerkscherm
- Gebruikersscherm
- Gebruikers bewerkscherm

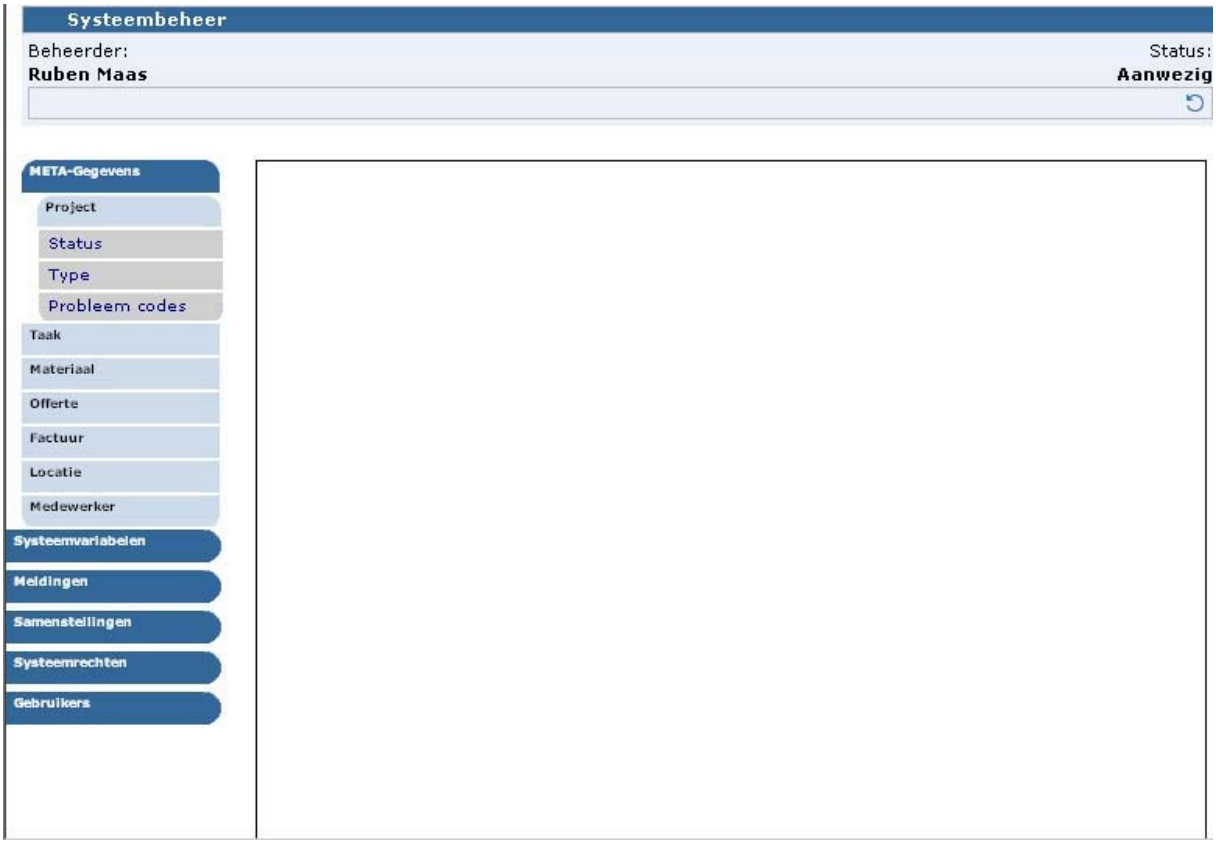
In *hoofdstuk 4.2* zullen de te ontwerpen schermen worden beschreven.

4.2 Schermontwerp

Binnen dit hoofdstukken wordt het schermontwerp besproken. Het schermontwerp bestaat uit een statische HTML-pagina van de te ontwikkelen pagina.

Hoofdscherm

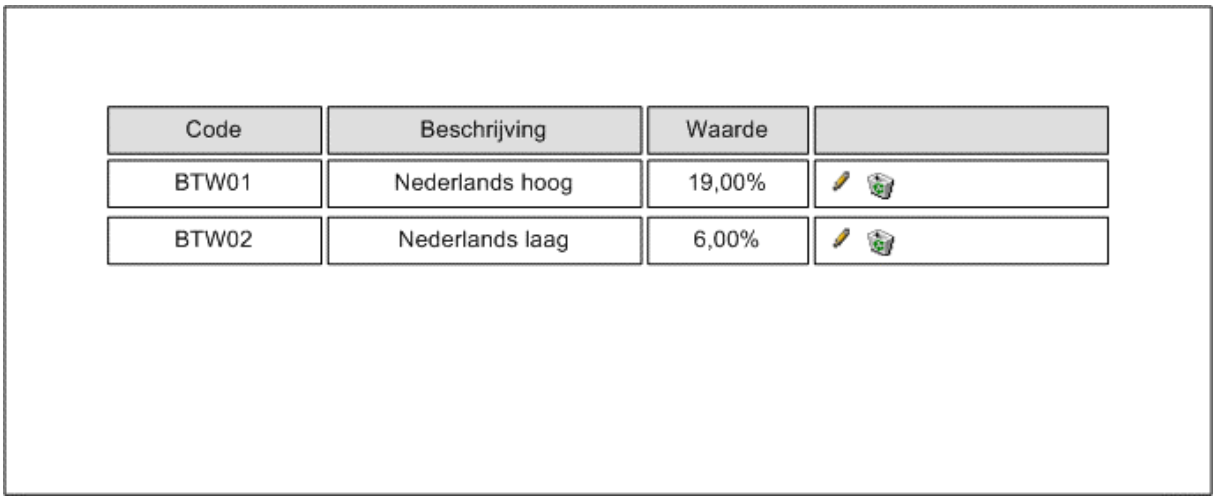
Het hoofdscherm heeft als doel de gebruiker voor de keuze te stellen welke instelling hij of zij wil gaan inzien of wijzigen.



Figuur 2.4 Hoofdscherm

Het hoofdscherm bestaat uit een menu waarin zich de diverse categorieën in bevinden en een zogenaamd iframe (inline frame) waarin zich de bewerkscherm in zullen bevinden.

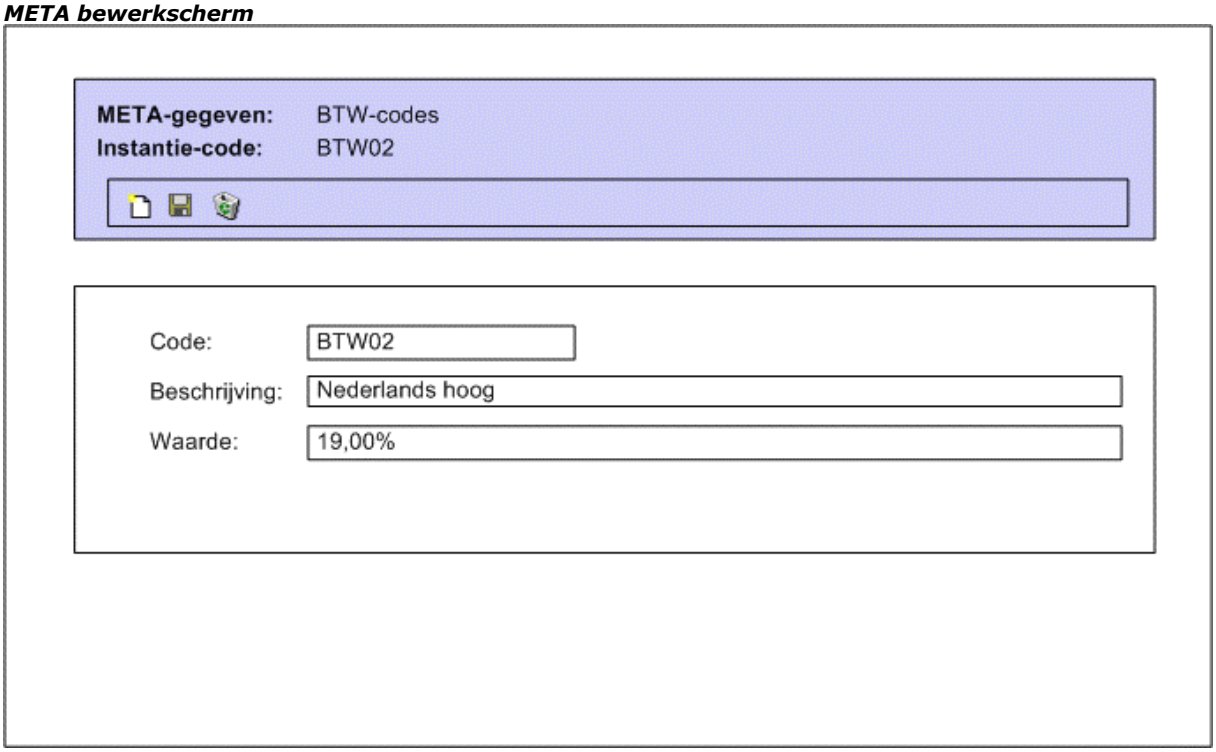
META instantiescherm



Binnen het factuurscherm is gekozen voor de volgende iconen:

	META gegeven bewerken
	META gegeven verwijderen

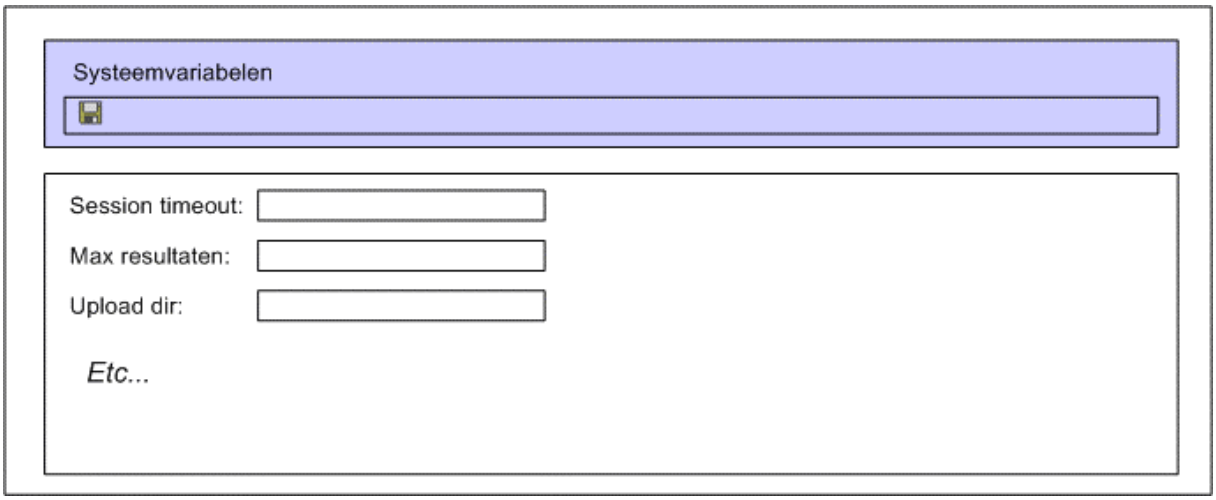
META bewerkscherm



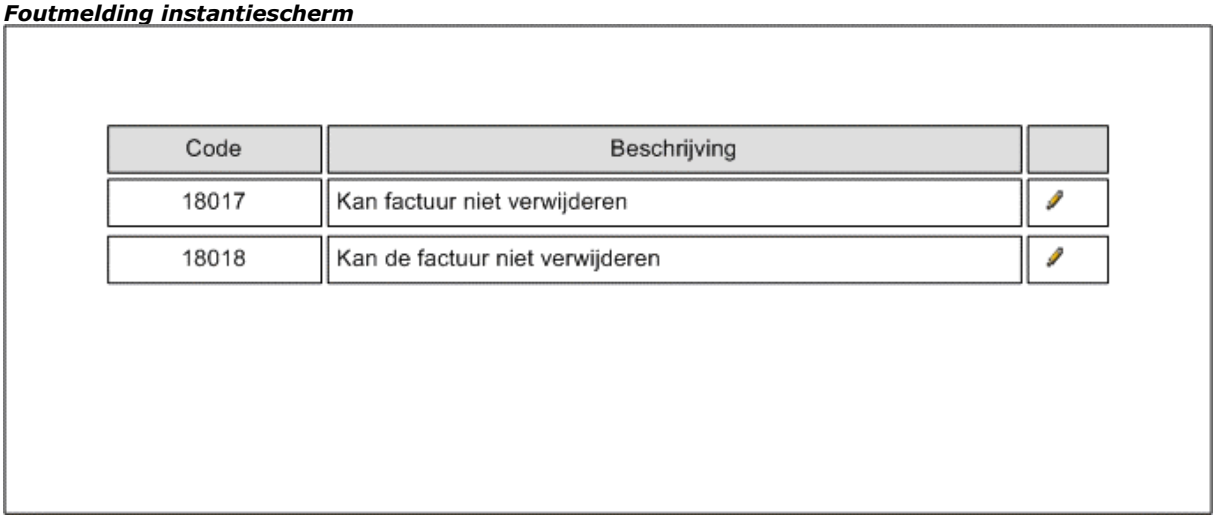
Binnen het META bewerkscherm is gekozen voor de volgende iconen:

	META gegeven aanmaken
	META gegeven verwijderen
	META gegeven opslaan

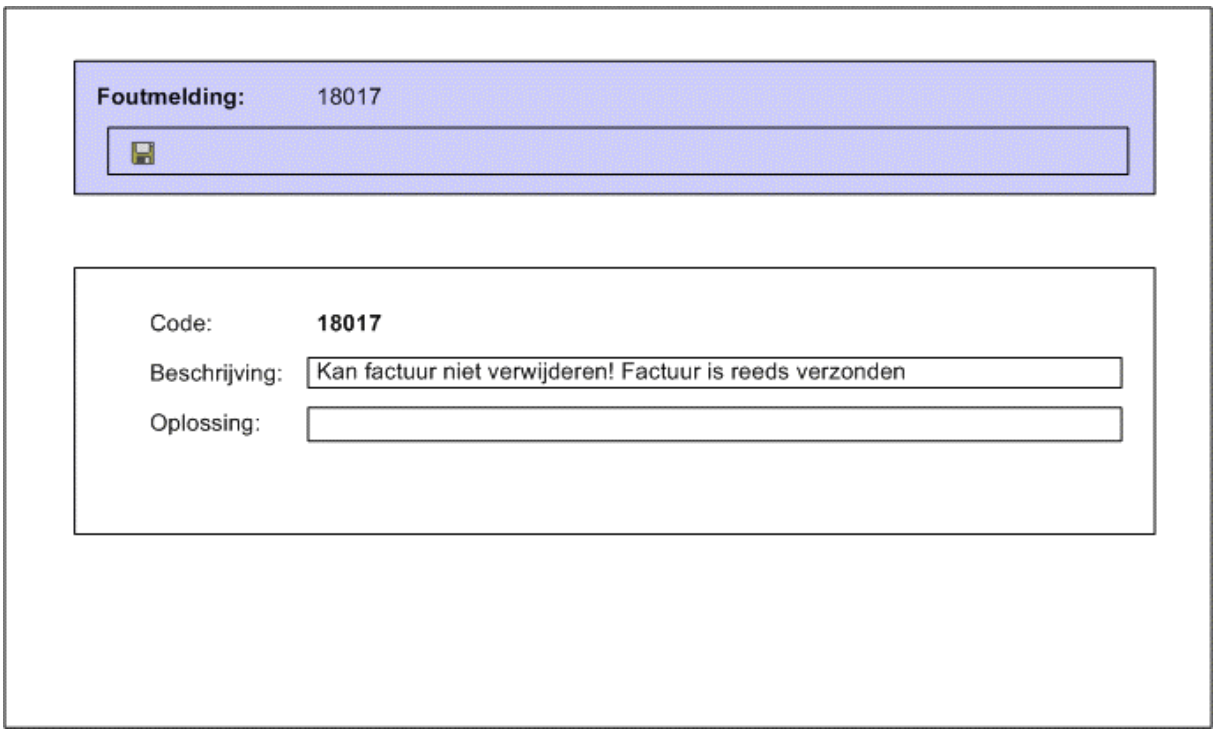
Systeemvariabelen bewerkscherm



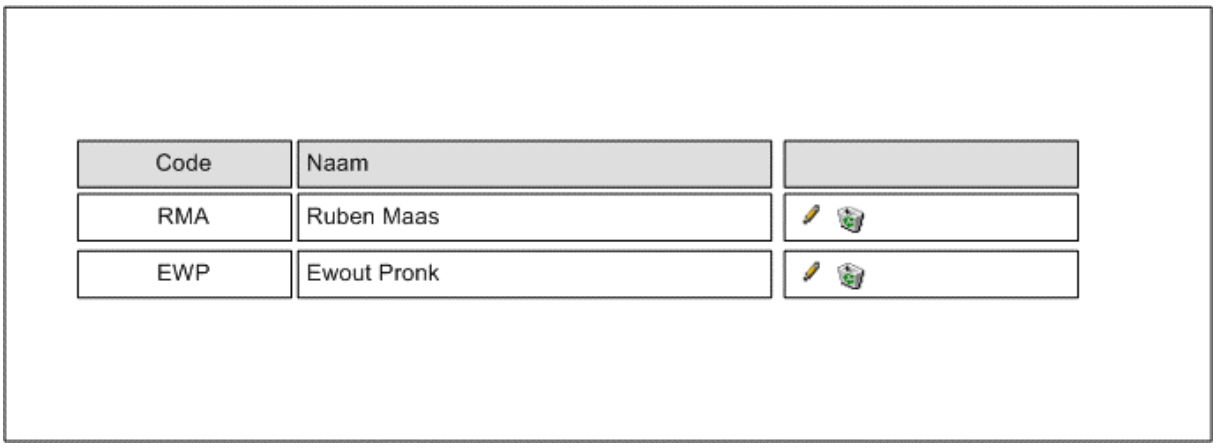
Foutmelding instantiescherm



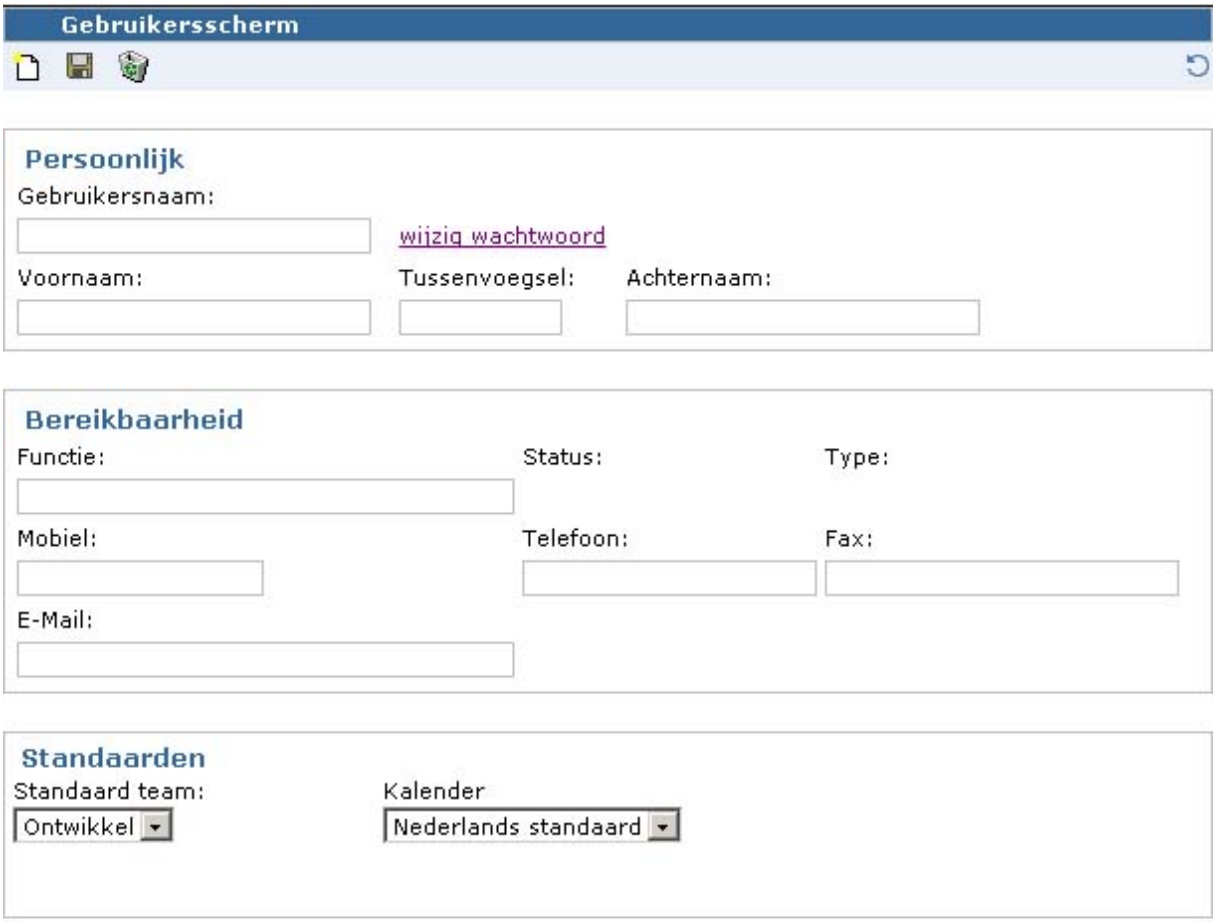
Foutmelding bewerkscherm



Gebruiker instantiescherm

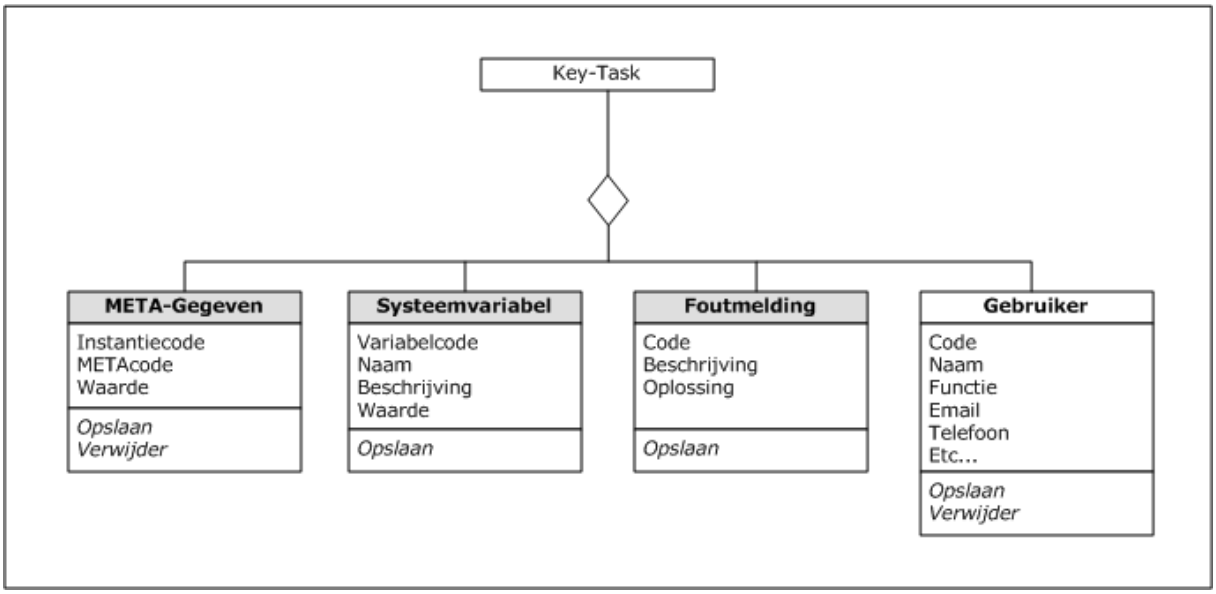


Gebruikers bewerkscherm



5 Ontwerp businesslaag

5.1 OMT



De volgende eigenschappen zijn van toepassing tot dit object:

Eigenschap	Datatype	Beschrijving
Instantiecode	Numeriek	Deze code geeft onderscheid het betreffende meta-gegegeven. Aan de hand van deze code weet het systeem welke tabel er geupdate moet worden en wat voor een integriteitcontroles er moeten worden uitgevoerd.
METAcode	Tekst	Deze code geeft het betreffende META record aan.
Aangemaakt	Datum/Tijd	Bevat de datum waarop het object is gecreeerd
AangemaaktDoor	Employee	Bevat een medewerker object van degene die de activiteit heeft aangemaakt
Waarde	Variant	Bevat de waarde van het META-gegegeven

	Deze eigenschappen kun je alleen lezen

Procedures	Resultaat	Beschrijving
Bewaren	Boolean	Zorgt ervoor dat de waardes omtrent dit object worden weggeschreven naar de database. Deze procedure geeft true terug indien geslaagd.
Verwijderen	Boolean	Zorgt ervoor dat het betreffende record verwijderd wordt uit de database

Het 'Systeemvariabel' Object

De volgende eigenschappen zijn van toepassing tot dit object:

Eigenschap	Datatype	Beschrijving
Variabelcode	Numeriek	Deze code identificeert een bepaalde variabelen
Naam	Tekst	Bevat de naam van de variabelen.
Beschrijving	Datum/Tijd	Bevat de beschrijving van de variabelen
Waarde	Variant	Bevat de waarde van de variabelen

	Deze eigenschappen kun je alleen lezen

Procedures	Resultaat	Beschrijving
Bewaren	Boolean	Zorgt ervoor dat de waardes omtrent dit object worden weggeschreven naar de database. Deze procedure geeft true terug indien geslaagd.

Het 'Foutmelding' Object

De volgende eigenschappen zijn van toepassing tot dit object:

Eigenschap	Datatype	Beschrijving
Code	Numeriek	Deze code identificeert een bepaalde variabelen
Beschrijving	Tekst	Bevat de beschrijving van de foutmelding
Oplossing	Tekst	Bevat een mogelijke oplossing voor de betreffende foutmelding

	Deze eigenschappen kun je alleen lezen

Procedures	Resultaat	Beschrijving
Bewaren	Boolean	Zorgt ervoor dat de waardes omtrent dit object worden weggeschreven naar de database. Deze procedure geeft true terug indien geslaagd.

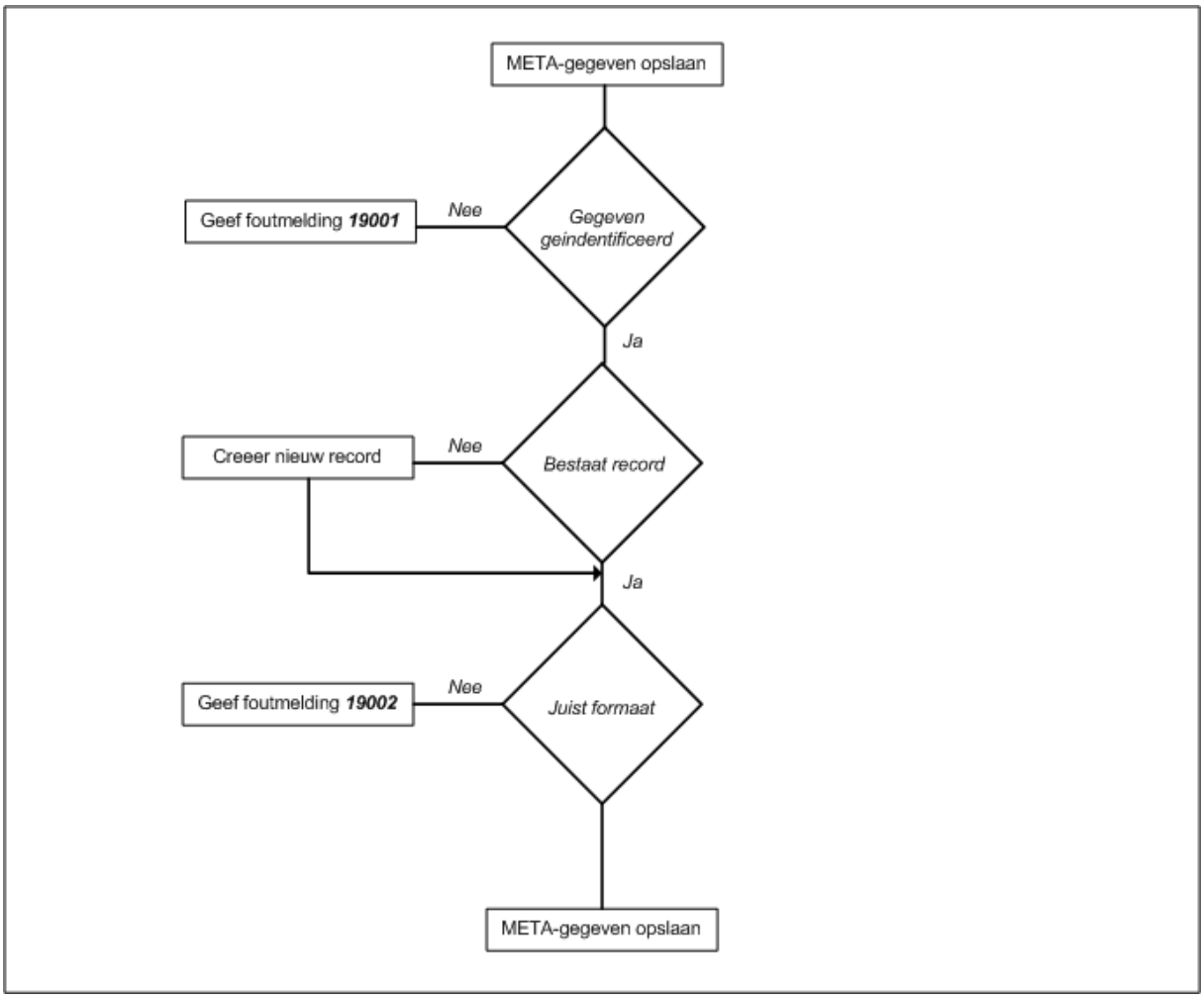
6 Ontwerp datalaag

Binnen dit hoofdstuk zal de datalaag worden ontworpen. De datalaag bestaat uit tal van procedures die de integriteit van de database bewaken. De procedures worden beschreven aan de hand van beslissingsmodellen.

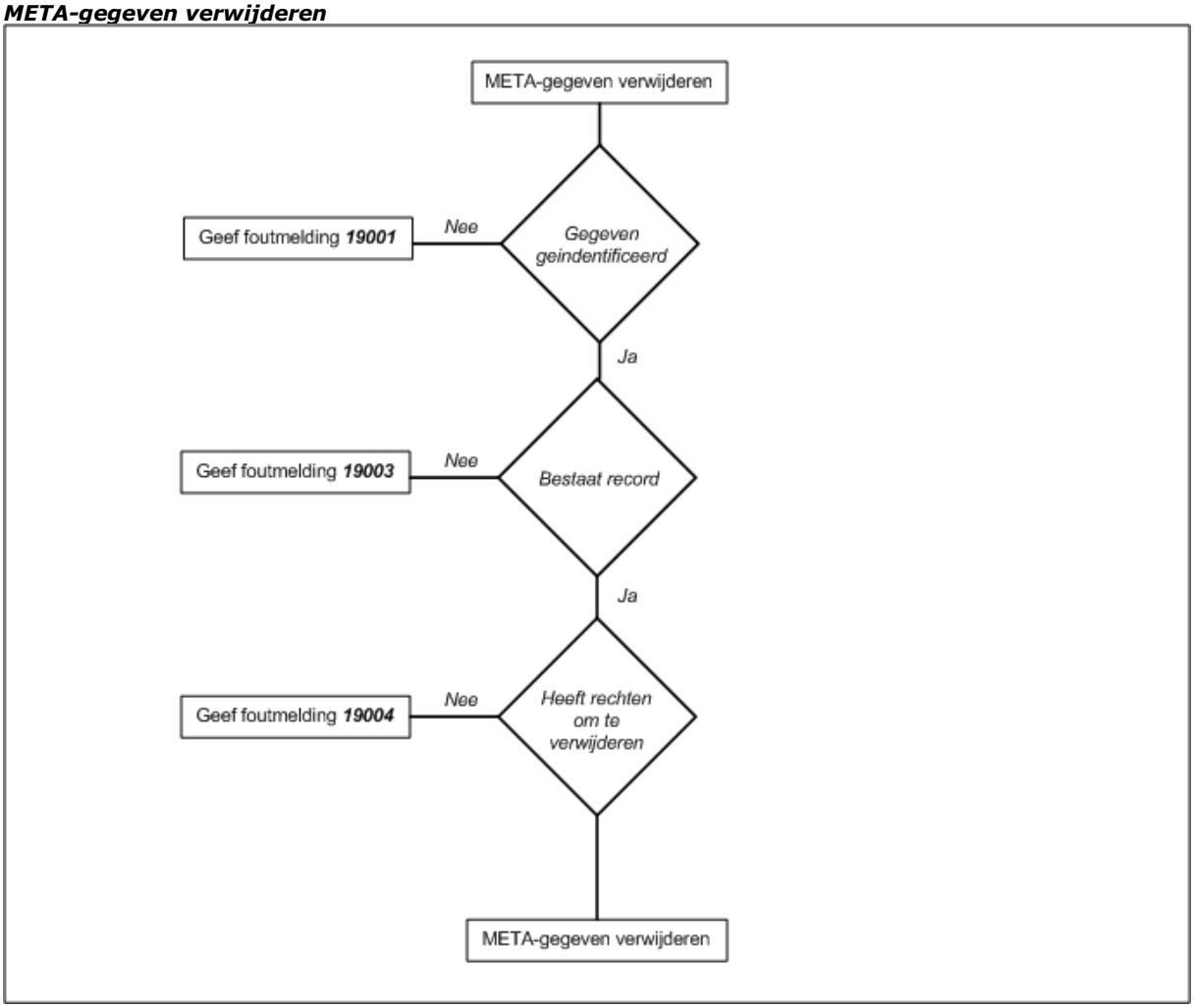
Nu de businesslaag is ontworpen kan ik bepalen welke integriteitcontroles er uitgevoerd dienen te worden. De volgende controles moeten worden uitgevoerd:

- META-gegeven opslaan
- META-gegeven verwijderen
- Systeemvariabel opslaan
- Foutmelding opslaan

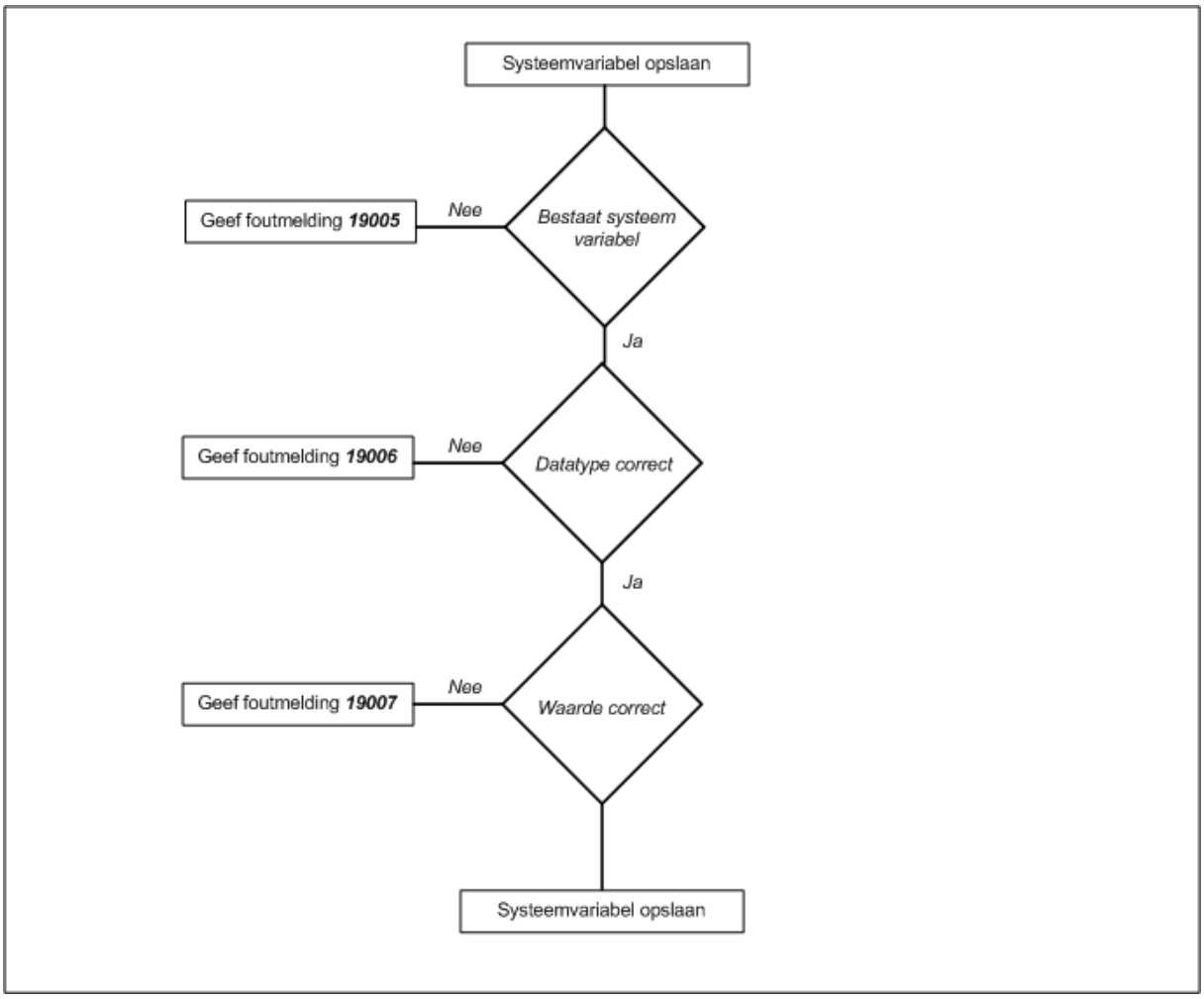
META-gegeven opslaan



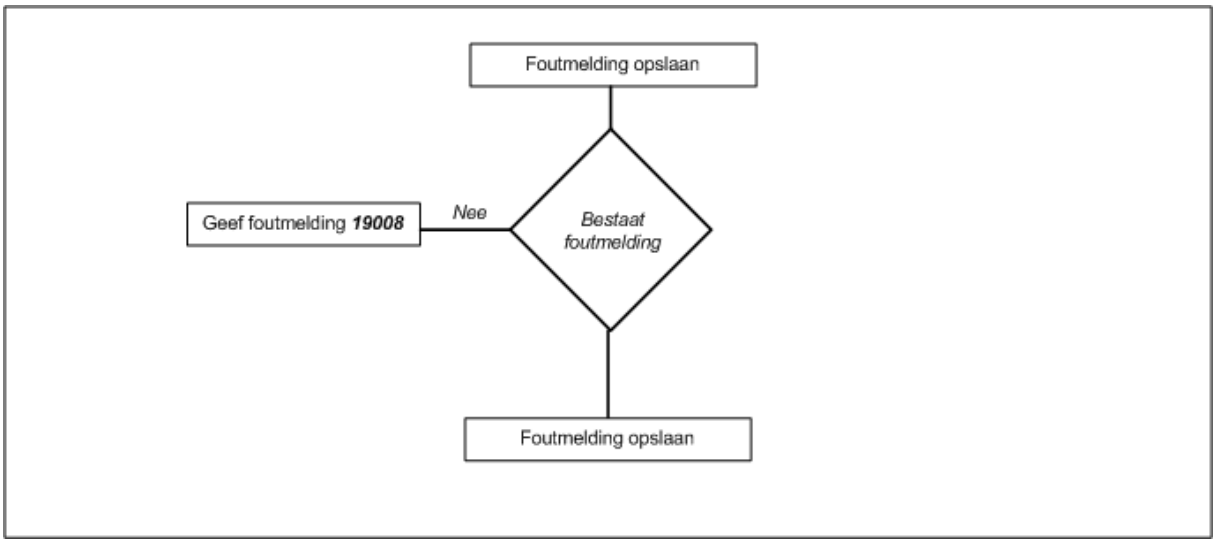
META-gegeven verwijderen



Systeemvariabel opslaan



Foutmelding opslaan



7 Aanpassingen externe componenten

Binnen dit hoofdstuk zal ingegaan worden in de aanpassingen die verricht moeten worden aan externe componenten van deze module/pilot. U kunt hierbij denken aan het toevoegen van knoppen in diversen bestaande schermen van Key-Task.

De volgende aanpassingen dienen verricht te worden:

Bron	Beschrijving
menu.asp	Verwijzing naar beheer-module opnemen

8 Bouweenheden

Binnen dit hoofdstuk staan de bouweenheden beschreven en staan deze ingepland.

8.1 bouweenheden

Numme	Naam	Code	Prioriteit
1.	Realisatie Datalaag	BE01	Hoog
2.	Realisatie Businesslaas	BE02	Hoog
3.	Realisatie Hoofdscherm	BE03	Hoog
4.	Realisatie META-gegevensschermen	BE04	Hoog
5.	Realsatie Systeemvariabelenschermen	BE05	Gemiddeld
6.	Realsatie Gebruikersschermen	BE06	Gemiddeld
7.	Realisatie Foutmeldingschermen	BE07	Laag
8.	Realisatie Aanpassingen externe componenten	BE08	Laag

8.2 Time-boxing

Code	Geschatte tijd	Werkelijke tijd
BE01	2 dagen	2 dagen
BE02	1 dag	1,5 dag
BE03	1 dag	1 dag
BE04	1 dag	1,5 dag
BE05	1 dag	1 dag
BE06	2 dagen	2 dagen
BE07	0,5 dag	0,5 dag
BE08	0,5 dag	0,5 dag

9 Invoering

De procedures zijn overwogen ten aanzien van de volgende punten.

- Hardware
- Software
- Opleiding
- Handleidingen

In de volgende paragrafen zal worden beschreven hoe de verantwoordelijkheden zijn per procedure en hoe de uiteindelijke procedure eruit ziet.

9.1 Verantwoordelijkheden invoering

Hieronder zullen per punt de verantwoordelijkheden van elke invoeringsprocedure beschreven worden.
De betrokkenen bij de invoering zijn:

- Ontwikkelaars (het verkopende bedrijf)
- Ondersteuners (helpdesk en dergelijke)
- Eindgebruikers
- Opdrachtgever

Hardware

Voor de hardware ligt de verantwoording bij de eindgebruikers. De hardware zal moeten voldoen aan de eisen zoals deze omschreven zijn in Hoofdstuk 7.1 De technische Architectuur van de hardware in de definitiestudie.
De Ontwikkelaars kunnen wel adviseren over de aan te schaffen hardware.

Software

Voor de software ligt de verantwoording ook bij de opdrachtgever. De software zal moeten voldoen aan de eisen zoals deze omschreven zijn in Hoofdstuk 7.1 De technische Architectuur van de hardware in de definitiestudie. In dit hoofdstuk staat tevens genoemd aan welke eisen de software moet voldoen.

Opleiding

De verantwoording voor de opleiding van de verschillende gebruikers ligt in eerste instantie bij de ondersteuners. De ondersteuners worden opgeleid door een consultant die van het verkopende bedrijf uitgaat. Daarnaast zullen de ontwikkelaar een handleiding schrijven dat beschrijft hoe het product werkt. Van de ondersteuners wordt verwacht dat zij deze handleiding goed kennen.

3.1.1 Handleidingen

De handleidingen zijn de verantwoording van de ontwikkelaars. Zij zullen mede in het kader van de opleiding een handleiding schrijven die de werking van het product omschrijft. De ondersteuners binnen het bedrijf kunnen zelf toevoegingen op deze handleiding schrijven. Echter zullen deze toevoegingen ook de bouwers moeten toekomen om deze extra informatie toe te kunnen voegen aan de handleidingen zoals deze zijn aangeleverd om zo een betere handleiding in de toekomst te kunnen uitleveren.

Testrapport pilot 'Beheer'

Naam document:	Testrapport pilot beheer
Bestandsnaam:	Testrapport_pilot_beheer_it_0.doc
Datum oplevering:	29/12/2003
Iteratie:	0
Omschrijving van de inhoud:	Dit document beschrijft de aanpak met betrekking tot de gebruikerstest iteratie 0 pilot 'Beheer'
In opdracht van:	Metrix B.V.
Auteur:	Ruben Maas

Inhoudsopgave

1 Inleiding 4

2 Plan van aanpak 5

2.1 Doel 5
2.2 De testmethode 5
2.3 Beschrijving 5
2.4 Toepasbaarheid 5
2.5 Voorwaarden 5
2.6 Verwerking resultaten 5
2.7 Deelnemers 5
2.8 Werkwijze 6
2.9 Hulpmiddelen 6

3 Resultaten 7

1 Inleiding

Dit document beschrijft de aanpak van het Heuristic Evaluation testen in het kader van de pilot ‘’Beheer’. In hoofdstuk 2 zal ik beschrijven hoe ik het testen ga aanpakken. De Resultaten staan beschreven in hoofdstuk 3.

2 Plan van aanpak

2.1 Doel

Het vroegtijdig onderkennen van problemen wat betreft de usability van de pilot ‘Beheer’.

2.2 De testmethode

De testmethode die gehanteerd zal gaan worden is Heuristic Evaluation testen.
De usability van de pilot ‘Beheer’ wordt getest door een aantal experst.

2.3 Beschrijving

Een groep experts (2-5) analyseren de usebility van de pilot. Aan de hand van de bevindingen van de experts wordt er bepaald welke aanpassingen er verricht moeten worden.

2.4 Toepasbaarheid

Deze test is toepasbaar indien de fase pilotontwikkeling doorlopen is. Deze test kan meerdere keren worden uitgevoerd, doordat de fase pilotontwikkeling meerdere iteraties kan hebben.

2.5 Voorwaarden

Er zijn tussen de 2 en 5 testpersonen aanwezig.
Er zijn geen scenario’s beschikbaar.

2.6 Verwerking resultaten

De deelnemers doen schriftelijk verslag van de test en deze worden vervolgens besproken met de ontwikkelaar. De ontwikkelaar staat vervolgens voor de keuze om aanpassingen te verrichten op de pilot ‘Beheer’. Mochten er één of meerdere aanpassingen verricht worden dan komt de fase pilotontwikkeling in een nieuwe iteratie. Dit betekent dat er ter aller tijde alsnog een nieuwe test wordt uitgevoerd.

2.7 Deelnemers

- Jasper Schut
- Frans Mientjes

- Dimitri Stoikof

2.8 Werkwijze

1. Deelnemers analyseren gedurende circa een uur de usability van de pilot ‘Beheer’
2. Bevindingen verzamelen en ordenen.
3. Bevindingen overdragen en bespreken.
4. Aandragen van verbeteringen.

2.9 Hulpmiddelen

Er wordt gebruik gemaakt van tien usability "heuristics" (controle punten):

1. Eenvoudige en natuurlijke dialoog (spreek de taal van de gebruiker) het systeem moet de taal van de gebruiker spreken, met woorden, zinnen en concepten waar de gebruiker mee vertrouwd is
   - voorkom systeem georiënteerde termen
   - presenteer informatie in een natuurlijke en logische volgorde
2. minimaliseer druk op geheugen van gebruiker
   - herkennen is beter dan herinneren
   - zorg dat men zo weinig mogelijk hoeft te onthouden
3. wees consistent
   - gebruikers moeten zich niet hoeven afvragen of verschillende woorden, situaties of acties hetzelfde betekenen
   - gebruik standaards
4. geef feedback
   - het systeem moet altijd aangeven wat er gebeurt of wat er is gebeurd
5. geef duidelijk uitgangen aan
   - gebruikers kiezen vaak verkeerde functies en hebben dan een duidelijke "nooduitgang" nodig
   - maak gebruik van "undo & redo"
6. wees flexibel en efficiënt
   - breng short cuts aan met name voor ervaren gebruikers
   - maak het mogelijk om veel gebruikte acties vast te leggen (voorbeeld rechter muisknop)
7. ontwerp goede foutboodschappen
   - help gebruikers bij het herstellen van fouten
8. voorkom fouten
   - nog beter dan een goed foutboodschap is het voorkomen van fouten
   - vraag je bij elke foutboodschap af: had deze fout vermeden kunnen worden?
9. esthetisch en minimale vormgeving
   - "less is more"
   - stem vormgeving af op de doelgroep
   - iedere extra informatie-eenheid op het scherm doet de relatieve zichtbaarheid afnemen
10. help en documentatie
    - liever niet nodig
    - overzichtelijk
    - stappenplan

niet te veel informatie

Resultaten

Bevindingen	Prioriteit
Het  doet denken aan de 'Refresh' functionaliteit	Laag
Je moetop de hele regel kunnen klikken bij het selecteren van een instantie	Gemiddeld
Schermindeling niet optimaal	gemiddeld

HANDLEIDING

do company

projectmanagement en advies

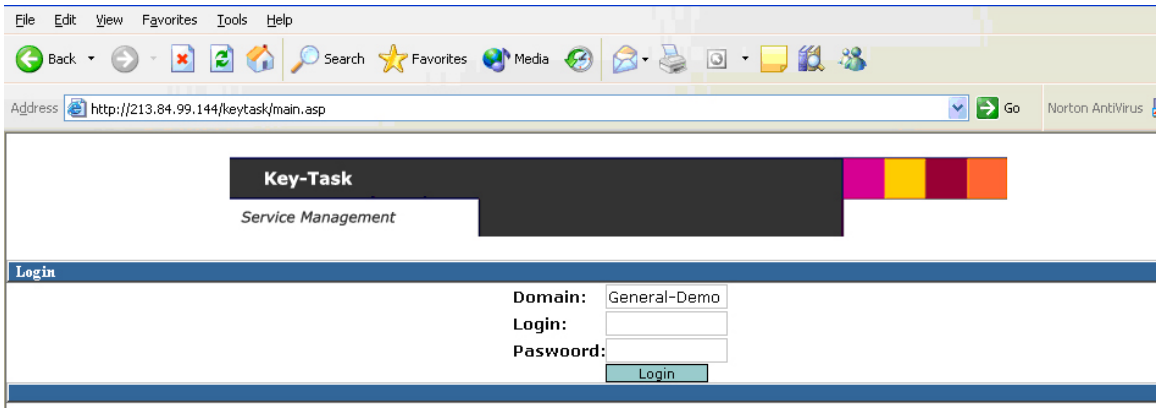
do tasks

Inhoud

1. Inloggen ***3***
2. Het aanmaken van een offerte ***4***
2.1 Het gebruiken van specificaties 9
2.2 Bestanden bijvoegen en bekijken 11
3. Het Project ***12***
3.1 Het factureren van afzonderlijke items 13
3.2 Het wijzigen/toevoegen van taken en materialen 16
3.2 Facturatie indicator 18
3.4 Het afsluiten van een project 19
4. Facturatie ***21***
4.1 Algemeen 21
4.2 Het factureren van een all-in project 21

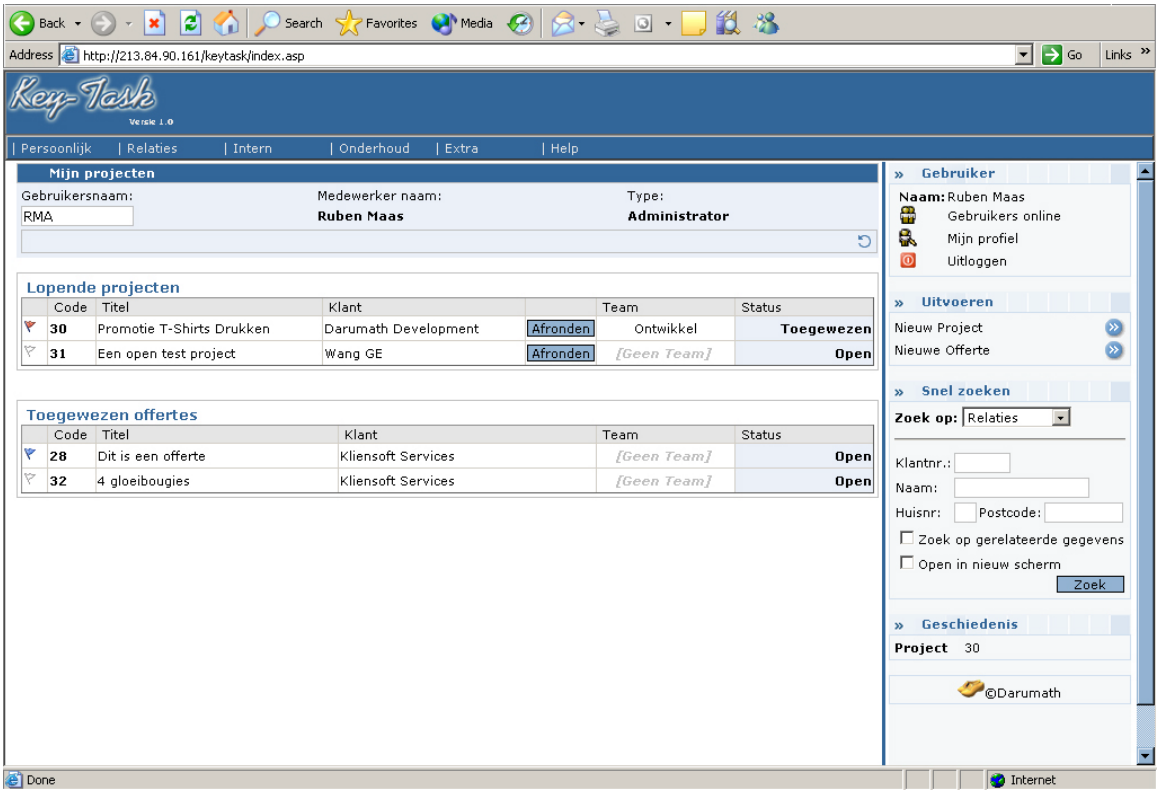
1. Inloggen

Voor het inloggen dient u met Internet Explorer naar het adres van de Do Tasks server te gaan. U krijgt het volgende scherm te zien.



U kunt de aan u verstrekte login informatie gebruiken om in te voeren in de velden “Login” en “Paswoord”. Vervolgens klikt u op de knop “Login”.

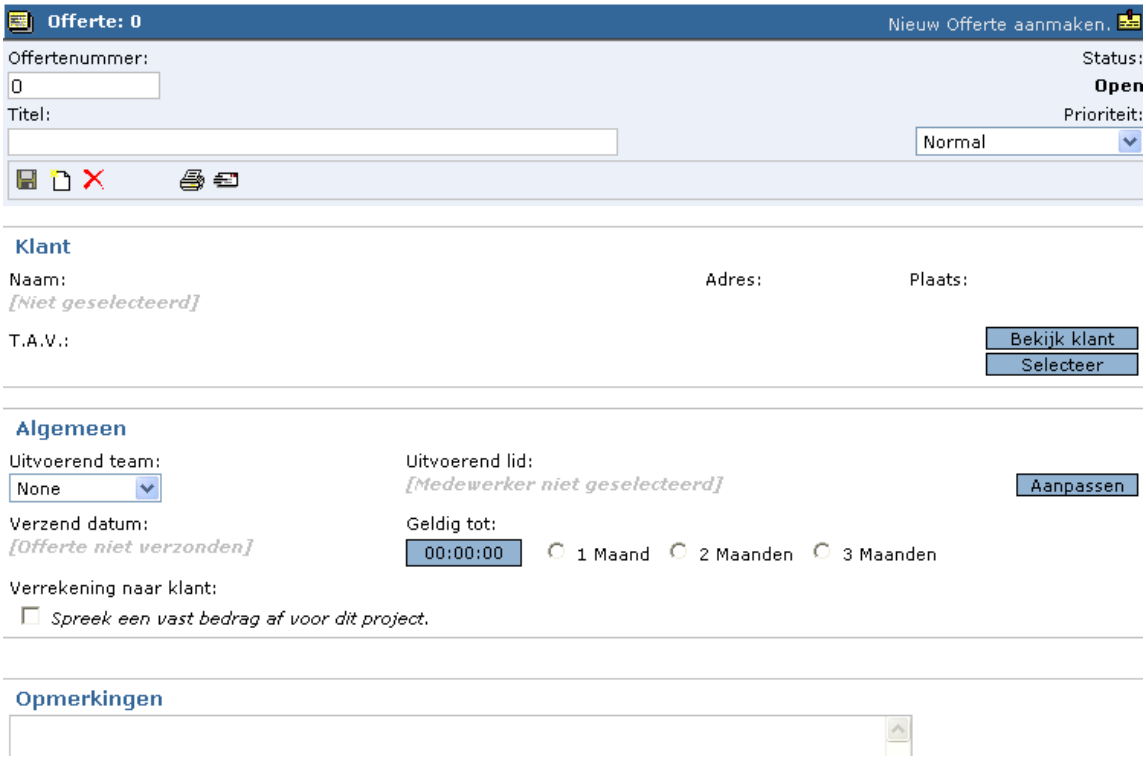
Na het inloggen krijgt u het volgende scherm te zien.



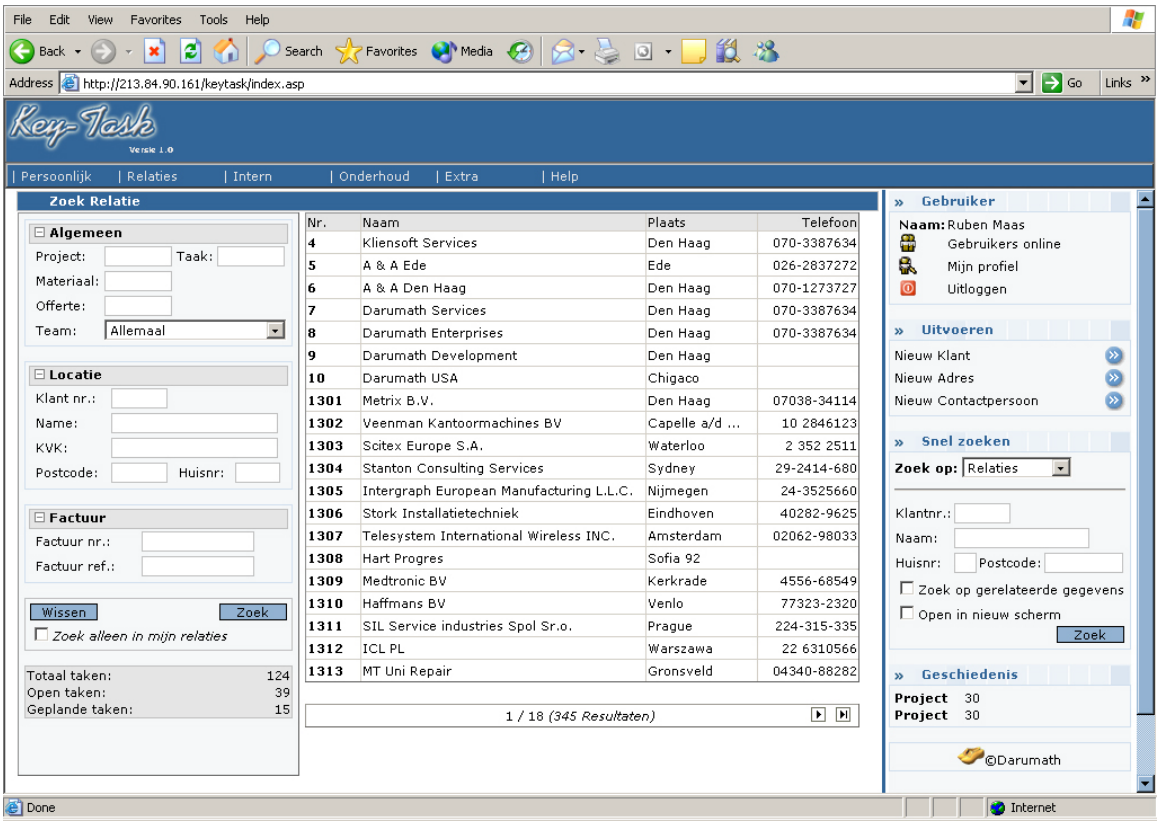
2. Het aanmaken van een offerte

Door in het start scherm aan de rechterkant op “nieuwe offerte” te klikken, kan een nieuwe offerte worden aangemaakt. Uiteraard zijn er meerdere mogelijkheden om een offerte te creëren. Bijvoorbeeld na het selecteren van een klant in het klanten zoekscherm onder het menu “Relaties”. De procedure die volgt is echter in alle gevallen identiek met als enig verschil dat er eventueel al bepaalde data voor u is ingevuld.

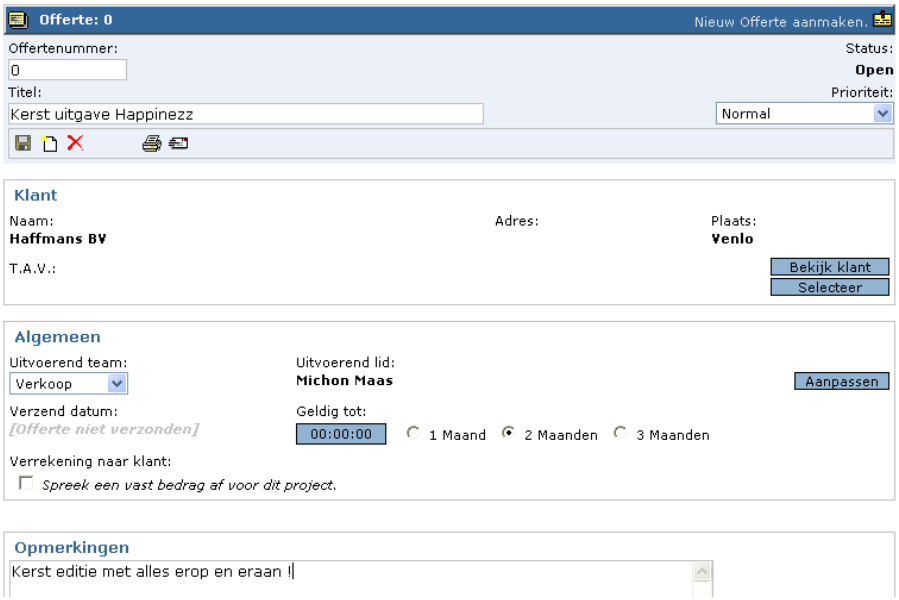
Er kan nu een offerte header worden aangemaakt. In de header wordt een beschrijving van de offerte en informatie omtrent; klant, verantwoordelijke en afspraken geregistreerd.



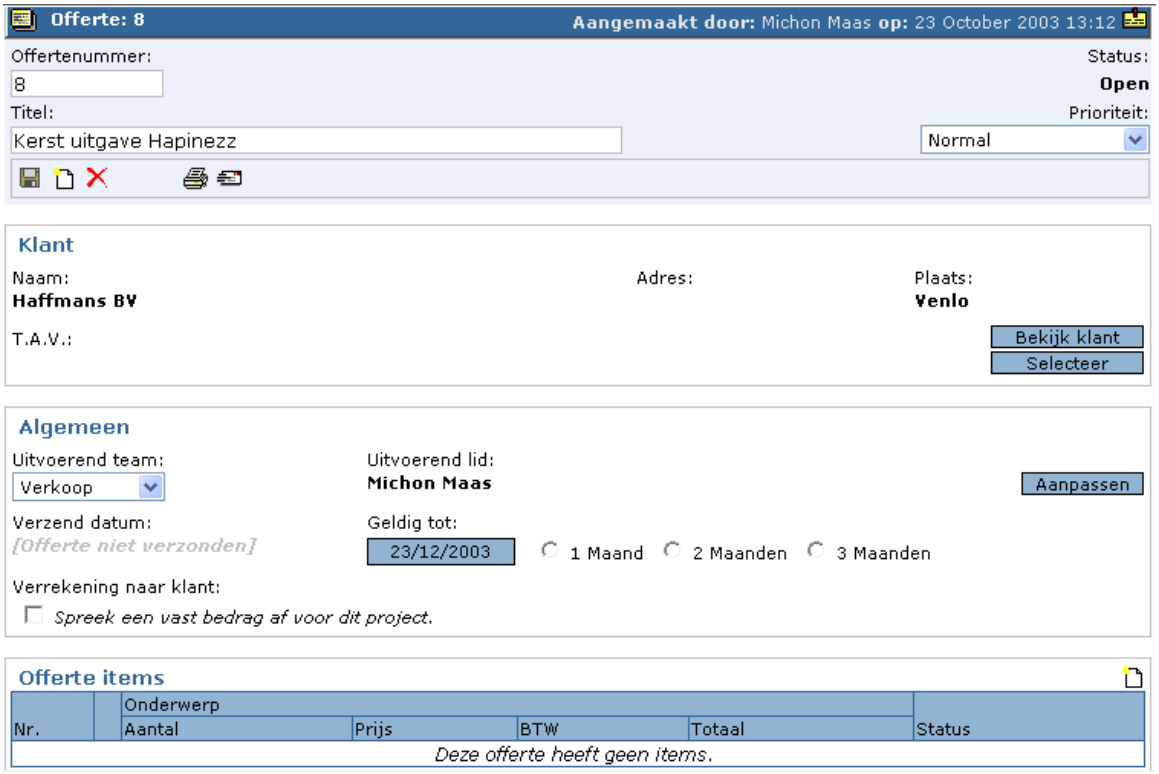
Selecteer de klant waarvoor de offerte dient te worden gemaakt door op de knop “selecteer” in de Klant data groep te klikken. U kunt vervolgens met behulp van het volgende scherm een klant selecteren. Door zoek criteria mee te geven kunt u de resultaten beperken.



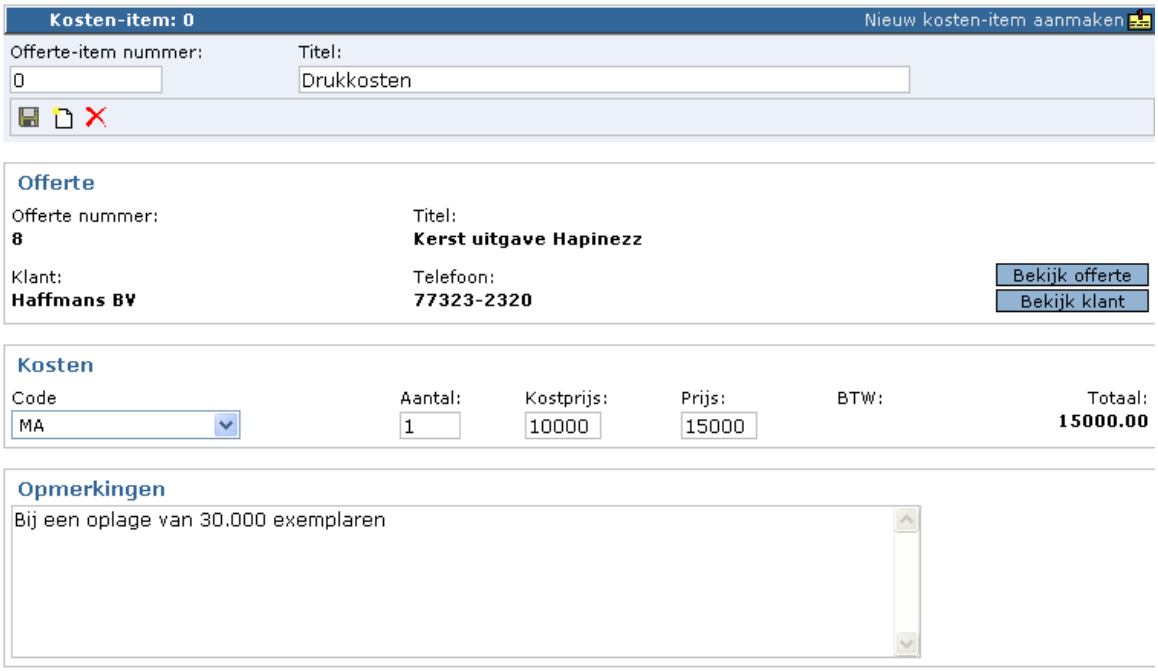
Selecteer de gewenste klant. Vervolgens komt u terug in de offerte header en kunt u de overige noodzakelijke informatie toevoegen. LET OP! Een klant bestaat uit twee niveaus. Op het hoogste niveau wordt de naam van de organisatie weergegeven. Als er op de “+” wordt geklikt worden de adressen (vestigingen) zichtbaar.



Als de header compleet is, kan er op het icoon geklikt worden. De header wordt opgeslagen en er kunnen items aan de offerte worden toegevoegd.



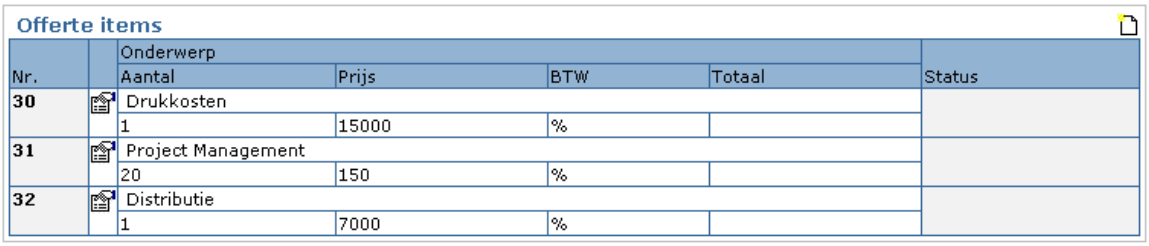
Door in de "Offerte items" data groep op het icoon te klikken kan er een item worden toegevoegd.



Nadat u op het icoon heeft geklikt wordt het item opgeslagen. Kies voor de code MA of TI om aan te geven of het hier om tijd gaat, bijvoorbeeld project management, of materialen of andere kosten. Vergeet geen Specificatie set toe te voegen "" als er specifieke zaken voor dienen te worden bijgehouden. Bijvoorbeeld de specificaties

voor drukwerk. LET OP, u dient het item wel eerst op te slaan voordat u een specificatie toegevoegd (zie ook paragraaf 2.1 voor het gebruik van specificaties).

U kunt nu meerdere items toevoegen door op het icoon te klikken of u kunt terug gaan naar de offerte door op de knop "Offerte bekijken" te klikken.



Offerte items

Nr.		Onderwerp / Aantal	Prijs	BTW	Totaal	Status
30		Drukkosten				
		1	15000	%		
31		Project Management				
		20	150	%		
32		Distributie				
		1	7000	%		

De offerte items zijn dan toegevoegd.

In deze fase kunnen:

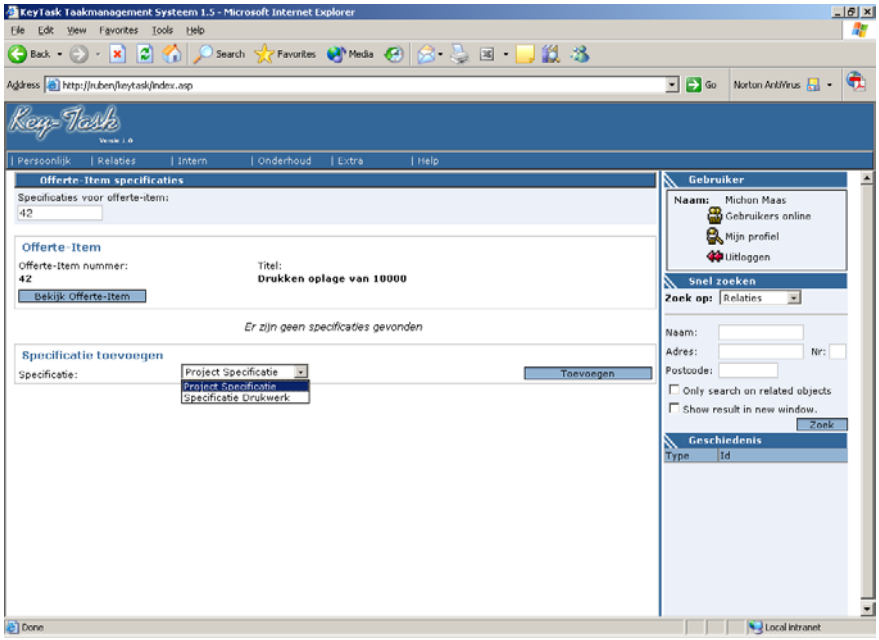
- Specificaties voor de gehele offerte worden vastgelegd (bijv. ten aanzien van druk, papier etc..) door op het icoon te klikken. Deze specificaties kunnen worden gekoppeld aan de offerte of aan een item door eerst een Item te kiezen en een specificatie set te selecteren. Als het gaat om drukwerk zal dit waarschijnlijk op item niveau gebeuren. Lees meer over het gebruik van specificaties in paragraaf 2.1. Deze specificaties worden meegenomen in de generatie van de offerte naar de klant.
- Documenten bijgevoegd en bekeken worden
- Offerte creëren voor leverancier. In het offerte item scherm kan hiervoor op het "" icoon geklikt worden.
- Offerte creëren voor klant. In het offerte scherm kan hiervoor op het "" icoon geklikt worden.
- Een kopie van de offerte gemaakt worden voor een aangepaste versie

Op het moment dat de offerte compleet is en klaar voor verzending naar de klant kan er op het icoon worden geklikt. De offerte die naar de klant dient te gaan wordt nu gecreëerd en de status van de offerte verandert naar verzonden.

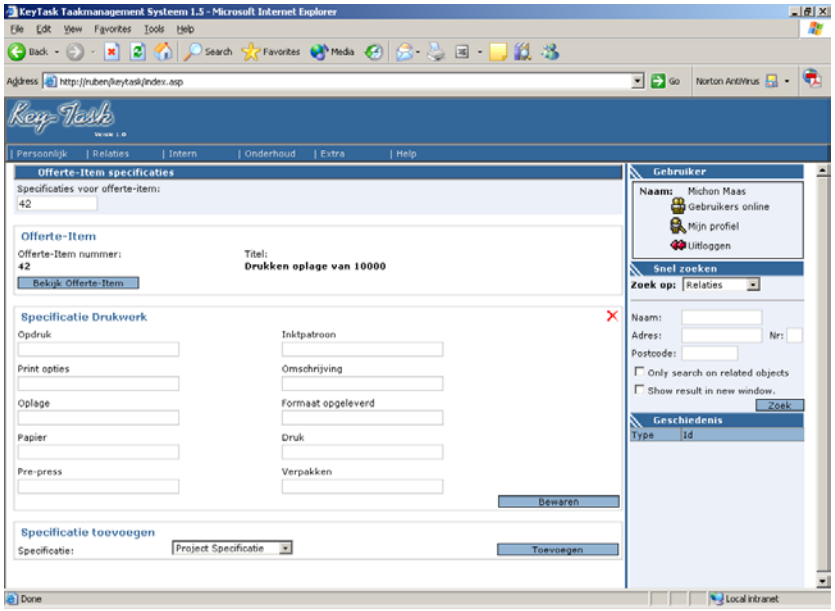
Op het moment dat de klant zijn goedkeuring heeft gegeven aan de offerte kan er op het icoon geklikt worden. Voor de offerte wordt nu een project gecreëerd.

2.1 Het gebruiken van specificaties

Door op het “  ” icoon te klikken kunnen er specificaties worden aangemaakt. Deze specificaties kunnen worden gebruikt bij het uitprinten van verschillende offertes en brieven. In dit voorbeeld wordt een item van een offerte genomen. Het gaat om het drukken van een bepaald maandblad. Om wat meer informatie voor de offerte aanvragen voor de klant te kunnen bijsluiten kan er gekozen worden voor het specificatieset “Specificatie Drukwerk”

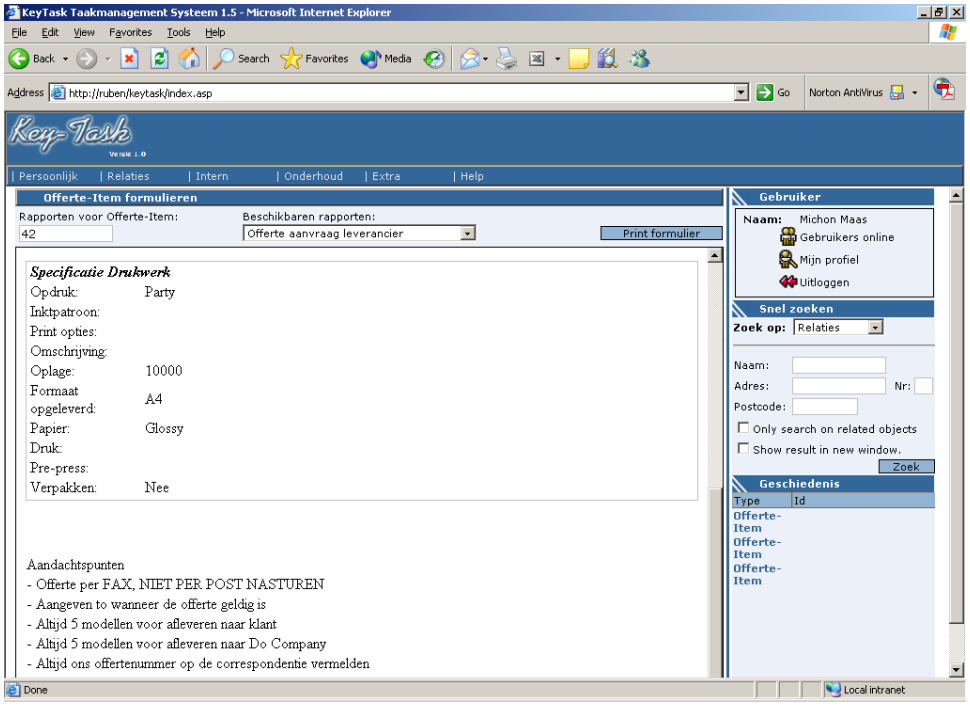


Nadat “Specificatie Drukwerk” is geselecteerd. Kan er op de knop toevoegen worden geklikt.



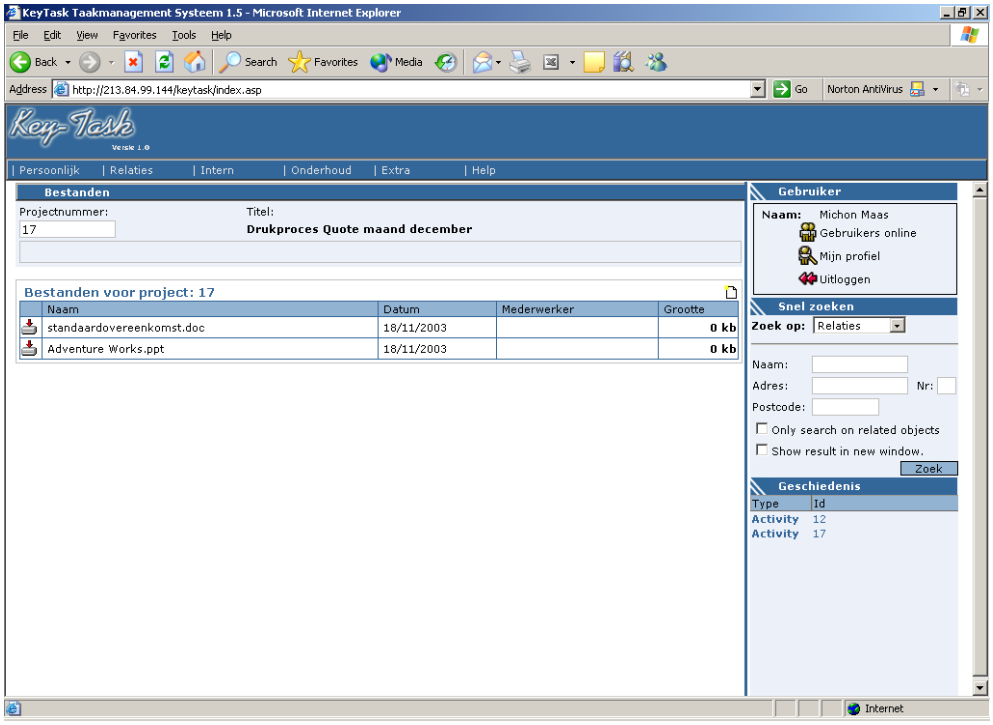
De velden zijn vanaf nu gekoppeld aan het specifieke item. Vulden de informatie in die noodzakelijk is en klik op “bewaren”.

Als u nu terug gaat naar het offerte item en een offerte aanvraag bij een leverancier print door op het “🖶” icoon te klikken zult u zien dat de specificaties voor het drukwerk worden meegezonden in de specificatie naar de brief.

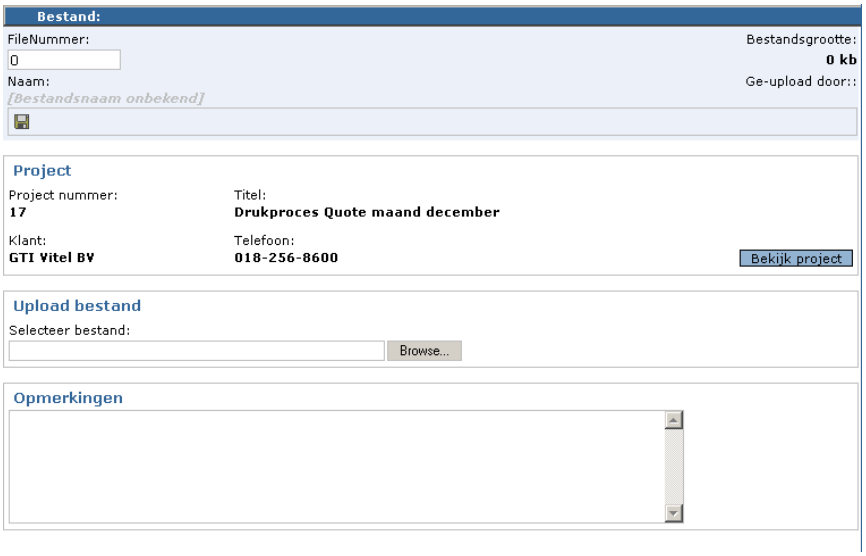


2.2 Bestanden bijvoegen en bekijken

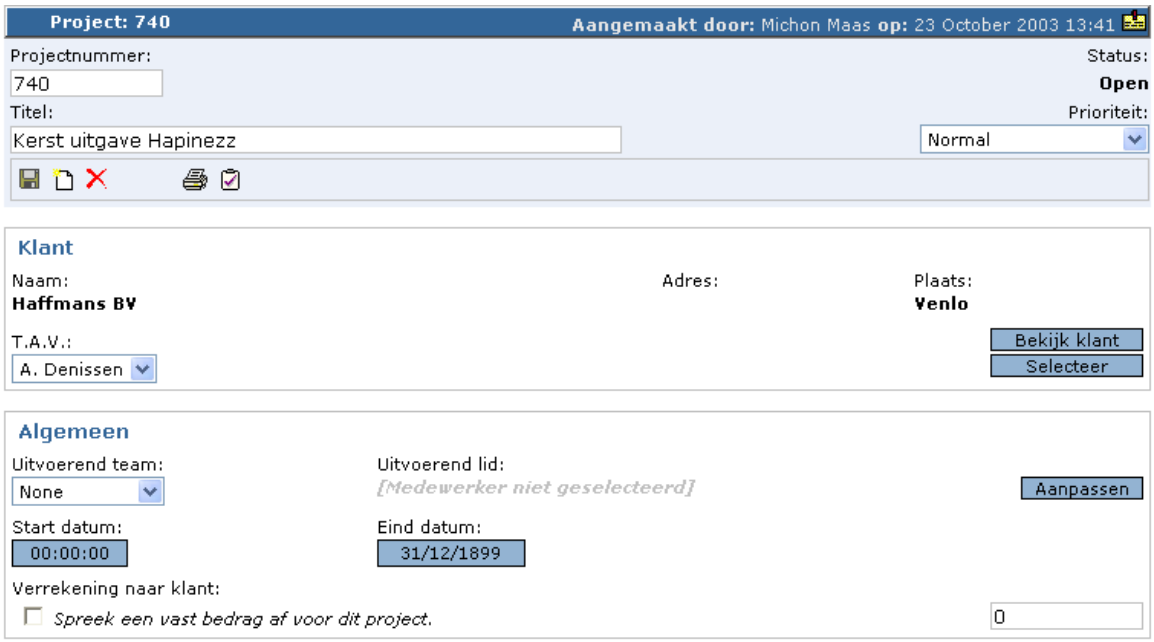
Als een offerte of project geselecteerd is kan er door op "" te klikken gekeken worden naar bijgesloten documenten. Door op het te klikken kan een nieuw document worden bijgevoegd.



Door op "browse" te klikken kan een bestand op de harde schijf worden geselecteerd. Er kan eventueel een beschrijving van het bestand worden toegevoegd in het "Opmerkingen" veld. Door op het "" icoon te klikken wordt het bestand naar de server gekopieerd en opgeslagen. LET OP, het kopieren naar de server kan afhankelijk van de omvang enige tijd in beslag nemen.



3. Het Project



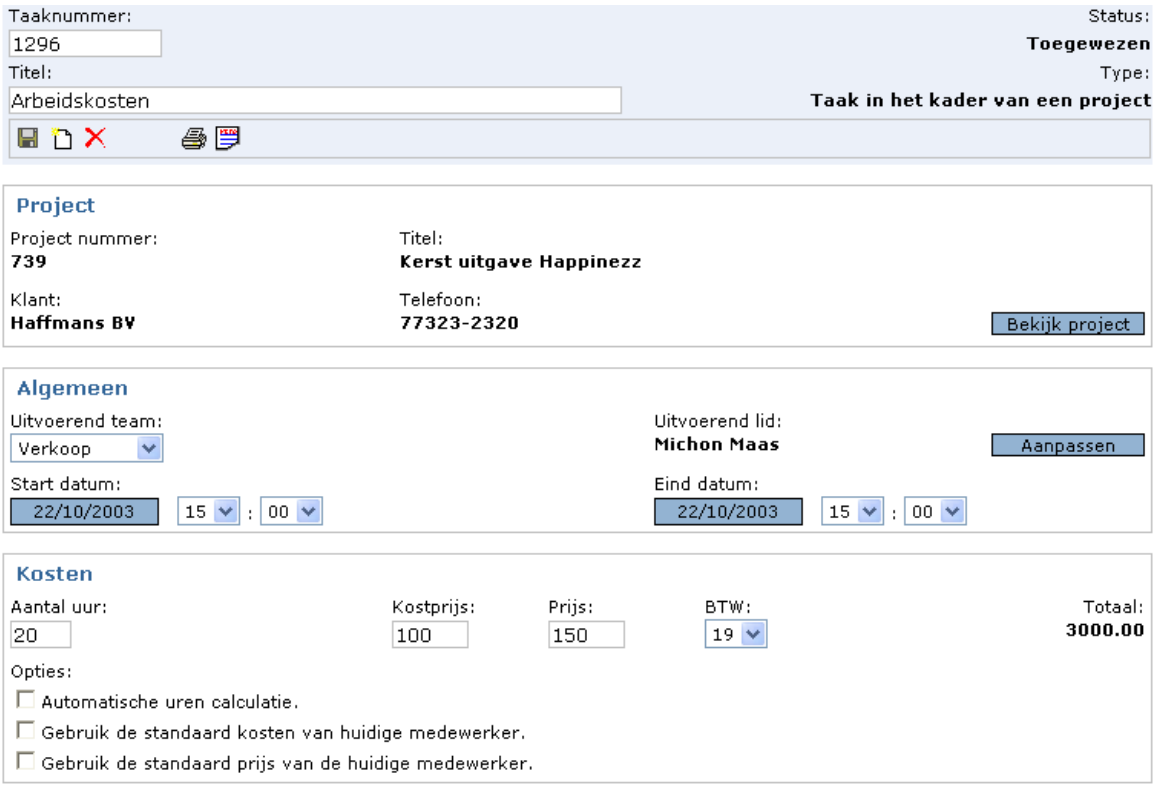
In deze fase kunnen:

- Document worden toegevoegd en bekeken (zie ook paragraaf 2.2)
- Specificatiesets kunnen worden aangevuld of gecreëerd (zowel voor het project als voor materiaal en taken). Zie hiervoor ook paragraaf 2.1.
- Orderbevestigingen voor leveranciers worden gecreëerd
- Een order voor de klant worden gecreëerd
- Project items naar een factuur worden verzonden
- Project afronden en factureren
- Een kopie gemaakt worden gemaakt van het project om een vergelijkbaar project op te starten
- Extra items worden toegevoegd door in de taken of materialen data groep op te klikken

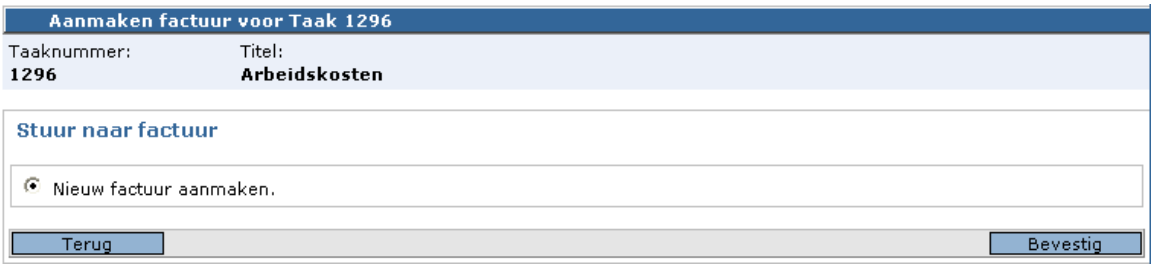
Om de project gegevens up-to-date te houden zullen statussen dienen te worden aangepast als er materialen binnen zijn gekomen of taken zijn afgerond. Door op het icoon voor een item te klikken kunnen details van taken en materialen worden bekeken en worden aangepast.

3.1 Het factureren van afzonderlijke items

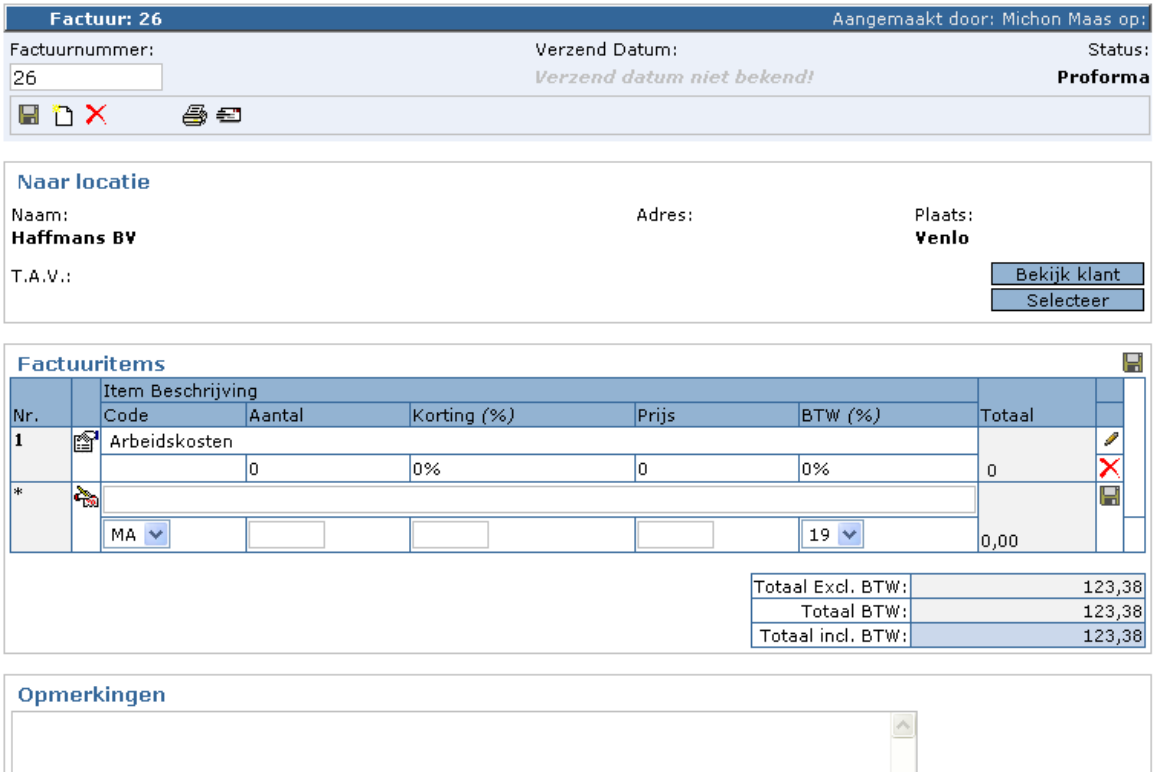
Door naar de details van een project item te gaan via het icoon kan een item naar een factuur worden verzonden doormiddel van het icoon



U krijgt vervolgens de volgende vraag.



Als u op de knop “Bevestig” klikt wordt er een nieuwe pro forma factuur aangemaakt.



Achter het icoon kan eventueel nog handmatig een item aan de factuur worden toegevoegd. Door op te klikken wordt het item opgeslagen.

Een item op een proforma factuur kan gewijzigd worden door op het icoon achter het item te klikken. Er wordt dan een pop-up scherm geopend waarop u het item kan wijzigen. Door op "Opslaan" te klikken wordt de wijziging doorgevoerd.

Door op te klikken verwijderd u een item van de proforma factuur.

Als een item uit een project naar een factuur is verstuurd komt wordt het ✓ weergegeven voor het item in het project scherm.

Taken

Nr.			Onderwerp	Start	Duur	Uitvoerder	Totaal	Status
1296	✓		Arbeidskosten	22/10/2003 15:46:45	0 minutes	Michon Maas		Toegewezen

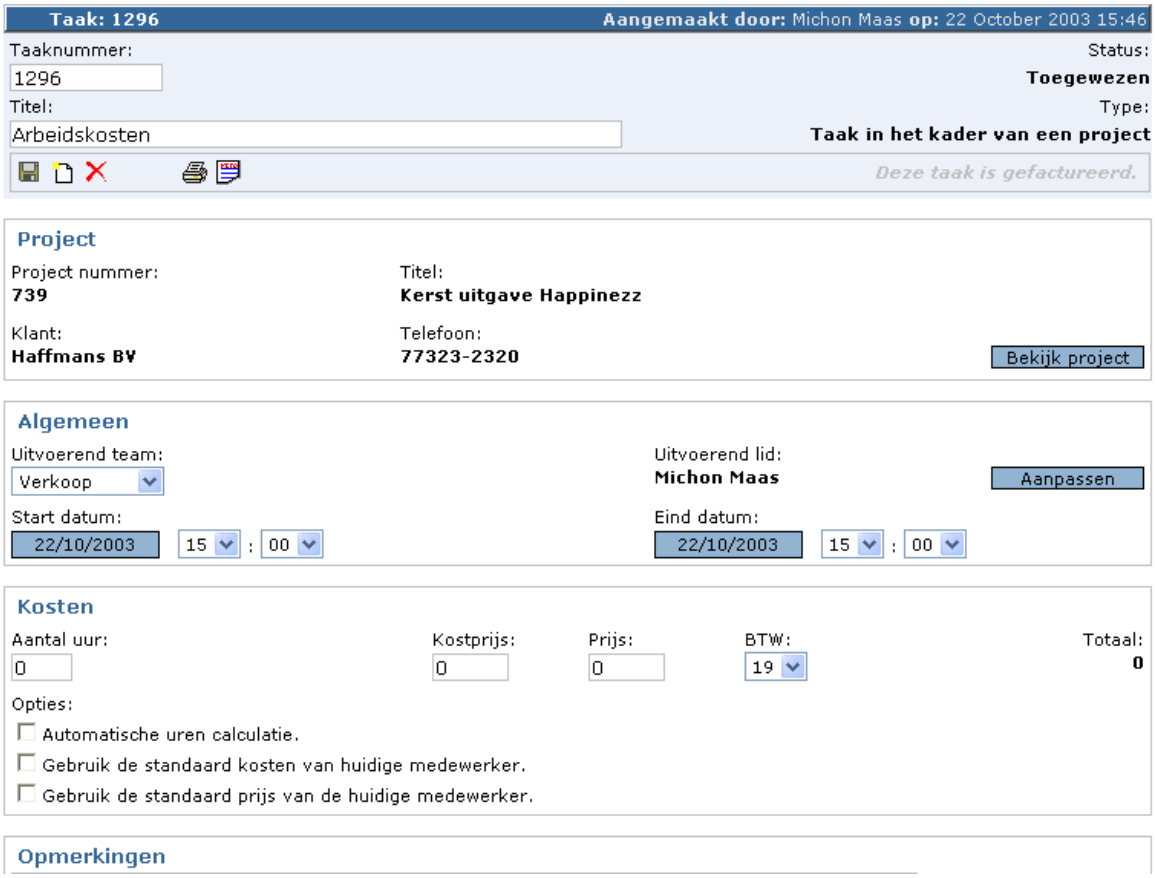
Materialen

Nr.			Onderwerp	Aantal	Prijs	BTW	Totaal	Status
1290			Drukkosten	1	130000	0	130000	In bestelling
1291			Distributie	1	50000	0	50000	In bestelling

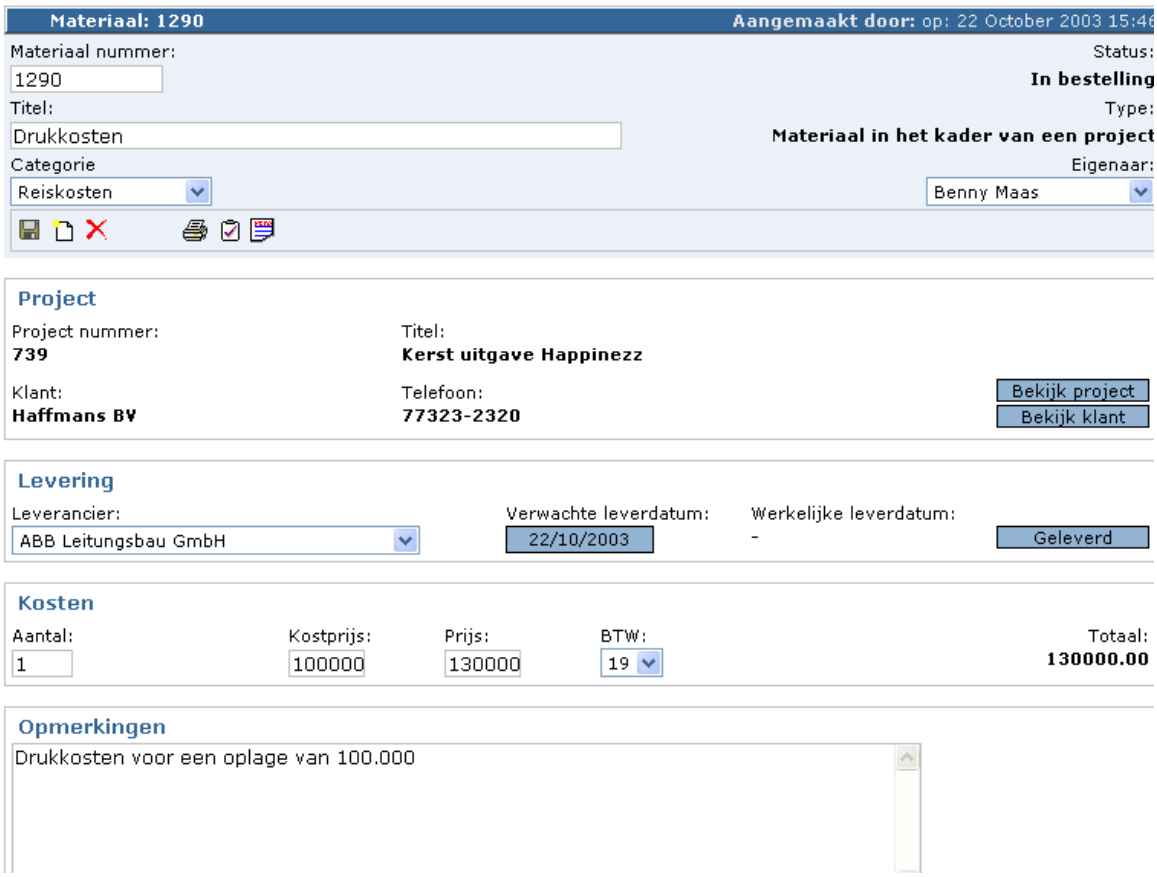
3.2 Het wijzigen/toevoegen van taken en materialen

Door op het icoon in de data groep van Taken of Materialen te klikken kan respectievelijk een taak of materiaal worden toegevoegd. Met behulp van het icoon kan naar detail informatie van het item worden gekeken en eventueel veranderingen worden aangebracht.

De Taak



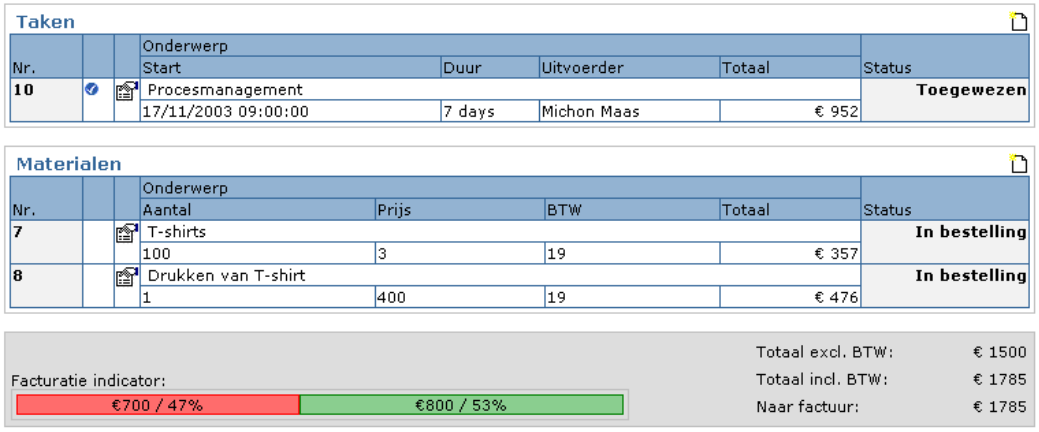
Een taak is in principe tijd die een zeker persoon aan een project besteedt. Er kan worden gekozen om aan de hand van de duur van de taak op basis van het werkrooster automatisch uren te laten calculeren. Er kan ook worden gekozen om de standaard kosten en/of prijs die aan de medewerker verbonden zijn mee te nemen in de calculatie.



Details van materialen kunnen worden aangepast en worden opgeslagen door op te klikken. Ook de status kan worden aangepast door bij levering van materialen of diensten op de knop "Geleverd te klikken.
De status wordt dan omgezet naar geleverd en de werkelijke leverdatum wordt automatisch ingevuld. Overige specificaties kunnen worden bijgehouden door op te klikken. Voor meer informatie over het gebruik van specificaties, zie paragraaf 2.1.

3.2 Facturatie indicator

Gedurende looptijd van een project wordt de actuele facturatie status weergegeven.



Taken

Nr.			Onderwerp / Start	Duur	Uitvoerder	Totaal	Status
10			Procesmanagement 17/11/2003 09:00:00	7 days	Michon Maas	€ 952	**Toegewezen**

Materialen

Nr.			Onderwerp / Aantal	Prijs	BTW	Totaal	Status
7			T-shirts 100	3	19	€ 357	**In bestelling**
8			Drukken van T-shirt 1	400	19	€ 476	**In bestelling**

Facturatie indicator:

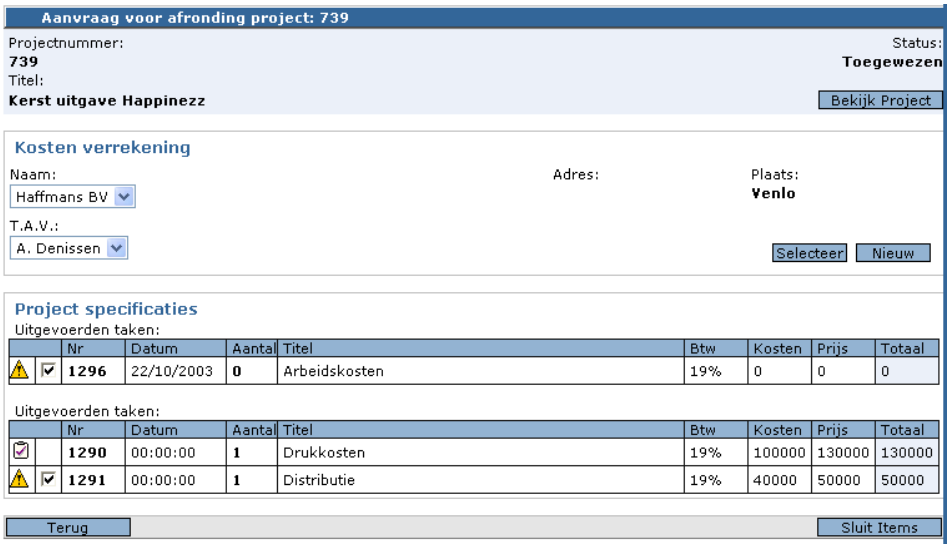
€700 / 47% | €800 / 53%

Totaal excl. BTW: € 1500
Totaal incl. BTW: € 1785
Naar factuur: € 1785

- De kleur rood geeft het percentage en bedrag aan dat nog niet op een pro-forma factuur is gezet.
- De kleur oranje geeft het percentage en bedrag aan dat op een pro-forma factuur staat, maar nog niet gefactureerd is.
- De kleur groen geeft het percentage en bedrag aan dat is gefactureerd.

3.4 Het afsluiten van een project

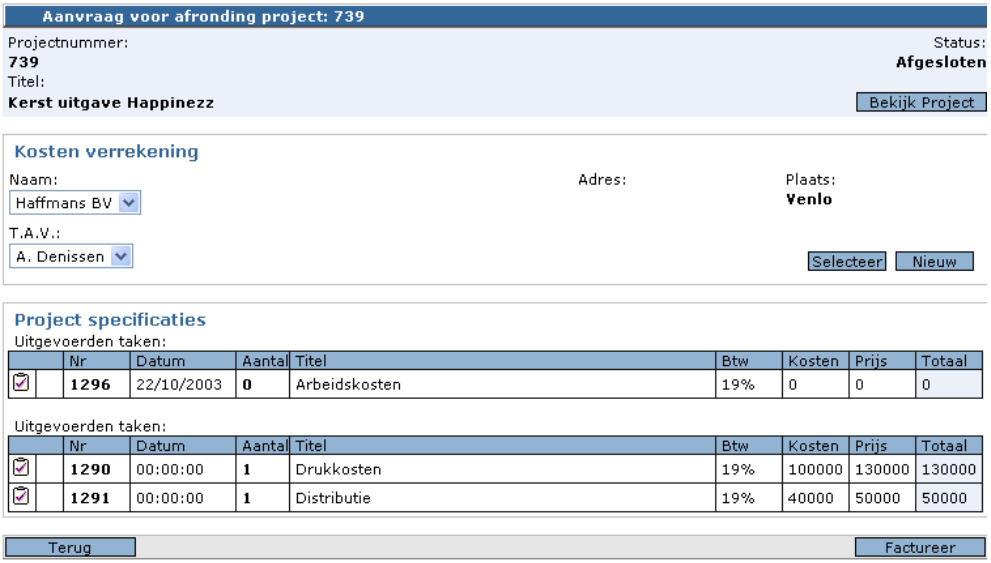
Als een project geselecteerd is kan deze worden afgesloten door op het icoon te klikken. Het onderstaande scherm wordt geopend.



Op het moment dat er een verzoek wordt gedaan om een project te sluiten wordt er een check gedaan of alle afzonderlijke projectonderdelen ook gesloten of geleverd zijn. Items die nog open staan worden gemerkt met het icoon. Gesloten en geleverde items zijn gemerkt met .

Door open items aan te vinken kunnen deze gesloten worden door op de knop "Sluit Items" te klikken.

Als alle items gesloten en geleverd zijn ziet het scherm er als volgt uit.



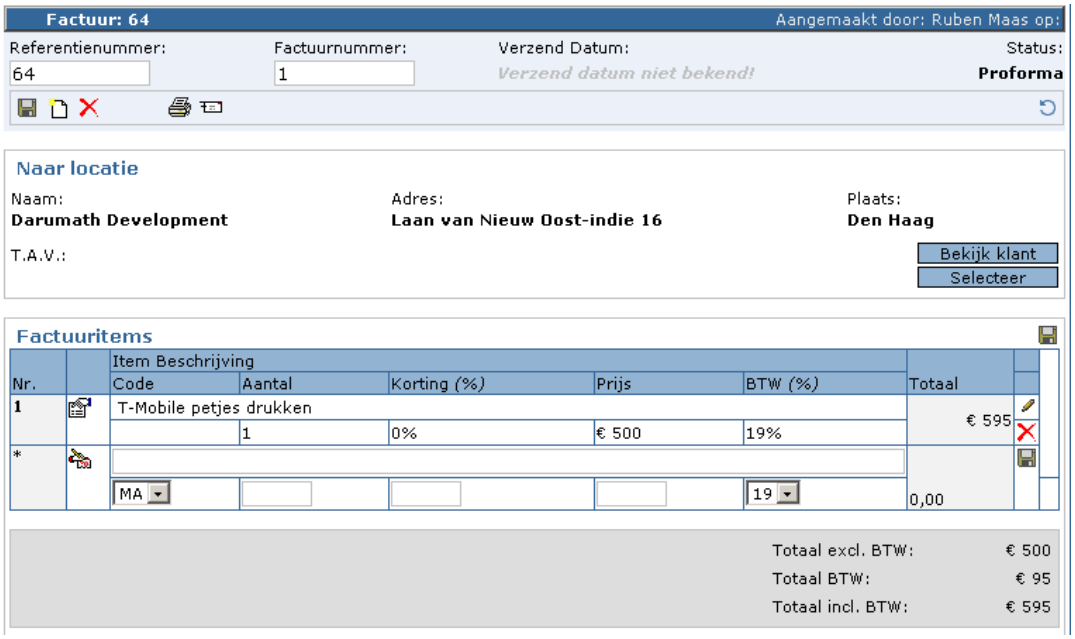
Het project kan nu naar een Pro-Forma factuur gestuurd worden.

Als er op de knop “Factureer” wordt geklikt, controleert het systeem of er al pro-forma facturen voor de klant klaar staan. Er kan vervolgens worden gekozen of er een nieuwe pro-forma factuur gemaakt moet worden of dat alle items aan een bestaande dienen te worden toegevoegd.

4. Facturatie

4.1 Algemeen

Onder het menu "intern" kunt u het item "Pro-Forma facturen" vinden. Door hierop te klikken kan er een overzicht verkregen worden van de Pro-Forma facturen die klaar staan. Door een Pro-Forma factuur te selecteren kunnen de details worden bekeken en eventueel worden aangepast , toegevoegd of verwijderd.



Als de pro-forma factuur correct en compleet is kan deze worden verzonden door op het icoon te klikken. De Pro-Forma factuur wordt nu een formele factuur met een opeenvolgend factuur nummer en kan worden geprint.

4.2 Het factureren van een all-in project

In principe werkt het factureren van een all-in project hetzelfde als het factureren van een normaal project. Het is echter niet mogelijk om aparte items te factureren.