Eindverslag
Project Verloskundeapplicatie

Datum	14 januari 2004
Student / Auteur	C. Kruijf, 99008079

Eindverslag
Project Verloskundeapplicatie

Datum	14 januari 2004
Student / Auteur	C. Kruijf, 99008079
Afstudeerperiode	30 augustus 2004 – 14 januari 2004
Opleiding	Haagse Hogeschool Informatica en Informatiekunde
Studiepad	Ontwikkeling van Software en Technische Infrastructuren (OSTI)
Organisatie	Verloskundeprakijk de Maatschap Lammenschans Bloemistenlaan 45/A 2313BB Leiden Tel: 071 5124040
Bedrijfsmentor	Mevr. van Iterson
Examinatoren	Dhr. N.J.J. Groot Dhr. F. Bögels

Referaat

Kruijf, C., Ontwerpen en ontwikkelen applicatie ten behoeve van het verloskundeproces. Leiden, verloskundepraktijk de Maatschap, 2005.

Dit document beschrijft en evalueert de werkzaamheden en opgeleverde producten die de auteur heeft uitgevoerd en opgeleverd tijdens een project dat in opdracht van de verloskundepraktijk de Maatschap is uitgevoerd.

Het project omvat het ontwerpen en ontwikkelen van een vernieuwde applicatie die het proces van verloskundigen kan ondersteunen. Het ontwerpen van een Grafische User Interface, een database, een interface tussen de GUI en de database en de ontwikkeling van de functionaliteiten. Het project is uitgevoerd in de periode van 30 augustus 2004 tot en met 14 januari 2005.

Descriptoren

- IAD
- UML
- Verloskunde
- Database
- GUI

Voorwoord

Voor u ligt het eindverslag dat ik heb geschreven naar aanleiding van mijn afstudeeropdracht die heb uitgevoerd ter afsluiting van de opleiding Informatica en Informatiekunde aan de Haagse Hogeschool. De opdracht is uitgevoerd bij de verloskundepraktijk de Maatschap te Leiden.

Bij deze wil ik graag mijn externe bedrijfsmentor dhr. A. Nederend bedanken voor de overlegmomenten tijdens het afstuderen. Verder bedank ik de opdrachtgever Mevr. S. van Iterson en de collega's voor de medische kennis en prettige samenwerking. Als laatste wil ik mijn examinatoren van de Haagse Hogeschool dhr. N.J.J. Groot en dhr. F. Bögels bedanken voor het volgen en sturen van het afstuderen.

Leiden, 14 januari 2005.
Cees Kruijf
Verloskundepraktijk de Maatschap.

Inhoudsopgave

1. Inleiding

De opleiding Informatica en Informatiekunde aan de Haagse Hogeschool wordt afgesloten door het zelfstandig uitvoeren van een afstudeeropdracht bij een organisatie. Dit verslag beschrijft het verloop van mijn afstudeeropdracht, die ik uitgevoerd heb bij de verloskundepraktijk de Maatschap lammenschans te Leiden.
Hierbij ga ik in op de procesgang en de totstandkoming van de opgeleverde producten en de gemaakte beslissingen. Vervolgens evalueer ik de gekozen aanpak, de uitgevoerde activiteiten en de opgeleverde producten.

Het doel van dit verslag is de examinatoren en de gecommitteerde die het afstuderen beoordelen, inzicht te geven in de omvang, diepgang en het niveau van de opdracht. Daarnaast dienen zij door middel van dit verslag inzicht te krijgen in de kwaliteit van de opgeleverde producten en de kwaliteit van het proces van uitvoering. Op basis van dit inzicht dienen zij voldoende in staat te zijn het afstuderen te beoordelen.

De opbouw van dit verslag is als volgt. In hoofdstuk 2 worden de omstandigheden van het project beschreven. Het daaropvolgende hoofdstuk tot en met hoofdstuk 7 beschrijven de uitvoering van het project, waarbij elke fase van het project een hoofdstuk vormt. Tot slot worden in hoofdstuk 8 de opgeleverde producten en uitgevoerde activiteiten geëvalueerd.

2. Projectschets

In dit hoofdstuk worden de omstandigheden van het project dat in het kader van het afstuderen is uitgevoerd beschreven. Als eerste wordt in paragraaf 2.1 De organisatie in kaart gebracht. In paragraaf 2.2 wordt het verloskundeproces toegelicht en in paragraaf 2.3 de uiteindelijke opdracht. Paragraaf 2.4 beschrijft de activiteiten die in het kader van het plan van aanpak zijn uitgevoerd.

2.1. Organisatie

De opdracht is uitgevoerd bij de verloskundepraktijk de Maatschap. In deze paragraaf wordt het ontstaan van de organisatie beschreven en wordt de plaats binnen de organisatie toegelicht.

2.1.1. Organisatorische inrichting

De verloskundepraktijk de Maatschap is sinds 1979 werkzaam in de wereld van verloskundigen. In die tijd bestond de praktijk slechts uit één verloskundige. Bovendien droeg de praktijk toen geen naam, en heette het de verloskundepraktijk. Een aantal jaren later, in 1994, kreeg de praktijk een tweede verloskundige en werd de naam van de praktijk 'Praktijk Noordeinde'. Deze naam was te danken aan de locatie van de praktijk. Sinds 1999 is de praktijk werkzaam op de huidige locatie in Leiden.

Op dit moment zijn er vijf verloskundigen werkzaam, hoewel dit aantal varieert. De kernactiviteiten van de praktijk zijn het controleren en sturen van de bevallingen. De praktijk houdt ook spreekuren waar cliënten met minder dringende vragen terechtkunnen op afspraak.

2.1.2. Plaats binnen de organisatie

De praktijk heeft op dit moment vijf werknemers die ook allemaal de eigenaar zijn. Aangezien er binnen de organisatie geen structuur is wat betreft verschillende afdelingen, was mijn plaats binnen de organisatie als ondersteunding van de automatisering. De werkplek die daarvoor gecreëerd was, was verre van ideaal. Een kamertje naast de kamer van het spreekuur.
Deze werkplek had geen mogelijkheid tot het internet. Om die reden is een extra werkplek gecreëerd op een locatie buiten de organisatie.
Om de werkdruk en –sfeer binnen de organisatie te kunnen vertalen naar de applicatie werd zoveel mogelijk gestreefd om te werken binnen de organisatie.

2.2. Het verloskundeproces

Om een beter inzicht te krijgen in de activiteiten die een verloskundige verricht en onder welke omstandigheden zij moeten werken is dit in deze paragraaf toegelicht.

2.2.1. Globaal verloop

Wanneer een vrouw onder normale omstandigheden meer dan twee weken afwijkt van de menstruatieperiode, bestaat de mogelijkheid tot overwegen van een zwangerschapstest. Als deze test uitwijst dan de vrouw inderdaad zwanger is, wordt een afspraak gemaakt met een verloskundige. De vrouw kan vervolgens kiezen of ze thuis of in een ziekenhuis wil bevallen.

In de laatste jaren is een tendens te zien waarin meerdere vrouwen ervoor kiezen om thuis te bevallen, wat de werkgelegenheid voor verloskundepraktijken bevorderd. Aan deze werkgelegenheid zitten echter enkele eisen. De praktijken die zich richten op vrouwen die thuis willen bevallen, hebben afgesproken dat ze alleen cliënten accepteren die in een bepaalde regio zitten. Zo mag de verloskundepraktijk de Maatschap te Leiden, geen cliënten behandelen die woonachtend zijn in Den Haag.
Daarnaast geldt ook dat de praktijk maar een maximaal aantal bevalling mag doen. Deze restrictie is door de overheid opgelegd. Dit komt ten goede van de cliënten, want deze restrictie is onder andere vastgesteld om een minimaal aantal uren verzorging te verlenen aan een cliënt.

Indien een vrouw besluit om thuis te bevallen en woonachtend is in de regio van de praktijk, zal deze telefonisch contact op moeten nemen met de praktijk. In dit gesprek wordt voor de cliënt een afspraak gemaakt in het ziekenhuis voor het maken van echo en afnemen van bloed voor de controle op bepaalde (erfelijke) ziekten/afwijkingen. Het labuitslag van het bloed bevat het volgende:

- Bloedgroep
- Rhesusfactor
- Irreguliere antistoffen
- Hepatitis-B
- Lues (SOA)
- Anti-HIV

Naast de bloeduitslagen, wordt de vrouw gewezen op het zo snel mogelijk behandelen van de ziekte ‘rode hond’ (onvruchtbaarheid) en het vermijden van ‘katten’. Vooral de uitscheiding van katten kan zorgen voor toxoplasmose, wat kan leiden tot misvorming tijdens de groei van de baby.

Als uit de gemaakte echo blijkt dat de vrouw inderdaad zwanger is, wordt een afspraak gemaakt in de 12^{e} week vanaf de laatste menstruatie. De datum van de laatste menstruatie is niet altijd zeker en daarom kan de datum van uiteindelijke bevalling altijd een tweetal weken verschillen.

Tijdens de eerste afspraak in de 12^{e} week worden de cliëntgegevens genoteerd en wordt het verloop van de verdere zwangerschap toegelicht. Aangezien sommige vrouwen geen kind willen hebben met het syndroom van down, zijn er tegenwoordig tests waarmee de kans hierop bepaald kan worden. De twee mogelijke tests ‘nekplooi’ en TST (Triple Serum Test) kunnen uitwijzen of het kind een niet normale ontwikkeling zal doorstaan. Aangezien de testen gebaseerd zijn op kansberekening, bestaat de mogelijkheid dat de

vastgestelde diagnose niet correct is. Om die reden is een derde test in ontwikkeling die de nauwkeurigheid van de diagnose vergroot. Deze laatste test is echter nog niet goedgekeurd door de overheid.

Als er niet wordt overgegaan tot een abortus en de anamnese (medische voorgeschiedenis) van de cliënt binnen de behandelmogelijkheden van de verloskundige valt, kan over worden gegaan tot het maken van de benodigde afspraken. Een uitzondering in de anamnese kan bijvoorbeeld een erfelijke ziekte zijn, of een eerdere bevalling waaruit een tweeling is voorgekomen. Een bevalling met een meerling mag namelijk niet door een verloskundige worden uitgevoerd, hiervoor wordt een gynaecoloog ingeschakeld.

In de 12^{e} tot de 24^{e} week wordt om de vier weken een afspraak gemaakt met de cliënt, waarin de bloeddruk en hoogte van de baarmoeder worden gecontroleerd. Daarnaast wordt er gecontroleerd of het kindje nog in leven is, door te luisteren naar het hartje. Vanaf de 33^{e} week is dit om de drie weken en vanaf de 36^{e} week om de twee weken. Indien zich in die weken tot de bevalling complicaties voordoen, wordt alsnog de gynaecoloog ingeschakeld, waardoor de vrouw niet meer thuis kan en mag bevallen.

Als alles goed verloopt zal de vrouw na ongeveer negen maanden na de laatste menstruatie, een baby op de wereld zetten.

2.2.2. Medische verantwoording

Verloskundigen moeten zich verantwoorden aan de medische verplichtingen. De wet die medici verplicht tot geheimhouding en het verplicht helpen van een mens in nood is de BIG-wet. In deze wet is tevens vastgelegd dat de gegevens van patiënten / cliënten minimaal tien jaar in een archief bewaard moeten blijven. Indien deze gegevens binnen de tien jaar onvindbaar zijn, verzuimt de organisatie zijn plicht.
De aard van het ‘misdrijf’ bepaalt of deze bestraft wordt met een boete of eventueel een verbod om voor een bepaalde tijd medische handelingen te verrichten, tenzij het levens kan redden.

Naast de verplichting om cliëntgegevens tien jaar lang gearchiveerd te houden, is er ook de medische geheimhoudingsplicht. Deze plicht verbied medici de gegevens van een patiënt of cliënt te verspreiden. Het is daarom noodzakelijk om verstandig om te gaan met de gegevens die in deze applicatie worden gebruikt. Deze omgang wordt vastgelegd door middel van een privacyverklaring.

2.3. Opdracht

Om meer inzicht te krijgen in de afstudeeropdracht worden onderdelen van het project hier toegelicht. De probleemstelling en opdrachtomschrijving vormen hiervoor de basis.

2.3.1. Probleemstelling

De verloskundepraktijk de Maatschap beschikt over een aantal bekwame verloskundigen met een beperkt inzicht in de informatiestromen en bijbehorende automatiseringsmogelijkheden. Om de uit te voeren werkzaamheden te kunnen automatiseren is een aantal jaren geleden besloten tot de aanschaf van een applicatie ter ondersteuning van het verloskundeproces.

Het verloskundepakket ‘Orfeus’ is gebruiksonvriendelijk. De applicatie heeft een onvriendelijke interface en een atechnisch karakter. De foutmeldingen waarmee de gebruiker geconfronteerd wordt zijn nauwelijks informatief te noemen. In sommige gevallen zijn deze zelfs beledigend. Naast de ondeskundige feedback laat de gebruikershandleiding ook te wensen over. Een aantal a4-tjes met screenshots en sporadisch een tekstuele toelichting.

De licentiekosten van € 2000,- per jaar voor deze applicatie zijn van een dusdanig hoog niveau dat voor deze terugkerende kostenpost een professioneler karakter mag worden verwacht.

De installatie van de applicatie is lastig voor een verloskundige die weinig tot geen kennis heeft van computers. De geleverde Compact Discs moeten in een door de leverancier vastgelegde volgorde worden geïnstalleerd. Indien dit begrepen is, wordt van de gebruiker verwacht dat er geen nieuwere versies van de bijgeleverde software Microsoft Access zijn geïnstalleerd. Indien dit laatste wel het geval is, wordt de oude versie geïnstalleerd met alle bijbehorende fouten en ‘exploits’.

2.3.2. Opdrachtomschrijving

Aan de hand van de omvang van de probleemstelling was besloten om een geheel nieuwe applicatie te ontwerpen. De afstudeeropdracht omvat daarom het ontwerpen en realiseren van een nieuwe applicatie ter ondersteuning van het primaire verloskundeproces. Het is vooral van belang dat ieder aspect van het begin tot het eindpunt van het client-verloskunde traject geheel of gedeeltelijk wordt ondersteund, tenzij anders overeengekomen met de opdrachtgever.

De applicatie zal vooral bestaan uit een overzichtelijke grafische user interface, zelfverklarende foutmeldingen en een logische en begrijpelijke gebruikershandleiding. Met begrijpelijk wordt er bedoeld dat de uitleg in de gebruikershandleiding niet te technisch mag zijn, omdat de gebruikers van de applicatie niet technisch zijn aangelegd op gebied van ICT.

2.3.3. Doelstelling

Voor de opdracht is er een doelstelling vastgesteld. *Het ontwerpen en realiseren van de verloskunde applicatie.* Deze doelstelling is tweeledig:

- Het ontwerpen van de applicatie
- Het realiseren van de applicatie

Hoewel de bovenste doelstellingen veel op elkaar lijken heb ik geleerd dat het ontwerpen en het realiseren van een software compleet verschillende handelingen zijn.

2.3.4. Methoden en technieken

Het eindproduct van deze afstudeeropdracht is een applicatie waar de verloskundigen in de toekomst mee moeten gaan werken. Voordat de applicatie überhaupt gebruikt gaat worden is het bevorderlijk om een nieuw product mee te laten ‘groeien’ met een organisatie. De resultaten moeten snel teruggekoppeld worden en de inbreng van de gebruikers moet van een hoog niveau zijn. Op deze manier is het meer de applicatie van de organisatie dan van de afstudeerder.

Met deze criteria in overweging genomen waren de alternatieven in mijn opzicht gering. De aangeleerde methode SDM voldeed in deze situatie totaal niet. Vandaar dat is besloten om een nieuwe methode toe te passen, genaamd IAD. (Interactive Application Development). Deze methode biedt een aantal grote voordelen die vooral toepasbaar zijn op deze afstudeeropdracht. Onder andere de mogelijkheid om de gebruiker te betrekken bij het ontwikkelen.

De gebruikte technieken kunnen worden gecategoriseerd in drie delen:

- Techniek voor het ontwerpen van de modellen
- Techniek voor het realiseren van de modellen voor de totstandkoming van een applicatie
- Overige technieken die het te doorlopen traject aanvullen.

De gebruikte techniek die in combinatie met de IAD-methode zal worden gehanteerd is UML (Unified Modelling Language). Hoewel de ervaring met de UML-technieken - net als de ontwikkelmethode - beperkt is, sluit deze techniek toch het beste aan op een object georiënteerde programmeertaal. Een objectgeoriënteerde aanpak is bij het realiseren van een nieuwe applicatie bevorderlijk, omdat het gemak in uitbreiding van de applicatie hierdoor wordt ondersteund.

2.3.5. Uitgangsituatie

Om de opdracht goed te kunnen uitvoeren is bij de verloskundepraktijk de Maatschap lammenschans een werkplek ingericht. Door het gebrek aan de nodige automatiseringsmogelijkheden is naast deze werkplek een tweede werkplek buiten de organisatie gecreëerd waar de toegang tot het internet en voorzieningen voor e-mail beschikbaar zijn.

De werkplek binnen de organisatie is vooral van belang om de sfeer van organisatie te kunnen verwerken in de toekomstige applicatie. Knelpunten zoals werkdruk en tijd voor het verwerken van de administratieve gegevens. Deze factoren kunnen niet worden opgevangen buiten de organisatie. De secundaire werkplek moet vooral aanvullen in de gebreken van de primaire werkplek.

Er bevindt zich binnen de organisatie geen tot gebrekkige kennis wat betreft automatisering. Daarom is besloten om een externe docent aan te stellen tot raadgever voor het toepassen van de methode.

Tijdens de opleiding informatica en informatiekunde heb ik informatie opgedaan over het gebruik van de technieken van UML. In vrije tijd is kennis opgedaan van de programmeertaal Delphi.

2.4. Plan van aanpak

De eerste activiteit die ik heb uitgevoerd is het opstellen van een plan van aanpak voor het gehele project. Het plan van aanpak wordt gebruikt voor het beheerst uitvoeren van activiteiten voor het bereiken van een eenmalig gesteld doel. In dit document worden afspraken over de opdracht tussen de opdrachtgever en de uitvoerenden gemaakt. Het maken van deze afspraken heeft tot doel om overeenstemming te bereiken met de opdrachtgever over de opdracht en over de uitvoering daarvan.
Door overeenstemming te bereiken wordt de kans verkleind dat de opdrachtgever en de uitvoerenden een verschillend beeld hebben van de uit te voeren opdracht, wat zou kunnen resulteren in een eindproduct dat niet voldoet.
In dit hoofdstuk beschrijf ik de totstandkoming van het plan van aanpak voor dit project. Als eerste beschrijf ik welke aanpak ik heb gekozen om het document op te stellen. Vervolgens behandel ik de totstandkoming van elk onderdeel van het plan van aanpak. Als laatste ga ik in de paragraaf “Afronding” in op de oplevering en afronding van het document.

2.4.1. Afweging soort plan van aanpak

Dit project is van een dusdanige omvang, dat ik voor aanvang van het project een goede basis wilde leggen. De methode en technieken die ik moest gaan gebruiken, waren voor mij vrij nieuw. Om het project daarom goed te kunnen beginnen heb ik overwogen wat het beste plan van aanpak voor dit project zou zijn.
In mijn opleiding heb ik een aantal diverse varianten plan van aanpak toegepast, maar - op een uitzondering nagelaten - nooit voor hele grote projecten. Ik wilde ervoor zorgen dat het plan van aanpak voldeed aan de eisen die ook in de praktijk worden gesteld. Daarom heb ik gekeken naar de eisen waar het plan van aanpak volgens ‘Cap Gemini’ aan moet voldoen. Deze resultaten heb ik vervolgens vergeleken met de methode die wordt aangeleerd aan de Haagse Hogeschool. Vervolgens heb ik gekozen voor een combinatie van het plan van aanpak die door *Cap Gemini* en door de Haagse Hogeschool wordt gehanteerd.

Vervolgens heb ik een hoofdstukindeling gemaakt van het document, waarbij ik de indeling die het gekozen plan van aanpak voorschrijft aan heb gehouden. Vervolgens is per hoofdstuk bekeken of de inhoud hiervan relevant was voor het project.
Hieruit bleek dat het hoofdstuk waarin de kosten worden geanalyseerd niet relevant bleken. Een voorwaarde aan dit project was dat er geen kosten gemaakt mochten worden. Hierdoor worden activiteiten waarin de kosten van het project besproken moeten worden irrelevant.
De overige hoofdstukken bleken wel relevant en zijn in de aankomende subparagrafen beschreven, met uitzondering van “Introductie” en “Projectopdracht”, waarin de aanleiding en opdracht worden beschreven. Dit is reeds in dit hoofdstuk gedaan.

Om een zo volledig mogelijk plan van aanpak te maken heb ik vergelijkbare documenten bestudeerd die ik eerder heb opgesteld. De elementen uit deze documenten die nuttig bleken heb ik aan dit plan van aanpak toegevoegd en eventueel aangepast

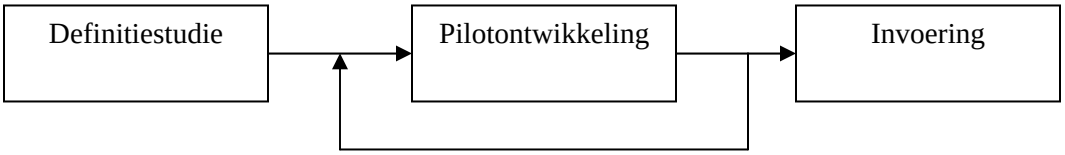
2.4.2. Aanpak

Het hoofdstuk “Aanpak” beschrijft de gekozen methode voor dit project en de activiteiten die per fase kunnen worden uitgevoerd in het kader van het project.
De keuze van de methode is IAD (Interactive Application Development). Hoewel ik geen eerdere ervaringen had met IAD, leek SDM een meer voor de hand liggende keuze. De beslissing om IAD als ontwikkelmethode te hanteren, vloeide voor uit de mogelijkheden om een applicatie iteratief te ontwikkelen. Het watervalmodel dat SDM hanteert was bij dit project niet wenselijk.

Daarnaast is de ontwikkelstrategie voor het project bepaalt.
Binnen de IAD-methode kan worden gekozen uit de vier hoofd ontwikkelstrategieën:

- Evolutionair ontwikkelen
- Incrementeel opleveren
- Incrementeel ontwikkelen (‘RAD’)
- Big-bang-invoeren

Bij IAD is bij een pilotinvoering sprake van het operationeel maken van een pilot, om vervolgens na een aantal weken feedback te ontvangen over deze pilot.
Aangezien één van de randvoorwaarden aan het project is dat het binnen 600 uur vervuld moet zijn, is er geen tijd om te ‘wachten’ op feedback.
Om toch een systeem op te leveren dat voldoet aan de eisen en verwachting van de opdrachtgever, zal hier tijdens de ontwikkeling om meer feedback gevraagd worden. Daarom is er gekozen voor het iteratief ontwikkelen, de RAD-methode.



2.4.3. Projectinrichting en voorwaarden

Het hoofdstuk “Projectinrichting en voorwaarden” beschrijft de samenstelling van de projectgroep. Daarnaast worden de voorwaarden aan het project van de opdrachtgever en opdrachtnemer vastgesteld.
Het doel van dit onderdeel is om de verwachtingen op elkaar af te stemmen, waardoor de rollen van de betrokkenen vast komen te liggen.

De projectgroep bestaat uit één projectuitvoerende, een begeleider ter technische ondersteuning en een opdrachtgever. De invulling die ieder persoon moest geven aan het project was duidelijk.

Na het opstellen van de voorwaarden kon dit hoofdstuk als afgerond worden beschouwd.

2.4.4. Plannen

Om het project beheersbaar te maken en om een schatting te maken van de benodigde tijd, heb ik een planning opgesteld. De planning is gebaseerd op die activiteiten die moeten worden uitgevoerd in het kader van de gekozen systeemontwikkelingmethode. Om een planning te kunnen maken heb ik allereerst per fase een schatting gemaakt van het aantal benodigde weken. Deze schattingen zijn gedeeltelijk gebaseerd op eerdere ervaring die ik heb opgedaan in het werken met een systeemontwikkelmethode. Hieronder is het resultaat daarvan opgenomen.

Mijlpaalproduct	**Weken***
Plan van aanpak	1
Definitiestudie	2
• Reverse Engineering	0.5
Pilotontwikkeling	9
Pilotinvoering	2
Afronden documentatie	0.5

Vervolgens zijn de activiteiten die in iedere fase moest worden uitgevoerd, opgesteld in een detailplanning. Ook in dit stadium is een schatting gemaakt van de tijdsduur die iedere activiteit in beslag zou nemen.

Omdat de fase pilotontwikkeling tijdens de fase definitiestudie wordt verdeeld in meerdere pilot, was het niet mogelijk om bij aanvang van het project een gedetailleerde planning voor deze fase te maken. De verdeling van de te ontwikkelen pilots was namelijk nog niet bekend op dat tijdstip. Voor iedere pilot is in het bijbehorende plan van aanpak een nieuwe planning opgesteld, met daarin de benodigde tijd voor iedere activiteit Indien nodig wordt deze planning na afloop van elke fase bijgesteld aan nieuwe ontwikkelingen en inzichten. De aanpassingen in de planning die tijdens dit project noodzakelijk waren, worden in dit document beschreven in de paragrafen “Afronding”. Deze paragrafen zijn opgenomen bij elk hoofdstuk dat een fase uit het project behandelt.

2.4.5. Kwaliteitsborging

In het hoofdstuk “Kwaliteitsborging” wordt beschreven op welke manier wordt gegarandeerd dat de gewenste resultaten voor zowel het product als proces worden nageleefd.
De eisen waar de resultaten van het project aan moeten voldoen, zijn in dit hoofdstuk onder de proceskwaliteit en productkwaliteit ontwikkeld. De kwaliteit van de producten is hieronder opgenomen.

De opgeleverde rapporten moeten voldoen aan de rapportage-eisen zoals die tijdens de module AV-03 aan de Haagse Hogeschool gesteld worden. Daarnaast moeten de rapporten naast technische informatie ook een uitleg voor de opdrachtgever bevatten. Het informatiesysteem moet met een beperkte uitleg te begrijpen zijn, omdat de gebruikers van het systeem geen technische achtergrond zullen hebben, zal de functionaliteit voor zich moeten spreken.. Naast deze functionaliteit zal tevens een gebruikershandleiding opgesteld worden, die bij eventuele vragen als ondersteuning zal dienen.

Vervolgens is de beschreven op welke manier er getest gaat worden, om de resultaten van de pilots te kunnen controleren. Naast het beschrijven van de voorgestelde maatregelen, is een onderdeel risicomanagement opgenomen. In dat onderdeel worden alle risico’s waarvan verwacht wordt dat ze eventueel zouden kunnen optreden, in een lijst opgenomen. Voor ieder risico is vervolgens vastgesteld wat de impact zou zijn in het project als dit risico zich zou voordoen. Om de impact van het risico enigszins te verkleinen, zijn de stappen opgesteld die worden gehanteerd als maatregel indien zich zo’n situatie voordoet.

2.4.6. Afronding

Om een concept van het plan van aanpak te kunnen opleveren, heb ik de ingeplande tijd voor het opstellen hiervan nodig gehad. Na de bespreking van het concept met de opdrachtgever is deze akkoord gegaan met het opgestelde document.

3. Definitiestudie

Na het opstellen van het plan van aanpak en de uitgebreide bedrijfsoriëntatie, ben ik begonnen met de eerste fase van het IAD-ontwikkeltraject; de definitiestudie. Het doel van deze fase was het in kaart brengen van de eisen en de functionaliteiten van het toekomstige systeem en op basis daarvan een systeemconcept te ontwikkelen.
Naast het ontwikkelen van een systeemconcept wordt in deze fase ook het systeemconcept opgesplitst in onafhankelijk te ontwikkelen systeemdelen, pilots. Dit hoofdstuk beschrijft de totstandkoming van het rapport definitiestudie. In de eerste paragraaf wordt de totstandkoming van het plan van aanpak beschreven. In paragraaf 3.2 wordt ingegaan op de systeemeisen, waarna het systeemconcept in 3.3 behandeld wordt. In paragraaf 3.4 wordt de technische structuur besproken en de organisatorische inrichting in de daaropvolgende paragraaf. Daarna worden de ideeën en beweegredenen van het pilotplan in kaart gebracht om af te sluiten in paragraaf 3.7 met de verificatie en validatie.

3.1. Plan van aanpak

Binnen IAD is de eerste activiteit van de fase definitiestudie het schrijven van een plan van aanpak. Aangezien ik weinig kennis had van IAD, heb ik mij het maken van een plan van aanpak georiënteerd met behulp van het boek van R.J.H. Tolido, “Het Evolutionair ontwikkelen van informatiesystemen”. In dit boek worden een aantal activiteiten onderscheiden die kunnen worden uitgevoerd in het kader van de definitiestudie.
Deze activiteiten heb ik vervolgens allemaal opgenomen in het plan van aanpak. Nadat duidelijk was wat het doel van iedere activiteit was is een van deze activiteiten komen te vervallen. De activiteit “Evalueren Pilot” had in dit stadium geen toegevoegde waarde, aangezien er nog geen pilots waren ontwikkeld.

Vervallen activiteiten:

- *Evalueren Pilot*

Naast de eerder genoemde activiteiten, is de activiteit evalueren pilot moedwillig weggelaten. In dit stadium is er nog geen pilot gemaakt en is het evalueren hiervan niet mogelijk.

Door het gebrek aan ervaring was het vrij lastig om een inschatting te maken van de te besteden tijd bij ieder onderdeel. Op basis van ervaring met eerdere projecten is besloten een inschatting te maken van de diepgang van de activiteiten en daaruit is een planning tot stand gekomen. Normaliter worden in een planning uren gebruikt als tijdsaanduiding. Mijn voorkeur ging uit naar een indeling in minuten. Voor deze indeling is gekozen omdat het gebruik van kommagetallen in uuraanduiding mijns inziens slordig oogt. Hoewel er niet wordt gelet op iedere minuut, vond ik 15 minuten voor een activiteit een professionelere weergave dan 0,25 uur.

Product / Proces	**Tijdsduur in minuten**
Het ontwikkelscenario	240
Pilotworkshop	780
De definitie van systeemeisen	2180
Het systeemconcept	480
Technische structuur	480
Organisatorische inrichting	160
Het pilotplan	1200
Verificatie en Validatie	700
Totaal aantal minuten	**6220**

Afbeelding 1: Ingekorte activiteitenplanning

In het totale plan van aanpak zijn 105 uur gepland voor de fase definitiestudie, omgerekend is dat 6300 minuten. Met eventuele uitloop is rekening gehouden en vandaar dat 80 minuten niet is ingepland.

Naast het plannen van de activiteiten is ook het ontwikkelscenario bepaald. De resultaten van dit onderzoek zijn verwerkt in het plan van aanpak voor de gehele opdracht.

3.2. Pilotworkshop

Een cruciale activiteit in de definitiestudie voor dit project is de pilotworkshop. Het doel van een pilotworkshop is om alle betrokken bij de systeemontwikkeling bij te laten dragen aan de systeemeisen, de functionele en technische specificaties van het systeem. Hoewel ik geen eerdere ervaring met het opstellen van pilotworkshop had, leek het mij een prima uitdaging om de toekomstige gebruikers te betrekken bij het project en tevens mezelf bij de organisatie.
De betrokkenheid van de gebruikers dacht ik te kunnen stimuleren door het houden van een kleine presentatie over wat mijn taak binnen de organisatie was. In deze presentatie moest ook duidelijk worden hoe de gebruikers invulling konden geven aan hun toekomstige applicatie.

Allereerst heb ik de activiteit “Zet workshopaccommodatie op” uitgevoerd. Door gebrek aan ervaring wist ik niet precies hoe ik hiermee moest beginnen. R.J.H. Tolido geeft een nogal ruime verklaring voor wat het doel van de workshop is en op basis daarvan heb ik geïnterpreteerd wat zijn intentie hiermee was. Een lijst met systeemeisen, een aanzet tot het systeemconcept, beschrijving van de technische architectuur en relevante documentatie waren hiervan een onderdeel. Om de verloskundigen een indruk te geven van mijn taak en hun taak in de ontwikkeling ben ik begonnen met het voorleggen van het materiaal en het opstellen van een aantal vragen die voor mij van belang waren:

Workshop Verloskunde-Applicatie

- *Wat zijn de struikelblokken waar jullie nu tegen aanlopen*
- *Wat vinden jullie makkelijk aan de huidige applicatie*
- *Hoe stellen jullie de nieuwe applicatie voor.*
- *De voorkeuren voor een nieuwe applicatie*

Na het uitkiezen van een geschikte locatie, moest er een datum geprikt worden waarop iedereen bij elkaar kon komen. Aangezien één van de verloskundigen altijd dienst heeft, was het risico dat die persoon weg werd geroepen aanwezig. Uiteindelijk was een vrijdag het meest geschikt, aangezien op die dag altijd de vergadering gehouden wordt en iedere verloskundige op de praktijk aanwezig is.

De eerste workshop verliep niet heel vlot. Naar mijn mening was de ongeïnteresseerde houding en het gebrek aan ICT-kennis binnen de organisatie een van de redenen dat dit niet heel vlot verliep. Daarnaast was het voor mij ook vrij nieuw en was het verkrijgen van het juiste resultaat uit de workshop vrij lastig. De verwachting was dat ik vrij veel feedback zou krijgen, zodra ik had voorgelegd hoe het systeem in de toekomst zou moeten gaan werken. Dit was niet het geval. De feedback was nogal mager, met reacties als “Dat ziet er wel anders uit.”.

Na de uitvoering van de eerste workshop, die niet zo vlot verliep, had ik besloten om nogmaals een workshop te houden met een andere soort voorbereiding. De vragen die ik opgesteld had, waren nu verwerkt in het systeemconcept. Bovendien was er, in plaats van de producten die Tolido voorschrijft, een lijst opgesteld met punten waar de huidige applicatie mijns inziens tekortschiet en de verbeteringen die kunnen worden doorgevoerd in de toekomstige applicatie.

De reacties op de lijst met verbeterpunten leidde tot een productieve workshop, waarbij de verloskundigen ook spontaan met ideeën kwamen. Op die manier was er soort kruisbestuiving ontstaan.

3.3. Systeemeisen

Na het uitvoeren van de tweede workshop kon op basis van de opgestelde lijst met verbeterpunten en de aanvulling van de verloskundigen een gedeelte van de systeemeisen worden vastgesteld. Deze eisen waren veelal niet-functionele systeemeisen. Voor de functionele systeemeisen is op de huidige applicatie Orfeus een eigen manier van reverse-engineering toegepast. De diepgang was vrij globaal, omdat er in de gebruikersovereenkomst van Orfeus staat dat reverse-engineeren op de applicatie verboden is en hiermee het contract direct kan worden ontbonden.

Die zelf opgestelde aanpak bestond uit een aantal stappen. Allereerst waren alle functionaliteiten die de bestaande applicatie te bieden had doorlopen en is de globale werking hiervan op papier vastgelegd. Vervolgens zijn de functies in kaart gebracht die door een verloskundige met regelmaat gebruikt worden. Tot slot is de structuur van de database in grote lijnen ook op papier vastgelegd.

De functionele systeemeisen moesten dusdanig worden geformuleerd dat deze meetbaar waren. De niet-functionele eisen zijn de eisen die niet inhoudelijk met het systeem te maken hebben, maar die randvoorwaarden stellen aan de manier waarop de functionele eisen dienen te worden geïmplementeerd.

3.3.1. Functionele systeemeisen

De functionele systeemeisen die uit de workshops zijn ontstaan, hadden niet allemaal dezelfde prioriteit. Om de ontwikkeling van het systeem op basis van de functionele systeemeisen gestructureerd aan te pakken, moest er worden achterhaald welke basiseisen er aan de applicatie gesteld worden en welke eisen minder belangrijk zijn. In een gesprek met de opdrachtgever moest duidelijk worden welke eisen cruciaal zijn voor het systeem.

Om de eisen prioriteiten toe te kennen is gekozen voor het gebruik van de MoSCoW analyse. De MoSCoW analyse kent de volgende indeling in prioriteiten:

- · Must Have (M);
- · Should Have (S);
- · Could Have (C);
- · Would like to but probably cannot have (W).

Nadat ik de vastgestelde eisen een prioriteit had toegekend en verdeeld in functionele en niet-functionele eisen, heb ik het resultaat hiervan besproken met zowel mijn opdrachtgever als mijn externe begeleider Na een herformulering van enkele systeemeisen – die ervoor zorgden dat deze beter meetbaar waren – was de opdrachtgever tevreden met het resultaat. De uiteindelijke lijst is hieronder opgenomen:

Systeemeis	Prioriteit
Het systeem moet de gegevens op de oude zwangerschapskaart kunnen realiseren.	M
De gegevens van cliënten die verwerkt worden in het systeem moeten grotendeels variabel zijn. (Doktoren, ziekenhuizen, ziektes, etc)	M
Het moet mogelijk zijn om te zoeken naar de cliënten op eventueel unieke gegevens	M
Het moet mogelijk zijn om ziekenhuizen bij te werken en er in te zoeken	M
Ziekenhuizen, verzekeraars, gynaecologen, doktoren, plaatsnamen, achternamen en voornamen. Deze gegevens moeten allemaal in een ‘zelfgroeiende’ lijst kunnen worden geselecteerd, zodra het systeem vereist dat deze worden ingevoerd.	M
Het systeem zal middels een functie de cliëntdata moeten kunnen exporteren naar een locatie buiten het systeem.	S
De gebruiker en/of beheerder kan alleen toegang krijgen tot het systeem middels een gebruikersnaam en wachtwoord	S
Er moet een mogelijkheid zijn om de gegevens te rapporteren	S
De LVR (Landelijke Verloskunde Registratie) vereist dat er periodiek een formulier wordt toegezonden met het aantal cliënten en extra gegevens, deze moet door het systeem kunnen worden gegenereerd.	C
De instellingen binnen de applicatie moeten kunnen worden veranderd door een beheerder	C
Gebruikers van het systeem moeten kunnen worden toegevoegd	C

De cliëntgegevens moeten worden gecontroleerd aan de hand van de voorwaarden van het LVR	W
Alle kosten die gemaakt worden in het cliënttraject moeten kunnen worden gerapporteerd	W

3.3.2. Niet-functionele systeemeisen

De niet-functionele systeemeisen hadden vooral betrekking op de gebruikersvriendelijkheid en de zekerheid dat de data waarmee gewerkt wordt correct is, en niet verouderd. Onderstaand zijn de niet-functionele systeemeisen opgenomen:

Systeemeis
De grafische user interface moet overzichtelijk zijn
Er mogen niet meerdere gebruikers tegelijkertijd met het systeem werken
Het systeem moet werken op een werkstation met een Windows operating systeem
Het systeem zal dag en nacht moeten kunnen draaien.
Registratie van eerste gesprek met cliënt mag maximaal een half uur duren
Foutmeldingen moeten concreet en begrijpelijk zijn

3.4. Systeemconcept

Op basis van de systeemeisen en het globale concept uit de workshop heb ik vervolgens een gedetailleerder systeemconcept ontworpen. Dit systeemconcept is een eerste echte opzet van het toekomstige systeem en dient als basis voor het pilotplan en de voorgenomen pilots. De gebruikte techniek voor het ontwerp van het systeem zijn de technieken van UML. UML is een techniek die object georiënteerd is. Na het raadplegen van het boek Praktisch UML, bleek dat het use-case diagram en het klassendiagram goed toepasbaar waren in deze situatie.

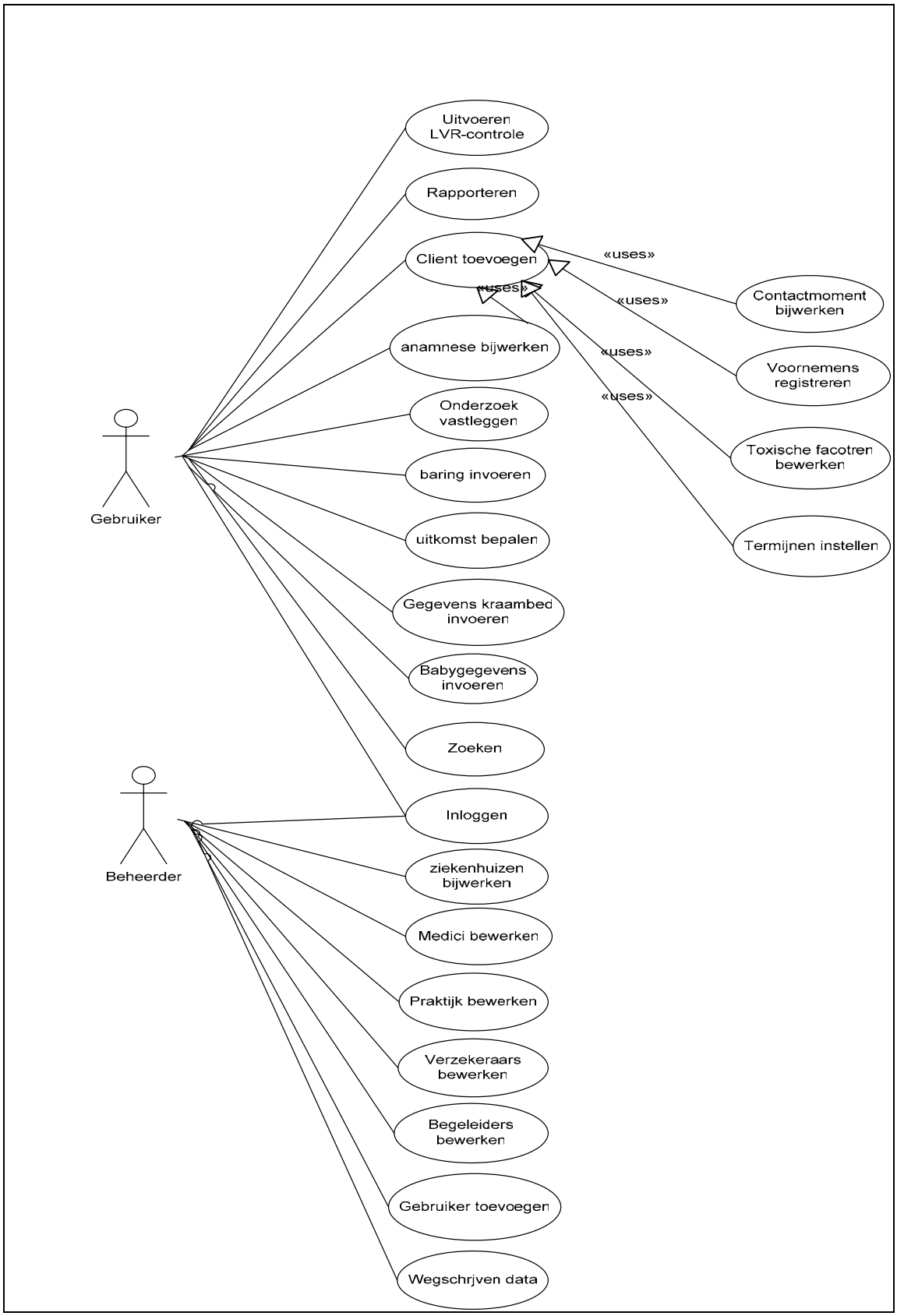
3.4.1. Use-case diagram

De toekomstige applicatie zal, net als in de huidige situatie, gebruikt gaan worden voor de registratie van gegevens van de cliënten, zoals personalia, medische geschiedenis en zwangerschap. Iedere gebruiker heeft evenveel rechten en zal een aantal standaardfuncties moeten kunnen uitvoeren. Dit leidt tot de actor ‘Gebruiker’: een gebruiker van het systeem met toegang tot alle handelingen die een verloskundige nodig heeft om de noodzakelijke gegevens te kunnen registreren.

Naast deze actor is er ook de actor ‘Beheerder’. De actor Beheerder is bevoegd tot het uitvoeren van functionaliteiten die te maken hebben met de niet-dagelijkse handelingen. Voorbeelden hiervan zijn het veranderen van data omtrent een ziekenhuis of een dokter, of een verhuizing van de praktijk.

Het use-case diagram bestaat hierdoor uit twee actoren:
- De gebruiker
- De beheerder

Na het vaststellen van de actoren, moesten er use-cases worden ontworpen. De functionele systeemeisen zijn allereerst goed bekeken om te kijken welke functies het toekomstige systeem moest gaan bieden. Op basis van deze eisen en de resultaten van de workshops zijn de globale functies van het systeem ontworpen. Dit heeft geleid tot het volgende use-case diagram.



Afbeelding 2: Definitieve use-case diagram

Voor iedere functionele systeemeis is op deze manier minimaal één use-case ontworpen, wat ervoor zorgt dat alle eisen door het systeem gedekt gaan worden.

Om meer inzicht te krijgen in de diepgang van iedere use-case zijn er beschrijvingen opgesteld. Deze beschrijvingen zijn vooral gericht op de gebruiker, de reactie van het systeem is weggelaten. Een voorbeeld van een use-case is hieronder weergegeven.

Use-case	Uitkomst bepalen
Preconditie	De cliënt is bekend in het systeem
Postconditie	De uitkomst van de zwangerschap van de cliënt is in het systeem vastgelegd
Beschrijving	1. De uitkomst wordt vastgelegd 2. De einddatum en de eventuele eindstatus van de zorg worden vastgelegd 3. Aantal vruchten / kinderen wordt vastgelegd, afhankelijk van de uitkomst. 4. Eventuele medische indicatie wordt bepaald

Afbeelding 3: Voorbeeld van een use-case beschrijving

In de use-case *uitkomst bepalen*, kan worden vastgelegd wat de uitkomst van de zwangerschap zal zijn. Dit wordt gedaan op basis van echo's en labuitslagen. Voordat echter de uitkomst van de zwangerschap bepaald kan worden, moet eerst de cliënt bekend zijn in het systeem.

3.4.2. Klassendiagram

Na het opstellen van de use-case beschrijvingen, schrijft UML voor om een klassendiagram te maken om een statische weergave van het systeem te krijgen. Om tot een volledig klassendiagram te komen moest eerst worden onderzocht welke klassen in het klassendiagram moesten komen.

Het opstellen van een groot klassendiagram was voor mij vrij nieuw, omdat mij aangeleerd is om met de technieken van Yourdon te werken. Deze technieken zijn op sommige gebieden vergelijkbaar, maar in veel opzichten verschillend. Bovendien is de techniek waarmee klassen voor het klassendiagram verkregen moesten worden hier niet toepasbaar. Deze techniek is dat alle zelfstandige naamwoorden uit een verkregen case worden opgeschreven en de 'echte' klassen op basis daarvan worden gedefinieerd.

In UML wordt beschreven dat eerst een zo groot mogelijke lijst met kandidaat-klassen moet worden gemaakt door middel van een brainstormsessie en het onderstrepen van de zelfstandige naamwoorden uit de probleemstelling. De resultaten hiervan konden worden gecombineerd tot een grote lijst met kandidaat-klassen. Naast het onderstrepen van de zelfstandige naamwoorden in de probleemstelling, zijn ook de zelfstandige naamwoorden uit de resultaten van de evaluatie van de huidige applicatie en de systeemeisen opgenomen in deze lijst.

De brainstormsessie was echter vrij lastig alleen te houden, dus is ervoor gekozen om samen met de opdrachtgever een korte brainstormsessie te houden om tot een volledige kandidaat-lijst te komen. De resultaten van de brainstormsessie zijn aan de lijst toegevoegd.

Met deze lange lijst van kandidaat-klassen kon worden begonnen om deze te verkleinen. Aangezien er in de eerste fase van een brainstormsessie ooit een fout antwoord is, bevatte de lijst zeer veel redundante kandidaat-klassen..

Uit deze lijst met klassen is vervolgens een selectie gemaakt op basis van de door UML voorgeschreven criteria. De eerste stap is het modelleren van de personen. Bovendien moesten de klassen aan de volgende criteria voldoen:

- De klasse is duidelijk
- Redundante klassen mogen niet meer voorkomen
- De klasse is niet irrelevant
- De klasse heeft geen ‘vage’ naamgeving.
- De klasse mag niet kwantificeerbaar zijn
- De klasse is geen attribuut

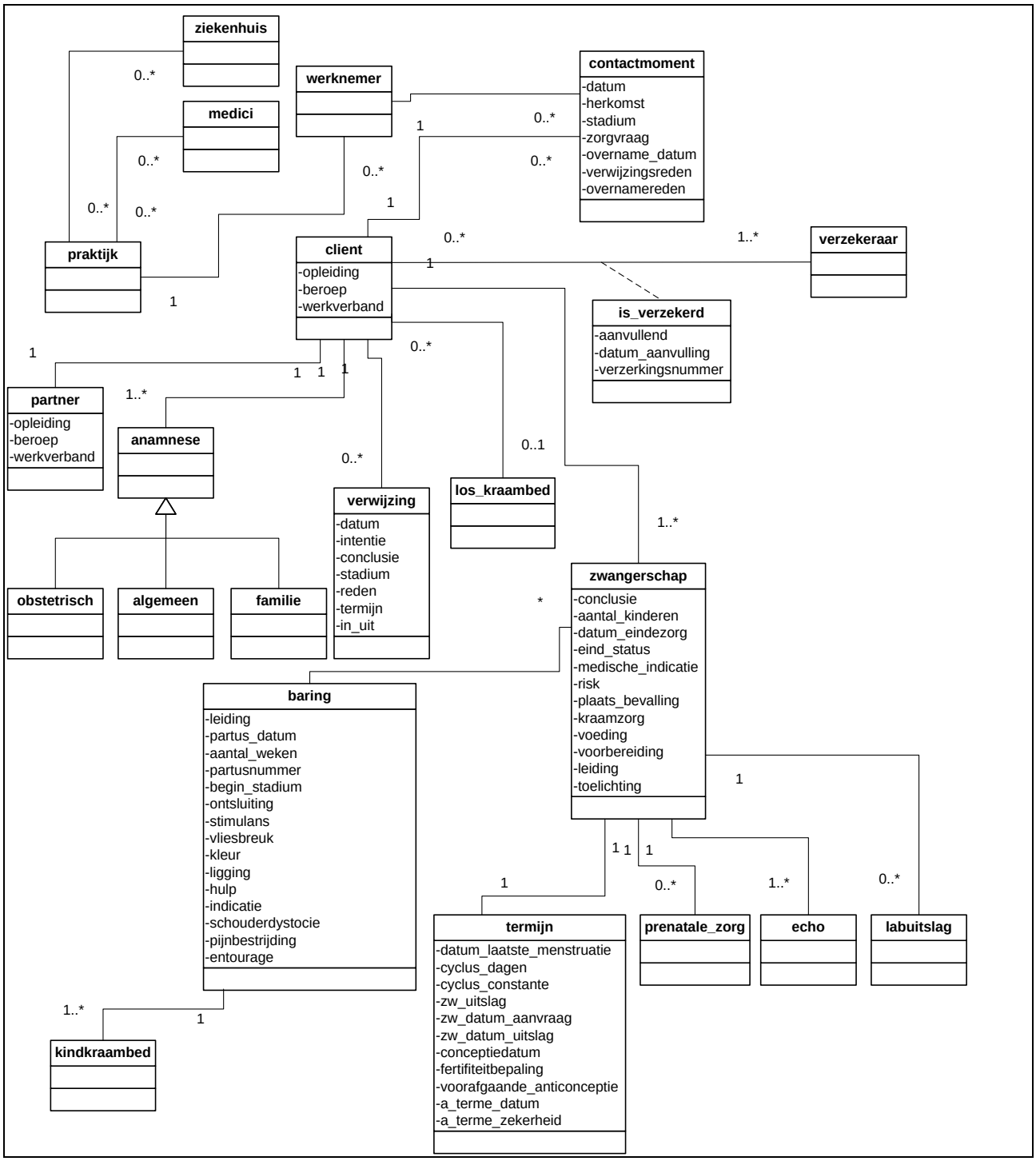
Na het doorlopen van alle stappen was de lijst met klassen tot een aantal van 27 gereduceerd. Vervolgens moesten de associaties worden geïdentificeerd. Een echte aanpak is hier niet voor. Dit maakte het tevens erg moeilijk om direct de juiste associaties te maken tussen de diversen klassen. De gekozen aanpak om de associaties te identificeren was om op een logische manier de klassen met elkaar te verbinden. Zo had de klasse *partner* uiteraard geen associatie met de klasse *termijn* (termijn van een zwangerschap).

Nu alle klassen in het klassendiagram waren opgenomen had ik, na een uitgebreid overleg met mijn begeleider over de overzichtelijkheid van een klassendiagram en het doel van een model, besloten om nogmaals de zes criteria te doorlopen. Het resultaat hiervan was, dat na een uitgebreide discussie ik kandidaat-klassen die ik aan het klassendiagram had toegevoegd, uiteindelijk opgenomen heb als attribuut van een echte klasse. De volgende klassen waren geen opzichzelfstaande klassen:

- Placenta
- Trauma
- Tijd
- Aanwezig
- Kind
- Kraambed
- Nacontrole
- Ziektebeeld

Het klassendiagram had op dit moment 21 klassen en de relaties tussen enkele klassen moesten worden vervangen of weggehaald. Uiteindelijk had ik besloten om het

klassendiagram voorlopig zo te laten en in een later stadium kon ik eventueel nog veranderingen aanbrengen. Het opstellen van een klassendiagram moet eigenlijk ook iteratief worden aangepakt en voor een volledig diagram kan er anderhalf tot twee jaar verstrijken. Het definitieve klassendiagram is opgenomen in afbeelding 4.



Afbeelding 4: Definitieve klassendiagram

Wat in dit klassendiagram waarschijnlijk direct opvalt is dat de attributen en operaties ontbreken en dat het klassendiagram uit 22 klassen bestaat. Het laatste komt omdat de klasse ‘is_verzekerd’ er in een later stadium aan toe is gevoegd.

De attributen en klassen zijn weggelaten in het diagram om het overzicht te kunnen bewaren. Als alle attributen per klasse zouden zijn opgenomen, wordt het onoverzichtelijk en dat is in strijd met het doel van een klassendiagram. Daarom is besloten om de attributen en operaties van de klassen op te nemen in de pilots

3.4.3. Subsystemen

De use-cases vormden de functionaliteiten die het toekomstige systeem later moest gaan bieden. Om in een later stadium de functionaliteiten gefaseerd te kunnen ontwikkelen is besloten om deze onder te verdelen in groepen, zogeheten subsystemen. Deze subsystemen stellen een verzameling functies voor die eventueel in een later stadium als separate pilot kunnen worden ontwikkeld.

De indeling is gemaakt op basis van de momenten waarop de verloskundigen de applicatie gebruiken. Deze zijn te onderscheiden in:

1. Eerste contactmoment cliënt
2. Zwangerschap
3. Administratie
4. Beheren

Deze vier subsystemen hebben de naamgeving te danken aan het stadium waarin de verloskundige de applicatie gebruikt. Subsysteem drie is wellicht een erg algemene naamgeving, maar zodra een van de verloskundigen de administratie moet bijwerken wordt een van de use-cases “rapporteren” of “LVR-controle” geraadpleegd. Die allebei van belang zijn voor de financiële administratie. De inhoud van de overige drie subsystemen is toegelicht in het document definitiestudie.

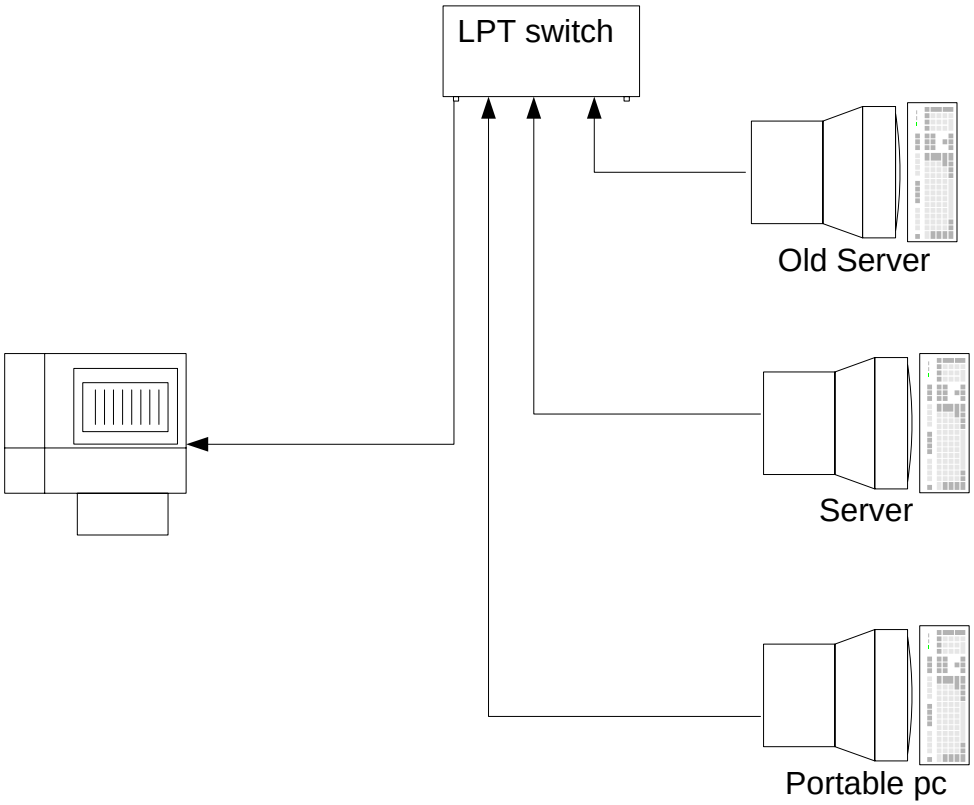
3.5. Technische structuur

De activiteit technische structuur omvat het testen van de huidige technische omgeving voor de realisatie van het systeemconcept. Het doel hiervan is het voorkomen van onvoorziene technische knelpunten en eventuele aanpassingen in een laat stadium van de opdracht. De twee eisen gesteld aan de technische structuur in de niet-functionele systeemeisen zijn:

- Het systeem moet werken op een werkstation met een Windows operating systeem
- Het systeem zal dag en nacht moeten kunnen draaien

De organisatie heeft kort geleden twee nieuwe werkstations met een goedwerkend Windows XP besturingssysteem aangeschaft, waarmee de eerste niet-functionele systeemeis werd gedekt. Het was ook geen probleem om één van de twee werkstations dag en nacht te laten opereren.

De werkstations waren echter niet verbonden in een Local Area Network. De huidige methode om de data consistent te houden, was door middel van het uitwisselen via een ZIP-diskette.



Afbeelding 5: Huidige technische structuur

Hierdoor is de niet-functionele systeemeis “Er mogen niet meerdere gebruikers tegelijkertijd met het systeem werken” onmogelijk te handhaven. Bovendien mochten er bij dit project geen kosten gemaakt worden. In een poging om het belang van deze situatie te schets in het licht van de vermoedelijke financiële consequenties, heb ik besloten om een analyse te maken van de kosten voor de aanleg van een Local Area Network.

Randapparatuur	Kosten in euro
Sweex Switch 10/100mbps	€ 13,75
15 Meter Cat 5e UTP kabel	€ 3,98
15 Meter Cat 5e UTP-kabel	€ 3,98
3 Meter Cat 5e UTP-kabel	€ 2,00
Aanleg	€ 25,00
Totale kosten	€ 48,71

Mijns inziens betrekkelijk lage kosten voor de aanleg van het netwerk, de eerste poging bleek echter tevergeefs. In deze huidige situatie was de conclusie dat de huidige structuur

grotendeels voldeed, maar er moest een alternatief bedacht gaan worden voor het consistent houden van de database.

Aangezien dit mijn project op een dusdanige manier belemmerde, heb ik mij na een aantal weken aan een tweede poging gewaagd in een vergadering. Na het schetsen van een aantal risico's en het wijzen op de verantwoordelijkheden van de BIG-wet, was de praktijk bereid om in het nieuwe jaar een netwerk aan te laten leggen voor het bedrag dat ik in de eerste instantie had berekend.

3.6. Organisatorische inrichting

Om de gebruikers wat meer te betrekken bij het project, is een kleine toelichting gegeven op de organisatie en manier waarop de gebruikers kennis gaan maken met het eindproduct. Naast de workshops en feedback van de opdrachtgever zullen de functionaliteiten en enkele belangrijke punten van het eindproduct gepresenteerd worden.

Dit is moedwillig onder dit kopje geplaatst, aangezien de nieuwe applicatie de gehele werkwijze van de gebruikers zal gaan veranderen. Ze zullen nieuwe stappen moeten aanleren en moeten wennen aan een vernieuwde applicatie. Wat uiteindelijk op kleine schaal grote organisatorische gevolgen heeft.

3.7. Pilotplan

Na het in kaart brengen van de organisatorische veranderingen, is de volgende activiteit in het definitiestudie-traject het opstellen van een pilotplan. Het pilotplan fungeert als een geprioriteerde lijst van pilots. Iedere pilot stelt bovendien een op zichzelf staand deelsysteem voor, dat onafhankelijk van de andere systemen kan worden geïmplementeerd.

IAD onderscheidt twee soorten pilots:

- kernel-pilots
- pilots

De eerste categorie zijn pilots die snel ontwikkeld en opgeleverd moeten worden, opdat daar dan feedback over verzameld kan worden.
De tweede categorie zijn de 'normale' pilots, waar geen eis aan de snelheid van oplevering wordt gesteld.

Voor het ontwikkelen van de applicatie, gezien deze twee soorten pilots, heb ik een indeling gemaakt waar de applicatie in grote lijnen uit zal bestaan. Waar de verloskundigen vooral mee te maken hebben is vooral de user interface. Naast de interface is er ook de gegevensopslag, de database. De communicatie tussen beide wordt verzorgd door de functionaliteiten van de applicatie. Deze drie delen vormen ook direct de basis voor het opzetten van de pilots:

- De GUI (Grafische User Interface)
- De functionaliteiten
- De gegevensopslag (de database).

Om de verkregen feedback in de workshops snel te kunnen terugkoppelen in iets wat voor de gebruiker 'tastbaar' is, is besloten om de Grafische User Interface onder te verdelen als een opzichzelfstaande kernel-pilot. Naast de GUI is ook de database onderverdeeld als kernel-pilot. Hiermee is de basis van de applicatie gelegd.

Omdat het ontwerpen en implementeren van de functionaliteiten een qua omvang vrij grote pilot zou worden, is besloten om deze verder onder te verdelen. Gelet op de systeemeisen zijn er twee pilots ontwikkeld. Subsysteem één tot en met drie bevatten de "Must Have", "Should Have" en "Could Have" eisen en het laatste subsysteem bevat de "Would Have" eisen. Dit is ook direct de onderverdeling van de pilots geworden.

In dit stadium van het project is tevens een laatste kernel-pilot gedefinieerd die als interface tussen de database en de GUI zal fungeren.

Kernel pilots:
1. *GUI*
2. *Database*
3. *GUI-database interface*

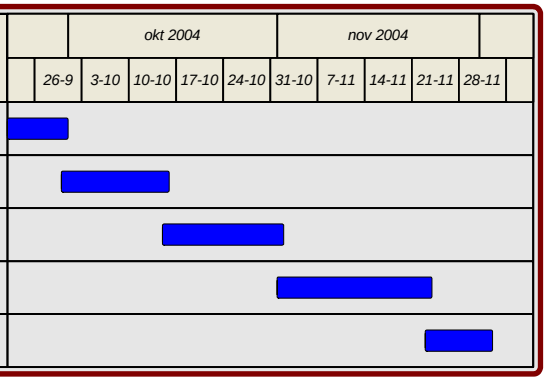
Normale pilots:
1. *Subsystemen 1 tot en met 3*
2. *Subsysteem 4*

Afbeelding 6: Gedefinieerde pilots

Doordat alle functionele systeemeisen worden gedekt door de subsystemen, was het niet verder noodzakelijk om deze nogmaals te controleren of deze ook door de pilots werden gedekt.

Na het definiëren van de pilots heb ik een schatting gemaakt van de tijdsduur die elke pilot in beslag zal gaan nemen. Op basis hiervan is een globale planning tot stand gekomen.

ID	Pilotnaam	Begindatum	Einddatum	Tijdsduur (dagen)
1	Grafische User-Interface	22-9-2004	30-9-2004	7d
2	Database	30-9-2004	15-10-2004	12d
3	GUI-Database Interface	15-10-2004	1-11-2004	12d
4	Subsystemen 1 tot en met 3	1-11-2004	23-11-2004	17d
5	Subsysteem 4	23-11-2004	2-12-2004	8d



Afbeelding 7: Pilotplanning

3.8. Verificatie en validatie

De laatste activiteit binnen de fase definitiestudie van de IAD-methode, is de verificatie en validatie. Hierin is de mogelijkheid om de onderdelen en tussenproducten die binnen de definitiefase tot stand zijn gekomen te controleren op correctheid.
De tussenproducten die mij van belang leken om te controleren waren:

- Het use-case diagram
- De functionele systeemeisen
- De subsystemen
- De pilots
- Het klassendiagram
- Niet-functionele systeemeisen

Om alle onderdelen te verifiëren zijn verbanden gelegd die mijns inziens logisch leken en waarmee goed te controleren valt of de producten consistent met elkaar zijn. Daaruit zijn de volgende verbanden ontstaan:

- Use-case – functionele systeemeisen
  Dit verband is gelegd om te controleren of alle gestelde functionele systeemeisen gedekt zijn door de use-cases in het use-case diagram.

- Use-case – subsystemen
  Dit verband controleert of alle use-cases zijn ondergebracht in de subsystemen die opgesteld zijn.

- Functionele systeemeisen – pilots
  Dit verband is gelegd om het bovenstaande nogmaals te controleren. Aangezien alle functionele systeemeisen in de subsystemen verwerkt zouden moeten zijn. Dit verband zou moeten uitwijzen of dit daadwerkelijk zo is en de opgestelde pilots zodoende correct zijn.

- Use-case – klassendiagram
  Dit verband controleert de juistheid van het klassendiagram, door te kijken of één of meerdere use-cases in het use-case diagram de klassen in het klassendiagram nodig hebben.

3.9. Afsluiting

Na de rapportcontrole op volledigheid is dit document als definitief gemarkeerd. Dit rapport heb ik vervolgens aan mijn opdrachtgever en de overige toekomstige gebruikers laten lezen, zodat zij een indruk kregen van de procesgang.
De opdrachtgever had naast een aantal vragen over de procesgang en de het doel van een pilot verder geen opmerkingen.

Voor de fase definitiestudie heb ik zes dagen meer tijd nodig gehad dan ingepland, dit kwam door de uitgebreide twee workshops en het opstellen van het klassendiagram. In de planning waren er twee weken niet ingepland, deze konden worden beschouwd als uitloop. De zes dagen die ik in deze fase als vertraging had opgelopen, veroorzaakten hierdoor geen direct probleem.

4. Pilot 1: Grafische User Interface

Na oplevering en goedkeuring van het rapport definitiestudie, is de pilotontwikkelfase gestart. Deze fase is het vervolg van de definitiestudie en heeft als doel het detailleren, ontwikkelen en realiseren van de pilots die in de vorige fase zijn gedefinieerd. In de definitiestudie is de eerste pilot, de *Grafische User Interface*, in een reeks van vijf. Dit hoofdstuk beschrijft de activiteiten die in het kader van deze pilot zijn uitgevoerd, met in paragraaf 4.1 het plan van aanpak. De functionele en technische structuur zijn in respectievelijk paragraaf 4.2 en 4.3 behandeld. Het pilotontwikkelplan in paragraaf 4.4 gaat in op de pilotdelen en bouweenheden, waarvan het ontwerp in paragraaf 4.5 en de bouw in paragraaf 4.6 aan bod komt. Vervolgens wordt in paragraaf 4.7 de invoeringsprocedure toegelicht, waarna in paragraaf 4.8 het testen en beoordelen van de pilot behandeld wordt. Dit hoofdstuk wordt afgesloten met de paragraaf afronding.

4.1. Plan van aanpak

Binnen de pilotontwikkeling stelt IAD dat er net als bij de definitiestudie een plan van aanpak wordt gemaakt. Om een goede planning te maken voor deze pilot heb ik allereerst gekeken naar welke activiteiten in het kader van een pilot moeten worden uitgevoerd. Voor iedere activiteit heb ik vervolgens gekeken of deze relevant was voor het project. De niet uitgevoerde activiteiten staan in het plan van aanpak genoemd, met de reden waarom deze niet zijn uitgevoerd.

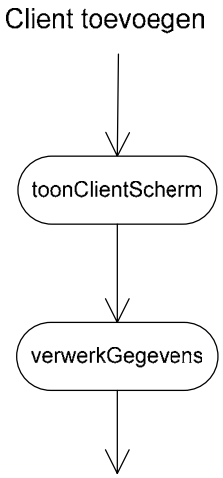
Van de overgebleven activiteiten is op basis van de beschrijvingen in het boek van Tolido een schatting gemaakt van de tijd die de uit te voeren werkzaamheden in beslag gingen nemen.

4.2. Functionele structuur

De activiteit die volgt na het plan van aanpak is het onderzoeken en ontwerpen van de functionele structuur van de pilot. In het geval van deze kernel-pilot is de functionaliteit vooral te zoeken in het ontwerpen van de schermen op een logische manier.
Naast het logisch ontwerp moest de interface ondersteuning bieden aan het toekomstige ontwerp van de functionele systeemeisen.

In het kader van de functionele structuur is de UML-techniek van activiteitsdiagrammen gebruikt om te bepalen welke schermen er ontwikkeld gaan worden. De activiteitsdiagrammen zijn globaal opgesteld voor iedere use-case van de actor *gebruiker* in het use-case diagram. De use-cases van de actor ‘beheerder’ bieden geen toegevoegde waarde aan het ontwerp.

Het activiteitsdiagram geeft inzicht in hoe een proces verloopt en welke subprocessen en / of beslissingen daarbij moeten worden uitgevoerd.



De activiteitsdiagrammen hebben een staat waarin deze een scherm vereisen. De wijze waarop deze schermen worden vertaald naar de GUI wordt in de technische structuur verder toegelicht.

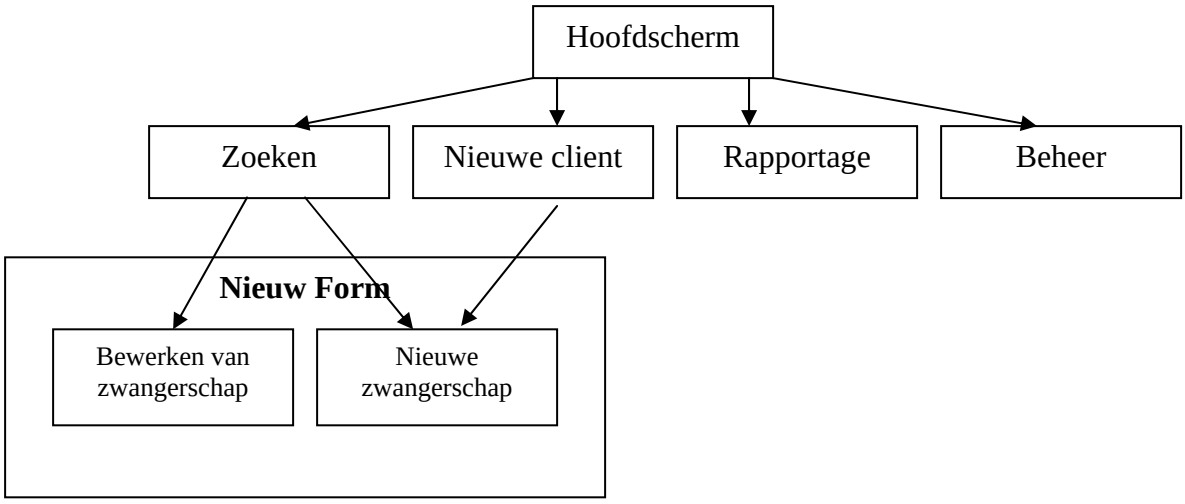
4.3. Technische structuur

Op basis van het systeemconcept, het klassendiagram, de functionele structuur en de niet-functionele systeemeisen is de technische structuur ontwikkeld. De applicatie is in grote mate een visuele weergave van de database, vandaar dat ook het klassendiagram wordt meegenomen in het ontwerp van de technische structuur.

De technische structuur van deze pilot is bedoeld om het juiste aantal schermen te ontwikkelen en in het kader van overdraagbaarheid de naamgeving vast te leggen.

Met het ontwikkelen van het juiste aantal schermen wordt bedoeld dat er niet teveel ‘geklikt’ mag worden, in verband met de gebruikersvriendelijkheid. Bovendien moeten de gegevens snel toegankelijk zijn en overzichtelijk worden weergegeven. Vandaar dat er in overleg met de opdrachtgever een aantal alternatieven voorgelegd zijn in hoe de functionaliteiten toegankelijk zouden worden.

Met het uiteindelijke resultaat dat er twee verschillende schermen (zogeheten forms) worden ontworpen met één extra inlogscherm.



Afbeelding 8: Functionaliteiten met schermontwerp

In bovenstaande afbeelding is te zien hoe het hoofdscherm is onderverdeeld in vier categorieën. Deze categorieën vormen op zichzelf geen nieuw scherm, maar zijn als functionaliteit onderdeel van het hoofdscherm.
De zoekopdracht kan resulteren in een nieuw scherm, zodra er een bewerking of een nieuwe zwangerschap wordt geselecteerd.
In het geval van een nieuwe cliënt zal ook een nieuw scherm worden geopend. Dit scherm zal echter hetzelfde scherm zijn als die van de zwangerschap, aangezien dat conform de reuseability programmeereisen is volgens het boek van Ghezzi: “Software Engineering”.

De functionaliteiten die betrekking hebben op de verwerking van de gegevens van de cliënt, zijn onderverdeeld in bepaalde groepen. Deze groepen zijn op een dusdanige manier ontworpen dat iedere groep een aantal klassen uit het klassendiagram bevat. Deze manier van ontwerp is gekozen omdat de applicatie een visuele weergave van de gegevens moet gaan voorstellen.

Het cliëntscherm zal een nieuw navigatiemenu bevatten door middel van een menu. Alle onderdelen van de cliënt zullen verdeeld worden onder een aantal gegroepeerde frames. Deze frames worden verborgen en getoond afhankelijk van de optie die in het menu wordt aangeklikt. De groepsnamen zijn allereerst in secties verdeeld en hebben bepaalde code toegekend gekregen, zodat de tabellen overzichtelijk blijven

Code	Groepsnaam
G1	Anamnese
G2	Partus
G3	Cliënt
G4	Contact
G5	EchoLab
G6	LosKraambed

G7	Verwijzingen
G8	PrenataleZorg

Afbeelding 9: Groepsnamen in cliëntscherm

Klassenaam	G1	G2	G3	G4	G5	G6	G7	G8
Algemeen	**X**							
Cliënt			**X**					
Contactmoment				**X**				
Echo					**X**			
Familie	**X**							
KindKraambed		**X**						
Labuitslag					**X**			
Los Kraambed						**X**		
Obstetrisch	**X**							
Onderzoek				**X**				
Partner	**X**		**X**					
Partus		**X**						
Prenatale_zorg								**X**
Termijn				**X**				
Verwijzingen							**X**	
Verzekeraar			**X**					
Zwangerschap				**X**			**X**	

Afbeelding 10: Klassen uit het klassendiagram afgezet tegen de groepen

De inhoud van de klassen die in de afbeelding zijn terug te vinden, wordt ondergebracht in de aangemaakte groepen. Deze groepen zijn echter alleen gemaakt voor het clientScherm van de *gebruiker*. De inhoud van de overige klassen die hier niet zijn ondergebracht kunnen alleen door de *beheerder* worden aangepast.

4.4. Pilotontwikkelplan

Na het ontwerp van de functionele en technische structuur kon het ontwikkelplan geschreven gaan worden. Een pilotontwikkelplan beschrijft hoe de pilot gaat worden gebouwd. Het opstellen van dit plan heb ik uitgevoerd volgens de werkwijze die voorgeschreven staat in het boek van IAD door R.H.J. Tolido.
Het pilotontwikkelplan dat ik tijdens de definitiestudie heb ontwikkeld heb ik hier als uitgangspunt genomen. Voor iedere pilot uit dit plan heb ik de volgende pilotdelen gedefinieerd:

1. Opzetten ontwikkelomgeving en bepalen naamgeving
2. Ontwikkelen scherm voor inloggen
3. Ontwikkelen van het hoofdscherm
4. Ontwikkelen van het clientscherm

Deze pilotdelen zijn vervolgens weer opgedeeld in bouweenheden.
Naast het definiëren van de pilotdelen en de bouweenheden heb ik ook de ontwikkelaanpak vastgelegd. Omdat ik de opdracht alleen moest uitvoeren was het bijvoorbeeld niet mogelijk om parallel te ontwikkelen. Vandaar dat de pilotdelen in chronologische volgorde worden ontwikkeld en ingevoerd.

4.5. Ontwerp software-bouweenheden

Na het vastleggen van het pilotontwikkelplan en het definiëren van de pilotdelen met bijbehorende bouweenheden, kon worden begonnen met het ontwerp van de bouweenheden.

4.5.1. Ontwerp bouweenheden

Het ontwerp van de software-bouweenheden was in dit stadium zoveel als mogelijk tekstueel gehouden. Daarbij was voor iedere bouweenheid globaal beschreven wat het resultaat moest zijn. Daarnaast was er een kort stappenplan opgesteld om tot het gewenste resultaat te komen. Een voorbeeld hiervan is de bouweenheid “Onderzoeken ontwikkelomgeving”.

> *Deze bouweenheid zal moeten leiden tot een geschikte ontwikkelomgeving.*
> *Om tot een geschikte omgeving te komen zullen de volgende stappen uitgevoerd worden.*
> *Allereerst zal de criteria van een geschikte omgeving worden bepaald. Aan de hand daarvan kunnen een aantal omgevingen worden geselecteerd. Op basis van deze vergelijking zal vervolgens een geschikte omgeving worden gekozen. De keuze zal hierna worden onderbouwd.*

Tijdens de bouw van deze bouweenheid zijn de resultaten verwerkt in het ontwerp van de bouweenheid.

4.5.2. Ontwerp testspecificaties

Om de gemaakte bouweenheden te testen zijn er testspecificaties opgesteld, zodat duidelijk wordt dat de werking van bouweenheden het verwachte resultaat geeft. Omdat de pilot hier een GUI betreft en praktisch alle bouweenheden gelijk aan elkaar zijn, is besloten om het opstellen van de testspecificaties enigszins anders aan te pakken dan in de andere pilots gedaan zal worden. Bij deze pilot draait het vooral om het gebruik van de GUI. Hoewel richtlijnen in gebruikersvriendelijkheid daar inzicht in kunnen geven, zijn het de verloskundigen die met deze applicatie zullen gaan werken. De testspecificaties zijn daarom opgesteld vanuit het oogpunt van de opdrachtgever en verdeeld in twee verschillende groepen:

- Look & Feel
  De look & feel testspecificaties zullen worden uitgevoerd door de opdrachtgever. Deze kan dan naar eigen inzicht bepalen of de ideeën die in de workshop naar voren zijn gekomen ook daadwerkelijk zijn terug te vinden in de GUI. Deze tests hebben vooral betrekking op de werking.

- Gebruikersvriendelijkheid
  Evenals de vorige tests worden deze uitgevoerd in overleg met de opdrachtgever. Het gaat er bij deze tests om of de applicatie gebruikersvriendelijk oogt en als zodanig wordt ervaren.

4.6. Bouw software-bouweenheden

Na de ontwikkeling van de software-bouweenheden kon worden begonnen met de bouw. Om de documentatie van de pilot volledig te houden is opgedane kennis tijdens de bouw teruggekoppeld naar het ontwerp van de pilot.

4.6.1. Onderzoeken ontwikkelomgeving

In de eerste bouweenheid heb ik onderzoek gedaan naar de meeste geschikte ontwikkelomgeving. In de opdrachtomschrijving was reeds vastgesteld dat Borland Delphi de gebruikte ontwikkelomgeving zou worden. Om in dit stadium te verzekeren dat die ontwikkelomgeving de juiste was, is een klein onderzoek gedaan naar de alternatieven.

De te kiezen ontwikkelomgeving moest de mogelijkheid bieden tot het ontwikkelen van een stand-alone applicatie. Hiermee wordt bedoeld dat deze niet afhankelijk is van een interpreter of een parser. Wanneer een applicatie wordt ontwikkeld met behulp van interpreters, bestaat de kans dat er beveiligingsfouten in de huidige versie zitten. Aangezien er geen ICT-deskundigen aanwezig zijn binnen de organisatie, kan niet worden verwacht dat de gebruikte programma’s om de zoveel tijd worden geüpdate.

De ontwikkelomgeving moet daarnaast ook de mogelijkheid bieden tot het creëren van een grafische user interface. Hoewel bijna iedere ontwikkelomgeving deze mogelijkheid

biedt, zijn er vaak restricties op de mogelijkheden hiervan. De vrijheid in het ontwikkelen van een GUI moest zo min als mogelijk restricties hebben.

Als laatste minimale eis, was het van belang dat de ontwikkelomgeving de mogelijkheid moest hebben tot het benaderen van een DBMS. Zonder deze mogelijkheid was communicatie met een database niet mogelijk binnen de randvoorwaarden van het project. Deze randvoorwaarden waren vooral het kostenplaatje en de tijdsduur.

Na het vaststellen van de eisen aan de ontwikkelomgeving kon worden gezocht naar geschikte kandidaten. De kandidaten zijn geselecteerd op kennis van de talen en door het uitproberen van een aantal diverse talen. Het resultaat was een vijftal geschikte ontwikkelomgevingen.

Programmeertaal
Delphi
C++
C
Visual Basic
Turbo Pascal

Afbeelding 11: Kandidaten ontwikkelomgevingen

Aangezien ik gekozen had voor de technieken van UML, was het ontwikkelen in een object georiënteerde omgeving noodzakelijk. Naast deze eis waren er nog vier andere eisen die de doorslag moesten geven voor de definitieve ontwikkelomgeving.

	Delphi	**C++**	**C**	**Visual Basic**	**Turbo Pascal**
Ondersteund Object georiënteerd programmeren	++	++	+	--	--
Kan zonder extra bibliotheekbestanden op iedere pc functioneren	+	++	++	-	++
Is bekend bij de projectgroep	++	--	++	++	+
Kan een Windows-GUI opleveren zonder extra bibliotheekbestanden	++	++	++	++	--
Heeft de mogelijk tot het gebruik ODBC	++	++	-	-	-

Afbeelding 12: Resultaat kandidaten

4.6.2. Bepalen programmeeromgevingen

Na het vaststellen van de ontwikkelomgeving, had ik ervoor gekozen om een bouweenheid te wijden aan het opzetten van de programmeeromgeving. De programmeeromgeving was het pakket dat gebruikt ging worden voor het werken met de

taal Delphi. Voor het programmeren in de taal Delphi is het programma Borland Delphi Enterprise Edition het meest geschikt. Deze maatschappij biedt als enige een gratis licentie voor studenten, met als voorwaarde dat er geen producten ontwikkeld werden met commerciële doeleinden.
Er waren verder geen problemen bij het opzetten van de programmeeromgevingen.

4.6.3. Bepalen naamgeving objecten

In Borland Delphi moet voor het gebruik van een object een instantie worden aangemaakt. Een instantie van een bepaald object moest in deze bouweenheid een vastgestelde naam krijgen toegewezen. Zo geeft Delphi iedere instantie van het type TButton de naam ‘button’ gevolgd door een nummer. Dit nummer is het aantal instanties van het object TButton in de applicatie.

Variabelen binnen de applicatie moesten een zinvollere naamgeving krijgen. Deze naamgeving zou een indruk geven van het doel van de variabele. Vandaar dat ik gekozen heb om structuur aan te brengen in de naamgeving. Deze structuur bestaat uit drie delen:

- Een instantie van een object uit Delphi begint met de naam van het object afgekort tot drie letters.
- Een zelfstandig naamwoord dat de inhoud beschrijft.
- Een afkorting of woord dat betrekking heeft op de naam van de knop

Een voorbeeld ter illustratie is de OK-knop bij het inlogscherm, deze is van het type TButton. Deze knop krijgt volgens deze structuur de naam ‘btnInloggenOk’.

4.6.4. Bepalen schermattributen inlogForm

Na het definiëren van de naamgeving voor de objecten in Delphi kon worden begonnen met het opstellen van de schermen oftewel ‘Forms’. Een ‘Form’ is een instantie van het object TForm. Dit object is de manier waarop in Delphi een scherm kan worden ontwikkeld. Vandaar dat de naam ‘Form’ ook gebruikt wordt als scherm.

Bij het opstellen van de schermattributen van het inlogscherm waren vrijwel geen problemen. De schermattributen die gedefinieerd moesten worden waren hoogte, breedte, kleur, titel, functionaliteit op de titelbalk en positie van de buttons (knoppen).
Om de veranderingen van de applicatie niet al te groot te maken, is ervoor gekozen om het inlogscherm qua lengte en breedte ongeveer hetzelfde te houden.

4.6.5. Ontwerp inlogForm

Net als dat het bepalen van de schermattributen vrij gemakkelijk en logisch was, was dat ook voor het ontwerp van de Form. De keuze om niet af te wijken van de huidige applicatie gaf de basis voor het inlogscherm. Hoewel dit het ontwerp vrij eenvoudig maakte, had ik toch besloten om door middel van prototyping de definitieve schermen te maken in overleg met de opdrachtgever.

Mijn besluit was om iedere form voor te leggen en aan de hand daarvan te overleggen of deze voldeden aan de eisen en op welk gebied deze eventueel verbeterd moesten worden.

Afbeelding 13 : Definitieve inlogscherm

Het eerste schermontwerp dat twee tekstvelden, twee labels en twee buttons had, was vrijwel direct goed.

Het verzoek was om zoveel mogelijk de engelse taal te vermijden, aangezien ik eerst de begrippen Username en password gebruikt had. Na deze kleine aanpassing was het scherm dat hier is opgenomen het definitieve resultaat.

4.6.6. Bepalen schermattributen hoofdForm

Na het ontwerp van het inlogscherm moesten de attributen voor het hoofdscherm worden bepaald. Voordat hiermee kon worden begonnen moest eerst worden geïnventariseerd welke functionaliteiten er in dit scherm verwerkt moesten worden en hoe de aanroep hiervan geïmplementeerd ging worden.

Om een indruk te krijgen van de wijze waarop de meeste functionaliteiten in grote applicaties worden ondergebracht, is een aantal applicaties van Microsoft en de huidige applicatie als uitgangspunt genomen. De applicatie Word XP maakt gebruik van iconen en een menubar om de functies aan te roepen. De iconen in deze applicatie representeren allemaal een logisch resultaat. Zo is het maken van een nieuw document een icoon met een leeg papiertje.

Omdat de functies van de verloskundeapplicatie niet eenduidig waren en niet makkelijk in een symbool zijn samen te vatten, is er voor gekozen om die manier van navigeren niet toe te passen. De menubar zal worden gebruikt om te kunnen navigeren binnen de applicatie. Voor de bevestigingen en het annuleren zullen knoppen worden gebruikt.

Om de schermen onderling niet teveel te laten verschillen is besloten om dit scherm de maximale breedte en hoogte toe te kennen, namelijk 1024 bij 768 pixels.

4.6.7. Ontwerp hoofdForm

Na het bepalen van de schermattributen kon worden overgegaan tot het ontwerp van het hoofdscherm. Dit scherm is net als bij het inlogscherm door middel van prototyping opgesteld. De moeilijkheid bij het ontwerp van dit scherm was dat er geen concreet voorbeeld was.

Allereerst is er een leeg scherm ontworpen met daarin de menubar. De titel van de applicatie en de overige schermattributen zijn vervolgens vastgelegd. Dit scherm moest vervolgens de use-cases die waren ontworpen in de definitiefase voor gebruiker en beheerder hier aanbieden als functie.

Deze use-cases zijn in de definitiefase onderverdeeld in subsystemen. Er is besloten om deze indeling te gebruiken als richtlijn in de menubar. Dit heeft geleid tot de volgende menu-functies:

- Cliënten
- Rapportage
- Beheer

Het subsysteem ‘zwangerschap’ kan niet worden benaderd zonder een cliënt te hebben geselecteerd of aangemaakt. Vandaar dat deze functies zijn ondergebracht in dat scherm in plaats van direct op het hoofdscherm.

Na het doorlopen van al deze stappen was het taak om het ontwerp van het scherm te laten controleren door de opdrachtgever. De opdrachtgever vond dat het scherm nogal leeg was. De suggestie hiervoor was om extra knoppen op te nemen, waarmee de belangrijke functies van de applicatie worden geactiveerd.

Vandaar dat besloten is om naast de menubar zijn een aantal knoppen op te nemen voor functies die een verloskundige vaak gebruikt, zoals nieuwe cliënt en zoeken. Deze knoppen zullen moeten bijdragen aan de tijdslimiet waarmee de verloskundigen een cliënt geregistreerd moeten hebben tijdens het spreekuur en de invulling van de applicatie moeten vergroten.

Het resultaat van het zoeken zal in een lijst worden weergegeven. Hier zal na het selecteren van een cliënt een knop worden geactiveerd waarmee direct het cliëntscherm kan worden benaderd.

Na deze aanpassingen was de opdrachtgever tevreden met het resultaat en kon worden begonnen met de volgende bouweenheid.

4.6.8. Bepalen schermattributen clientForm

Na het ontwerpen van het hoofdscherm, was het bepalen van de schermattributen van het cliëntscherm de volgende bouweenheid. Het cliëntscherm was op het gebied van inhoud het grootste scherm. Alle groepen, gedefinieerd in de technische structuur moesten in dit scherm worden ondergebracht. Bovendien moesten daarnaast de mogelijkheden tot het invullen van een aantal attributen worden bepaald.

In deze bouweenheid moesten echter alleen de schermattributen worden ontworpen en die waren bijna exact hetzelfde als die in de vorige bouweenheid. In deze bouweenheid waren daarom verder ook geen problemen.

4.6.9. Ontwerp clientForm

Na het bepalen van de schermattributen moest de volgende bouweenheid “Ontwerp clientForm” worden gerealiseerd. Het scherm voor de cliënten is een scherm dat hergebruikt kan worden, omdat het mogelijk moet zijn om meerdere cliënt tegelijkertijd te kunnen bewerken. Hierdoor was het vooral van belang dat in ieder scherm duidelijk aangegeven werd welke cliënt in het scherm geselecteerd was. Bovendien moest daarnaast ook het zwangerschapsnummer en de bloedgroep worden aangegeven.

Om dat te kunnen realiseren was in overleg met de opdrachtgever een oplossing bedacht om een extra statusbar onderin het scherm aan te brengen die deze content weergaf. De inhoud van de groepen was van dusdanige omvang dat deze nooit op één scherm paste. Vandaar dat er een extra navigatie moest worden ontworpen voor het cliëntscherm. Om de navigatie binnen de applicatie waar mogelijk consistent te houden, was besloten om wederom een menubar te maken.

Het clientscherm bevatte de twee subsystemen, “Contactmoment cliënt” en “Zwangerschap”. In de navigatie is er onderscheid gemaakt tussen deze twee. Dit leverde een leeg cliëntscherm op met navigatiemogelijkheden. Aangezien er al een eerder overleg was geweest over de inhoud van het scherm, zou extra overleg pas nuttig zijn zodra de groepen waren ontworpen.

4.6.10. Ontwerpen groepen basisstructuur

Nadat de opzet van het scherm was ontwikkeld en gebouwd kon worden begonnen met de groepen die in de basisstructuur waren gedefinieerd. Iedere groep in de basisstructuur krijgt een unieke kleur en de indeling van de attributen wordt bepaald door de volgorde waarop dit in de huidige applicatie ook is gerealiseerd. Het resultaat van de indeling die in de basisstructuur naar voren was gekomen, is in deze bouweenheid gedetailleerd.

Allereerst moesten de kleuren worden gedefinieerd per groep. Ik heb voor een webdesign bedrijf een cursus kleurencontrasten gevolgd. In dit project was het onderdeel gebruikersvriendelijkheid met betrekking tot kleurencontrasten goed toepasbaar. Hierin wordt de nadruk gelegd op het gebruik van verschillende kleuren voor het onderscheiden van functionaliteiten. Er moest voor iedere functionaliteit een duidelijke kleurverschil zijn. Om het voor de verloskundigen aantrekkelijk te maken, heb ik per groep drie verschillende kleuren opgesteld. Daaruit kon vervolgens een lijst met kleuren worden gekozen die gehanteerd zouden worden per groep. Het definitieve resultaat hiervan is hieronder opgenomen en zo ook terug te vinden in het document van de eerste pilotontwikkelfase.

Groepsnaam	Kleurcode
grpAnamnese	clSkyBlue
grpPartus	clLime
grpClient	clInactiveCaptionText
grpContact	clTeal
grpEchoLab	$00FF008A (Purple)
grpKindKraambed	clOlive
grpLosKraambed	clButtonFace
grpPrenataleZorg	clWhite
grpVerwijzingen	clCream

Afbeelding 14: Groepen met bijbehorende kleurcodes

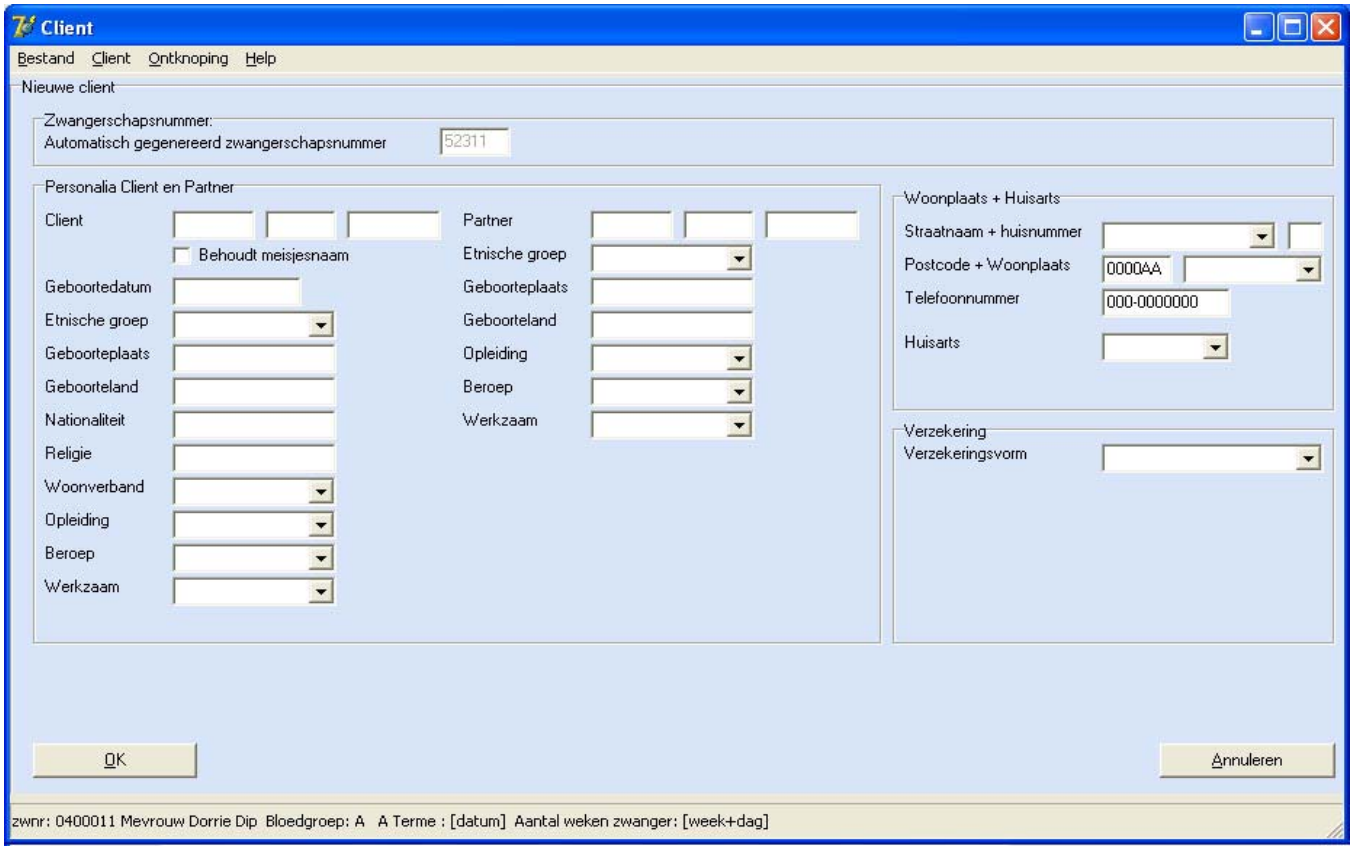
In bovenstaande afbeelding zijn de kleurcodes opgenomen als codes die in Delphi een kleur representeren. Voor de paarse kleur is geen gedefinieerde code in Delphi vandaar dat het hexadecimale getal '$00FF008' gebruikt wordt.

Nadat de kleuren gedefinieerd waren, moesten de groepen worden gevuld met tekstvelden, labels en andere invulvelden. Er was besloten om voor ieder attribuut een label op te nemen met een soortgelijke / aanvullende naamgeving en daarbij een tekstveld waarin de attributen van de klassen konden worden aangepast.

Voor de duidelijkheid is dit toegelicht aan de hand van een voorbeeld van de groep Cliënt. Bij deze groep zijn - eerder in deze pilot gedefinieerd - de klassen cliënt, partner en verzekeraar ingedeeld. De indeling op het scherm is op basis van eigen inzicht gedaan.

Vervolgens heeft de groep de kleur toegekend gekregen en zijn de labels en tekstvelden en overige invoervelden opgesteld aan de hand van de attributen van de klassen. Er is voor gekozen om statische velden als lijst op te nemen waaruit één selectie mogelijk is. Een voorbeeld van zo'n statisch veld is woonverband.

Dit heeft geleid tot het scherm opgenomen in onderstaande afbeelding.



Afbeelding 15: GUI groep Cliënt

4.7. Invoeringsprocedure

Na het bouwen van de bouweenheden moest er een invoeringsprocedure worden opgesteld. Het opstellen van een invoeringsprocedure heeft tot doel om het operationeel maken van de pilot succesvol te laten verlopen. De invoeringsprocedure voor deze pilot had als enige uitgangspunt de controle op de aanwezige hard- en software. Daarvoor zijn er volgens het “black-box”-principe een drietal testcases opgesteld.
Volgens dit principe wordt een testcase opgesteld en wordt aan de hand van de gegeven invoer de verwachte uitvoer bepaald. Indien de verwachte uitvoer niet voldoet aan de verkregen uitvoer, dan heeft dat tot resultaat dat de test is mislukt.

Testspecificatie Invoeringsplan		
Test	**Invoer**	**Verwachte uitvoer**
Bibliotheekbestanden zijn aanwezig	Controleren of de bibliotheekbestanden aanwezig zijn op het werkstation	De bestanden zijn aanwezig
Voldoende harde schijf ruimte	Controleren of het werkstation voldoende ruimte heeft	Het werkstation heeft voldoende ruimte

MySQL is aanwezig	Controleren of het DBMS MySQL reeds geïnstalleerd is.	MySQL is aanwezig en functioneert naar behoren

4.8. Testen en beoordelen pilot

In de activiteit “Testen en beoordelen” heb ik vervolgens de uitgevoerde pilot beoordeeld. Allereerst zijn de testresultaten van de bouweenheden en de invoeringsprocedure worden uitgewerkt en opgenomen.

Er waren drie cases die een onverwacht resultaat hadden. Deze cases zijn vervolgens toegelicht en is er een oplossing voor bedacht.

Mislukte Case # 1.3
De naamgeving voor de labels is aangepast om een concretere beschrijving te geven van de ingevulde waarde. Dit heeft er toe geleid dat sommige groepen opnieuw moesten worden ingedeeld.

Mislukte Case # 1.4
De kleur van de groep “Anamnese” was op een beeldscherm met lage resolutie onleesbaar geworden. De oplossing hiervoor was de kleur aanpassen tot de kleur clSkyBlue.

Mislukte Case # 3.2
Er was reeds een versie van MySQL aanwezig op het werkstation, echter was dit een verouderde versie en ondersteunde het opslagtype InnoDB niet. Daarom is besloten om een gratis versie te installeren die deze mogelijkheid wel ondersteund.

Afbeelding 16: Mislukte cases uit pilot één

4.9. Afronding

Aangezien er voor iedere mislukte case een oplossing was bedacht, was de conclusie dat de pilot naar behoren was uitgevoerd. De ontwikkeling van de schermen had echter wel meer tijd in beslag genomen dan bij het opstellen van de planning was aangenomen. Vooral de overzichtelijkheid in de indeling van de labels en tekstvelden was tijdrovend. De uitvoering van de definitiefase had echter ook al zes dagen extra in beslag genomen, zodat de geplande uitloop van twee weken nog maar een dag bedroeg.

5. Pilot 2: Database

Na de beoordeling en afronding van de vorige pilot is er begonnen met de ontwikkeling van de tweede pilot. Tijdens de definitiefase is het ontwerpen en implementeren van de database als tweede pilot gedefinieerd. Deze pilot beschrijft de activiteiten die in het kader van deze ontwikkeling zijn uitgevoerd.
De opbouw van dit hoofdstuk verschilt met die van het vorige hoofdstuk. In de eerste paragraaf wordt de totstandkoming van het plan van aanpak voor de ontwikkeling van de pilot beschreven. Na deze paragraaf worden in paragraaf 5.2 de activiteiten die zijn uitgevoerd in het kader van het pilotontwikkelplan beschreven. Vervolgens worden het ontwerp en de bouw van de software-bouweenheden in respectievelijk paragraaf 5.3 en 5.4 toegelicht. Het ontstaan van de invoeringsprocedure is in paragraaf 5.5 toegelicht. Met het testen en beoordelen van de pilot in paragraaf 5.6 en de afronding 5.7 wordt dit hoofdstuk afgesloten.

5.1. Plan van aanpak

Net als in de vorige pilot heb ik allereerst een plan van aanpak opgesteld. In dit plan van aanpak zijn de activiteiten die in het kader van deze pilot zijn uitgevoerd opgenomen. In dit plan van aanpak is wederom een afweging gemaakt welke activiteiten uitgevoerd worden. De vervallen activiteiten zijn vervolgens toegelicht. In deze pilot waren dat er meer.

De meest belangrijke activiteiten die in deze pilot in mijn ogen vervallen zijn, zijn de technische en functionele structuur. Er was in deze pilot geen sprake van functies die in een functionele structuur globaal zouden moeten worden toegelicht. Daarnaast zou de technische structuur in de vorm van een klassendiagram kunnen worden ontworpen. Het besluit was om het klassendiagram in de definitiestudie te hanteren als basis voor het ontwerp van de database.

Na het toelichten van de vervallen activiteiten, heb ik op basis van de overgebleven activiteiten een schatting gemaakt van de tijdsduur. Deze activiteiten heb ik vervolgens verwerkt in een planning.

5.2. Pilotontwikkelplan

Na het opstellen van het plan van aanpak is de pilot verdeeld in pilotdelen. Door de geringe omvang van de pilot was ervoor gekozen om de pilot te verdelen in twee delen. Die weer onderverdeeld zijn in de volgende bouweenheden:

1. Opzetten databaseomgeving
2. Ontwerpen database
3. Vertalen van klassen naar tabellen

De bouweenheid in het opzetten van de omgeving was in de vorige pilot ook al gedefinieerd en dit bleek goed te werken, zodat deze ook in deze pilot is gedefinieerd.

Het opzetten van de daaropvolgende bouweenheden vormden een obstakel. Na de eerste pilot is gebleken dat het benoemen van de bouweenheden niet mijn sterkste kant is. Bij het benoemen van de bouweenheden ontstond hetzelfde probleem. In overleg met mijn begeleider had ik besloten dat de naamgeving niet heel erg van belang was, zolang er geen twijfel bestaat over de inhoud van de bouweenheid.

Daarop is gekozen om als tweede bouweenheid het ontwerpen van de database in MySQL te definiëren. De attributen in het klassendiagram hadden nog geen type toegewezen gekregen en dat kon in deze bouweenheid worden uitgewerkt. De SQL-statements voor de implementatie zijn in de derde bouweenheid gedefinieerd.

5.3. Ontwerp software-bouweenheden

Na het definiëren van de bouweenheden was de volgende activiteit het ontwerp van de bouweenheden. Het ontwerp moest dusdanig worden opgezet dat op basis hiervan de implementatie kon worden gerealiseerd.

5.3.1. Ontwerp bouweenheden

Bij het ontwerp van de bouweenheden is net als bij de eerste pilot allereerst een doel en stappenplan opgesteld. Op basis hiervan is de ontwikkeling begonnen en is het ontwerp aan de hand van opgedane kennis tijdens de bouw gewijzigd en aangevuld.
Om het document van de pilotontwikkeling een professioneel karakter te geven is besloten om het stappenplan na de bouw te verwijderen. De stappen die in het stappenplan gevolgd zijn, zouden moeten blijken uit de resultaten die in het ontwerp zijn verwerkt.

Ook bij deze pilot heb ik gezocht naar diagrammen die het ontwerp van de bouweenheden kracht bij konden zetten. Maar naast het klassendiagram zijn er geen geschikte modellen binnen UML die aanvulling gaven op de ontworpen bouweenheden.

Bij het ontwerp van de bouweenheden hebben zich vrijwel geen problemen voorgedaan. Het behouden van de referentiele integriteit van de database vormde echter wel een obstakel. Tijdens de fase van het opzetten van de databaseomgeving was de versie van MySQL die het opslagtype InnoDB ondersteunde, als een bètasoftwarepakket beschikbaar. Dit betekende dat er eventuele fouten in de software zouden kunnen zitten waar bij niet gegarandeerd kon worden dat de tabellen en bijbehorende data gewaarborgd zouden blijven. Het ontbreken van die garantie betekende dat de referentiele integriteit alleen gewaarborgd kon worden door dit softwarematig op te lossen.

In een later stadium van het project was er een stabiele variant uitgebracht van MySQL die het behoud van gegevens wel degelijk garandeerde. Om de applicatie te ontlasten van deze controle, was besloten om de database aan te passen met het type InnoDB.

5.3.2. Opstellen testspecificaties

Na het ontwerp van de bouweenheden heb ik de testspecificaties opgesteld. Deze testspecificaties zijn net als in de eerste pilot niet per bouweenheid gedefinieerd. De reden hiervoor was dat deze bouweenheden elkaar overlappen. Om die reden is ervoor gekozen om de testspecificaties te verdelen in twee groepen. Bovendien is het klassendiagram uit de definitiefase in deze pilot ook gebruikt. Vandaar dat er een groep is gedefinieerd waarin testcases beschreven zijn die de consistentie tussen de voorgaande diagrammen en de database controleren.

De eerste groep beschrijft drie cases die in het kader van de consistentie worden uitgevoerd.

Testspecificatie Voorgaande diagrammen	
Case #	**Test**
1.0	Klassen in klassendiagrammen zijn gelijk aan de tabellen in de database
1.1	Klassenrelaties komen overeen met de database
1.2	Kardinalen in klassenrelaties zijn consistent met de plaatsing van de vreemde sleutels in de tabellen in de database

De tweede groep beschrijft de testcases waarmee de database wordt gecontroleerd op volledigheid en de correct werking.

Testspecificatie Database	
Case #	**Test**
2.0	De vreemde sleutels in de tabellen hebben allemaal een index
2.1	De vreemde sleutels in de tabellen zijn allemaal gedefinieerd als ‘FOREIGN KEY’ met de juiste constraints.
2.2	Iedere tabel heeft hetzelfde type; ‘InnoDB’.
2.3	De constraints (ON Update en On Delete) geven het gewenste resultaat bij iedere tabel.
2.4	Er komen geen dubbele referenties voor in database, zoals werknemer -> contactmoment -> werknemer.
2.5	De gebruiker(s) voor MySQL kan inloggen met het opgegeven wachtwoord.
2.6	De gebruiker(s) voor MySQL heeft alleen beschikking over de toegewezen rechten.

De reden om de cases globaal te definiëren is dat in de eerste vier gevallen er voor iedere tabel een specificatie zou moeten worden opgenomen. Dit zou resulteren in ongeveer een 80-tal extra cases. Dit maakt het document slordig en bied geen toegevoegde waarde. Indien een case mislukt bij een bepaalde tabel zal dit worden opgenomen.

De resultaten van deze verzameling testspecificaties zijn in het testen en beoordelen van de pilot uitgewerkt en waar nodig toegelicht.

5.4. Bouw software-bouweenheden

Na het ontwerp van de software-bouweenheden konden de bouweenheden worden gebouwd. Deze paragraaf beschrijft de bouw van de bouweenheden.

5.4.1. Opzetten databaseomgeving

Het opzetten van de databaseomgeving kon worden verdeeld in drie verschillende delen:

- Het aanmaken van een database in MySQL.
- Ontwerpen van de gebruikers van de database met bijbehorende wachtwoorden
- Invoeren van de gebruikers in de database.

Allereerst moest er een database aangemaakt worden in MySQL waarin de tabellen konden worden geplaatst die in de volgende bouweenheid zouden worden ontwikkeld. De naam van de database was niet relevant, toch is er gekozen voor dezelfde naamgeving als die van de applicatie. Vervolgens is de database ‘W!se Delivery’ aangemaakt in MySQL.

Vervolgens moesten de gebruikers van de database worden aangemaakt. Deze gebruikers zijn niet van hetzelfde type als in het use-case diagram. Hoewel er wel een normale gebruiker en een beheerder worden aangesteld in de database, is dit uit veiligheidsoverwegingen, zodat de database niet ongevraagd gewist kan worden.
Om onderscheid te maken tussen de twee verschillende gebruikers wordt er in MySQL gebruik gemaakt van ’privileges’. Dit zijn rechten die een gebruiker krijgt toegewezen door het DBMS en daardoor in staat wordt gesteld om operaties uit te voeren op de database.
De ‘normale’ gebruiker krijgt alleen de rechten die nodig zijn om data in de tabellen te lezen, te wijzigen, te verwijderen en toe te voegen.

Beide gebruikers krijgen vervolgens een inlognaam en wachtwoord toegekend. Dit wachtwoord is gegenereerd door een “random password generator”, zodat deze voor een mens geen logische combinatie van letters of cijfers is. Ter illustratie is de sql-query opgenomen die de ‘normale’ gebruiker implementeert.

```
Info    INSERT INTO `user` ( `Host` , `User` , `Password` ,
        `Select_priv` , `Insert_priv` , `Update_priv` , `Delete_priv`
        , `Create_priv` , `Drop_priv` , `Reload_priv` ,
        `Shutdown_priv` , `Process_priv` , `File_priv` , `Grant_priv`
        , `References_priv` , `Index_priv` , `Alter_priv` ,
        `Show_db_priv` , `Super_priv` , `Create_tmp_table_priv` ,
        `Lock_tables_priv` , `Execute_priv` , `Repl_slave_priv` ,
        `Repl_cliënt_priv` , `ssl_type` , `ssl_cipher` , `x509_issuer`
        , `x509_subject` , `max_questions` , `max_updates` ,
        `max_connections` )
        VALUES (
        'localhost', 'Info', PASSWORD ( 'Tt5jal21a' ) , 'Y', 'Y', 'Y',
        'Y', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N',
        'N', 'N', 'N', 'N', 'N', 'N', '', '', '', '', '0', '0', '0');
```

Afbeelding 17: SQL-statement voor het invoegen van de gebruiker ‘Info’.

5.4.2. Ontwerpen database

Na het opzetten van de databaseomgeving kon worden begonnen met het ontwerpen van de standaarden voor de database. In deze bouweenheid stond het ontwerpen van de typen voor de tabellen centraal. Daarnaast was de keuze van het opslagtype een aandachtspunt in deze bouweenheid.

Met het ontwerpen van de typen wordt hier verstaan dat er afspraken zijn vastgelegd over de typen die de attributen voor de tabellen moeten gaan hebben. Een voorbeeld van zo’n afspraak hiervan is dat het standaard MySQL-type voor datum en tijd niet gebruikt zal gaan worden in het ontwerp van de tabellen. De reden hiervan is dat het standaardtype de Amerikaanse notatie gebruikt, terwijl de Europese notatie gewenst is.

In deze bouweenheid zijn verder weinig problemen opgetreden.

5.4.3. Vertalen van klassen naar tabellen

Na het ontwerpen van de standaarden voor de database kon worden begonnen met het vertalen van het klassendiagram naar een relationeel implementatiemodel. Omdat het vertalen van een klassendiagram naar een implementatiemodel een tijdrovend maar eenvoudige activiteit is, waren er weinig problemen.

In de definitiefase was ik in de veronderstelling dat het klassendiagram 1-op-1 vertaald kon worden. Rekening houdend met eventuele toekomstige uitbreidingen van de applicatie en in het kader van overzichtelijkheid is besloten om enkele tabellen op te splitsen in meerdere tabellen.

Bij de klasse *obstetrische anamnese* kan er sprake zijn geweest van een negatieve en positieve uitkomst van de zwangerschap. In beide gevallen dienen er extra gegevens te worden geregistreerd. In de tabel *obstetrisch* zou kunnen worden volstaan met een veld waarin wordt aangegeven of de bevalling negatief of positief is. Besloten is om twee tabellen te ontwerpen voor deze attributen, waardoor de tabellen ‘obs_abortus’ en

'obs_zwangerschap' zijn ontwikkeld. Deze beslissing is genomen om de tabellen overzichtelijk te houden en om eventuele uitbreiding van de applicatie mogelijk te maken. Mocht het zo zijn dat vrouwen met meerlingen zich mogen aanmelden bij de praktijk dan maakt deze oplossing dat mogelijk.

Na het omzetten van de klassen naar de tabellen, moesten de vreemde sleutels en relaties worden aangebracht. Hoewel de inhoud van de tabellen in vrijwel geen enkel geval verwijderd mag worden, is de "ON DELETE"-restrictie wel opgenomen.

Ter illustratie is hier het implementatiemodel van de tabel werknemer, van de gelijknamige klasse opgenomen.

```
CREATE TABLE `werknemer` (
  `nummer` int(11) NOT NULL default '0',
  `voornaam` varchar(150) default NULL,
  `tussenvoegsel` varchar(150) default NULL,
  `achternaam` varchar(150) default NULL,
  `straatnaam` varchar(150) default NULL,
  `huisnummer` varchar(50) default NULL,
  `postcode` varchar(6) default NULL,
  `woonplaats` varchar(150) default NULL,
  `telefoonnummer` varchar(11) default NULL,
  `geboortedatum` varchar(10) default NULL,
  `wachtwoord` varchar(50) default NULL,
  `praktijk_nummer` int(11) NOT NULL,
  PRIMARY KEY  (`nummer`),
  INDEX (praktijk_nummer),
  FOREIGN KEY(praktijk_nummer) REFERENCES praktijk(nummer) ON UPDATE
CASCADE ON DELETE RESTRICT
) TYPE=InnoDB;
```

Afbeelding 18: Implementatiemodel van de tabel werknemer

5.5. Invoeringsprocedure

Na de bouw van de laatste bouweenheid moest er een invoeringsprocedure worden geschreven. De invoeringsprocedure heeft als doel het operationeel maken van de pilot succesvol te laten verlopen.

Allereerst is de procedure geschreven die de invoering van het implementatiemodel beschrijft. Deze procedure is apart geschreven omdat de invoering van de tabellen op een bepaalde volgorde moesten worden geïmplementeerd. Deze volgorde wordt bepaald door de aanwezigheid van vreemde sleutels in de tabellen en de index hierop. Het DBMS MySQL staat niet toe dat de tabellen met een index op een nog niet aanwezige tabel worden ingevoerd.

Om die reden zijn de tabellen verdeeld in drie groepen, waarbij de eerste groep geen vreemde sleutels heeft, de tweede groep een vreemde sleutel heeft naar een van de tabellen uit de eerste groep. De laatste groep zijn de overgebleven tabellen, waarop de restrictie van het DBMS niet meer van toepassing is, omdat alle vreemde sleutels naar een tabel verwijzen uit de eerdere groepen.

Die groepen zijn ingedeeld door eerst een lijst te maken van de onderlinge relaties met betrekking tot de vreemde sleutels. Op basis van die lijst zijn de letters A,B en C toegekend aan respectievelijk groep één, twee en drie.

De invoeringsprocedure voor de eerste bouweenheid bestaat uit twee verschillende delen. Het eerste deel bestaat uit de gebruikers aanbrengen in de MySQL gebruikersdatabase. Het tweede gedeelte is het instellen van de aanwezige ODBC-software. Het eerste gedeelte van deze procedure was vanzelfsprekend, de opdeling was echter van belang, omdat de instelling in ODBC niet getest kunnen worden als de gebruikers niet zijn ingevoerd.
Ondanks dat ik nog niet eerder met een dergelijk wijze van communicatie had gewerkt, was het zoeken op internet naar de precieze mogelijkheden en beperkingen ervan een onderdeel van de procedure. Het bleek echter zeer makkelijk te begrijpen en hierdoor was dit gedeelte zonder verdere problemen opgezet.

5.6. Testen en beoordelen

Het testen en beoordelen van de pilot bestond net als in de vorige pilot uit de beschouwing van de resultaten van de uitgevoerde testcases. Uit de testcases zijn de mislukte cases toegelicht. Op basis van de mislukte en de gelukte cases is tot een oordeel gekomen over de tweede pilot in het pilotontwikkeltraject.

De eerste set testcases specificeert de resultaten de controle op de correcte samenhang tussen het eerder opgestelde klassendiagram en het implementatiemodel.

Case #	**Test**	**Resultaat**
1.0	Klassen in klassendiagrammen zijn gelijk aan de tabellen in de database	Mislukt
1.1	Klassenrelaties komen overeen met de database	Mislukt
1.2	Kardinalen in klassenrelaties zijn consistent met de plaatsing van de vreemde sleutels in de tabellen in de database	OK

Mislukte case #1.0
De klassen die in het klassendiagram waren aangebracht, kwamen niet overeen met de tabellen in de database. De reden hiervoor was, dat klassendiagram tijdens het ontwerp van de tweede pilot gewijzigd was. Dit was echter niet doorgevoerd in de ontwikkelingen van deze pilot.
De oplossing hiervoor was om te vergelijken waar de verschillen zaten en deze in de database aan te brengen.

Mislukte case #1.1
Deze mislukte testcase was een direct gevolg op de eerste. Doordat de klassen niet overeenkwamen met de tabellen, waren de relaties tussen de klassen niet overeenstemming met de database.
Na het aanpassen van het klassendiagram zijn de testcases #1.0 en #1.1 nogmaals uitgevoerd en ditmaal met succes.

Na de toelichting op de mislukte cases en het verwerken van de oplossing zijn de testcases voor het implementatiemodel uitgevoerd. Bij het uitvoeren van deze testcases was het resultaat positief.
Na de verwerking van die resultaten zijn tot slot de testcases voor de invoeringsprocedure uitgevoerd. De resultaten van die testsets waren tevens succesvol.

5.7. Afronding

Na het schrijven van de conclusie voor het document en de controle van de rapportopmaak, kon het document worden gemarkeerd als definitief. Het resultaat is als bijlage opgenomen in dit verslag.

Door de omvang van het implementatiemodel was de geplande tijd voor de pilot niet gehaald. Bouweenheid 3 nam drie dagen meer tijd in beslag dan vooraf ingeschat. Ook de invoeringsprocedure bleek van grotere omvang dan aangenomen. Na het opstellen van de tweede pilot was er al een uitloop van vijf dagen. Omdat in het plan van aanpak een uitloop was opgenomen, waren er geen maatregelen nodig om deze vijf dagen in te halen.

6. Pilot 3: GUI – Database Interface

Na de beoordeling van de pilot en de aanpassingen aan de planning kon met de ontwikkeling van de derde pilot in het ontwikkeltraject worden begonnen. Deze pilot zoals in de definitiefase is gedefinieerd, realiseert de interface tussen de database en de grafische user interface. In dit hoofdstuk wordt beschreven welke activiteiten in het kader van deze pilot zijn uitgevoerd.
In de eerste paragraaf worden de activiteiten en beslissingen in het kader van het plan van aanpak opgenomen. De functionele en technische structuur zijn in de paragrafen 6.2 en 6.3 beschreven. Op basis van die structuur is in paragraaf 6.4 een pilotontwikkelplan opgesteld waarin de pilotdelen en bouweenheden beschreven worden. Het ontwerp en de bouw worden in paragraaf 6.5 en 6.6 besproken. Gevolgd door de invoeringsprocedure van de bouweenheden in paragraaf 6.7. Tot slot wordt het testen van de pilot en de afronding besproken in respectievelijk paragraaf 6.8 en 6.9

6.1. Plan van aanpak

Net als bij de vorige pilots is ook voor deze pilot een plan van aanpak geschreven. In dit plan van aanpak is op basis van de vorige pilots en de lijst met activiteiten die voorgeschreven worden door IAD besloten welke activiteiten in het kader van deze pilot zouden worden uitgevoerd. De vervallen activiteiten zijn vervolgens toegelicht met de reden waarom deze niet zijn uitgevoerd.
Voor iedere uit te voeren activiteit is een schatting gemaakt van de tijd die deze activiteit in beslag zou gaan nemen. Op basis van die schattingen is een gedetailleerde planning opgesteld.

6.2. Functionele structuur

Voor de aanvang van deze pilot speelden veel onbekende factoren mee. De wijze waarop de communicatie moest worden gerealiseerd en hoe dit naar Delphi vertaald moest worden, waren hier de basis voor. Om die reden is besloten om in de functionele structuur hieraan inhoud te geven. Allereerst moest de manier van communiceren worden vastgelegd en vervolgens de wijze waarop Delphi de communicatie met MySQL kan realiseren.

6.2.1. Communicatiemogelijkheden

Bij het ontwerpen van de functionele inhoud van de pilot is allereerst een kort onderzoek gedaan naar de alternatieven voor het ontwerpen van de interface. Als eerste zijn de stappen ontworpen die voor de communicatie vanaf de applicatie tot het DBMS moeten worden uitgevoerd. Dit heeft geleid tot de vier volgende stappen.

Stap	
1	De applicatie roept een functie aan
2	Deze functie vertaalt de aanroep naar een SQL-query
3	De SQL-query wordt uitgevoerd en het resultaat wordt teruggegeven aan de functie
4	De functie retourneert het resultaat.

Afbeelding 19: Communicatiestappen interface

De interface zal vervolgens aan de tweede stap en de derde stap invulling geven. Vervolgens zijn alle mogelijke alternatieven op die stappen in kaart gebracht. Die criteria zijn ontstaan op basis van het boek van Ghezzi en programmeerervaring.

I. Is deze wijze van implementatie OO (Object Oriented).
II. Zorgt deze wijze van implementatie voor een overdraagbare applicatie.
III. In hoeverre is de applicatie – gezien de wijze van implementatie - uitbreidbaar.
IV. Is deze wijze van implementatie correct. (correct; gezien het gebruik van UML en Delphi).

Afbeelding 20: Criterialijst van de alternatieven

De alternatieven zijn in een kruistabel afgezet tegen de criteria. Aan de hand van het bepalen in welke mate ieder alternatief voldeed aan de gestelde criteria, kon de beste manier van implementeren voor beide stappen worden bepaald. Deze uitkomst is vervolgens verwerkt in de tweede en derde stap. Dit heeft geleid tot de volgende definitieve communicatiestappen.

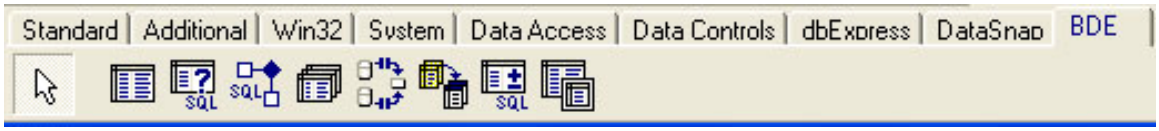
Stap	
1	De applicatie roept een functie aan
2	De functie die de aanroep vertaald naar een SQL-query is eigendom van een klasse
3	De klasse roept een functie van de databaseklasse aan en deze voert de query uit en geeft het resultaat terug aan de eigenaar. De klasse zal op diens beurt de benodigde operaties uitvoeren en het bewerkte resultaat teruggeven aan de applicatie.
4	De functie retourneert het resultaat.

Afbeelding 21: Definitieve communicatiestappen

Na dit korte onderzoek is de wijze van communicatie door middel van een voorbeeld in een sequencediagram opgesteld. Dit sequencediagram geeft de bovenstaande stappen in een diagram weer.

6.2.2. Delphi en MySQL

Aangezien ik geen eerdere ervaringen had met het gebruik van MySQL in een Delphi applicatie is allereerst onderzocht op welke manier Delphi dit ondersteunt. Uit het gekozen alternatief van communiceren is gebleken dat een aparte databaseklasse dit zal gaan realiseren.
Om die reden zullen de componenten die de verbinding met de database verzorgen worden ondergebracht als attribuut in de klasse. Het doorlezen van een aantal handleidingen en 'tutorials' voor het programmeren in Delphi met MySQL bracht snel uitkomst. De klasse van Delphi die ondersteuning bood aan het gebruik van de communicatie met een database middels ODBC was 'DBTables' in combinatie met BDE-componenten.



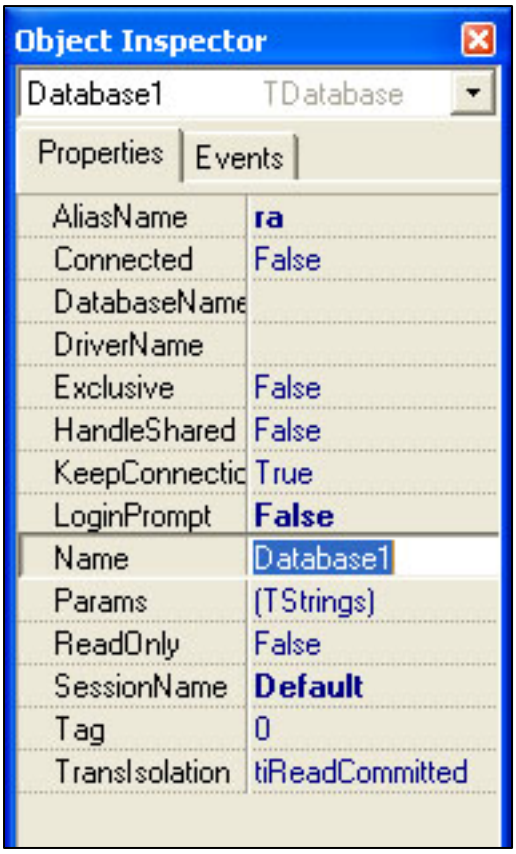
Afbeelding 22: BDE-componenten

Uit deze lijst met componenten zijn echter maar vier componenten die in de databaseklasse gebruikt gaan worden. Deze lijst met componenten is ontstaan door het volgen van de online handleidingen.

- TDatabase
- TDatasource
- TQuery
- TTable

Afbeelding 23: De vier benodigde componenten

Bij ieder component is vervolgens onderzocht welke attributen benodigd zijn voor een succesvolle communicatie. Ter illustratie is het componenten TDatabase toegelicht, in de bijlage van de derde pilot zijn al deze componenten opgenomen.



Om met de instantie 'Database1' een connectie op te zetten met een MySQL-database zijn de volgende variabelen van belang.

- Connected
- AliasName / Databasename
- LoginPrompt
- Params

Zodra '*Connected*' op True staat wordt verbinding gezocht met de MySQL-database met behulp van de variabelen *AliasName* of *DatabaseName*.

AliasName is hier de naam die toegekend is aan de ODBC-connectie, in dit voorbeeld 'ra'.

DatabaseName is de werkelijke naam van de database, die alleen ingevuld hoeft te zijn, indien er geen sprake is van de *AliasName*.

Loginprompt zorgt ervoor dat er een loginscherm verschijnt zodra een verbinding met de database wordt gezocht.

Indien de *loginPrompt* op 'false' staat en er geen *AliasName* is opgegeven is een vereiste dat in *Params* (Parameters) de velden 'User name' en 'Password' zijn opgenomen met geldige waarden.

Hieruit blijkt dat het component TDatabase de verbinding met de database tot stand kan brengen, door middel van ODBC (AliasName).

6.2.3. Functiebeschrijvingen

Na de resultaten van het onderzoek naar componenten was vastgelegd, kon worden begonnen met het beschrijven van de functies die de interface moest gaan ondersteunen. In de definitiefase was reeds een opzet gemaakt van deze functies en procedures, maar uit de gekozen oplossing bleek dat enkele functies niet in de interface konden worden ondergebracht.
De functies zoals "Invoegen_gegevens", "Opvragen_gegevens" en "Inloggen" zijn in de interface komen te vervallen. Deze waren bedoeld om een globale indruk te geven van de interface en zullen worden verwerkt in de bijbehorende klassen, zoals ook in het gekozen alternatief is beschreven.
Daarnaast moesten er wel een aantal extra functionaliteiten ontwikkeld gaan worden. Voordat er een operatie uitgevoerd kon worden, moest er allereerst gecontroleerd worden

of er reeds een connectie met de database tot stand was gebracht. Indien dit niet het geval was, moest deze alsnog worden gemaakt. De functie ‘checkConnection’ en de procedure ‘connect’ zijn hiervan het gevolg.

Voor het gebruik van een instantie van een klasse dient deze te worden ondergebracht in het geheugen. In Delphi wordt dit gerealiseerd door de procedure ‘create’. Deze geheugenruimte kan vervolgens weer vrijgegeven worden door de procedure ‘Free’. Voor het gebruik van de vier componenten geldt hetzelfde principe. Om de initialisatie te groeperen tot één stap, zijn de procedure ‘Create’ en ‘Free’ van de databaseklasse aangepast om de operaties van de componenten hier in onder te brengen.

De functies en procedures van de klasse zijn vervolgens zonder problemen verder uitgewerkt in de vorm van pre- en postcondities.

6.3. Technische structuur

Na het functionele gedeelte van de pilot te hebben ontworpen, was het ontwerp van de technische structuur de volgende activiteit in het ontwikkeltraject van de pilot. De technische structuur had bij deze pilot vooral betrekking op het ontwerp van de structuur van de klasse in Delphi. Daarnaast is de naamgeving van de functies, procedures en attributen hier uitgewerkt. Die naamgeving is grotendeels gebaseerd op de resultaten in de functionele structuur.

De opgedane kennis van de latere stadia in de pilot is in de technische structuur toegepast. Dit heeft geresulteerd in de volledige beschrijving van het component.

```
unit database;

interface

uses DBGrids, DB, DBTables, Grids, Dialogs, SysUtils, cliënt,
logging;

type Tdb = class
  db: TDatabase;
  ds: TDatasource;
  q: TQuery;
  t: TTable;
  Log: TLog;
  error: string;
  public
    function sql_query( query: string ): TDatasource;
    function select_table( tableName: string ): TDatasource;
    constructor Create(); overload;
    procedure Free; overload;
  private
    procedure errorToLog( error: string );
```

```
    procedure writeToLog( logString: string);
    function checkConnection: boolean;
    procedure connect;
end;
```

Afbeelding 24: Databaseklasse in Delphi

In een later stadium bleek ook dat er een mogelijkheid moest zijn om opgetreden fouten en beveiligingsovertredingen te registreren. Het bijhouden van een logbestand bracht uitkomst. Om die reden zijn het niet eerder verklaarde attribuut 'Log' en de bijbehorende functies 'errorToLog' en 'writeToLog' hier onder gebracht.

6.4. Pilotontwikkelplan

De volgende activiteit was het opstellen van een pilotontwikkelplan aan de hand van de functionele en technische specificaties. Dit pilotontwikkelplan beschrijft de wijze waarop deze pilot zal worden gebouwd.
Allereerst zijn de pilotdelen ontworpen. Voor elk pilotdeel zijn er bouweenheden gedefinieerd. De pilotdelen heb ik vervolgens verdeeld in de volgende bouweenheden:

1. Het vaststellen van het juiste gebruik van Delphi met betrekking tot objecten
2. Implementeren van de databaseklasse
3. Ontwikkeling van de publieke functies.(public)
4. Ontwikkeling van de beschermde functies (private)
5. Onderzoeken en aanpassen foutafhandeling functies en procedures.
6. Onderzoeken en ontwikkelen registratie foutafhandeling.

Om kennis op te doen in het gebruik van een klasse in Delphi is een inleidende bouweenheid in het eerste pilotdeel opgenomen. Om het doel hiervan in een aantal woorden samen te vatten is een redelijke onduidelijke naamgeving ontstaan. Het doel hiervan is daarom extra toegelicht in het bijbehorende rapport. De meeste bouweenheden zijn echter vanzelfsprekend.

Vanuit beveiligingsoverwegingen is besloten om het laatste pilotdeel op te stellen (Bouweenheid 5 en 6). Met beveiliging wordt als voorbeeld een foutieve inlogpoging bedoeld. Naast het vastleggen van de overtredingen van de beveiliging, worden tevens de fouten geregistreerd. De registratie van mislukte queries of connectieverlies met de database zijn voorbeelden van foutregistraties.

6.5. Ontwerp software-bouweenheden

De volgende stap was het ontwerpen van de opgestelde bouweenheden. Het ontwerp van de bouweenheden moet van een dusdanig detail zijn, dat de implementatie hiervan mogelijk is.

6.5.1. Ontwerp bouweenheden

De aanpak om de bouweenheden zoveel mogelijk tekstueel te beschrijven, was in de eerdere pilots goed bevallen. De bouweenheden waren in dit stadium op basis van de functionele en technische specificaties tekstueel beschreven. Het ontwerp van de bouweenheden is vervolgens op basis van nieuwe kennis en inzicht tijdens de pilotontwikkeling aangevuld.
Een voorbeeld van een tekstuele beschrijving bij bouweenheid 4: “Ontwikkeling van de beschermde functies”.

In deze klasse zijn er vier beschermde functies, waarvan er twee niet eerder zijn gedefinieerd, namelijk errorToLog en writeToLog. Allereerst zal ik de al eerder besproken functies toelichten: checkConnection en connect.

<u>checkConnection</u>
De functie checkConnection is een functie met als resultaat een variabele van het type Boolean. Deze controleert of de er een geldige connectie is met de database. Indien dit het geval is wordt een ‘true’ teruggegeven, in het andere geval een ‘false’.

<u>Connect</u>
Deze procedure maakt een connectie met de database.

De nog niet eerder besproken procedures zijn ontworpen in het kader van de registratie van de foutafhandeling en worden in laatste bouweenheid behandeld.

Het tekstuele ontwerp van deze bouweenheden kon ook hier niet worden aangevuld door een van de UML-diagrammen.

6.5.2. Opstellen testspecificaties

Voor iedere bouweenheid heb ik testspecificaties opgesteld die voldoen aan het ‘black box’ principe. Ter illustratie zijn hieronder de testcases van bouweenheid 3: “Ontwikkeling van de publieke functies” opgenomen.

Case #	**Test**	**Invoer**	**Verwachte uitvoer**
3.0	De create functie werkt correct	dbKlasse = TDb.Create()	dbKlasse is een instantie van TDb De variabelen db, ds, q, t en Log zijn ook geinitialiseerd.
3.1	Selecteren van een tabel geeft een geldige tabel terug	dbKlasse.select_table(‘medewerker’)	dbGrid bevat de tabel medewerker.
3.2	Het uitvoeren van de functie sql_query werkt correct	dbKlasse.sql_query(‘Select * from medewerker’)	dbGrid bevat de twee medewerkers.

3.3	De Free functie geeft alle variabelen en de dbKlasse vrij	dbKlasse.Free()	De variabelen db, ds, q, t en Log zijn vrijgegeven. De dbKlasse is vrijgegeven.

Afbeelding 25: Testspecificaties bouweenheid 3

De testspecificaties zijn grotendeels tijdens de bouw opgesteld. Het inzicht in de werking van sommige bouweenheden werd tijdens de bouw namelijk vergroot. Een voorbeeld hiervan is dat bij de initialisatie van de databaseklasse de overige variabelen ook moesten worden geinitialiseerd. Het resultaat hiervan is te zien in de 'verwachte uitvoer' van bouweenheid 3.

6.6. Bouw software-bouweenheden

Na het opstellen van de tekstuele ontwerpen kon worden begonnen met de bouw van de bouweenheden. Deze paragraaf beschrijft het bouwen en het testen van de bouweenheden.

6.6.1. Het vaststellen in het juiste gebruik van Delphi met betrekking tot objecten

In het ontwerp van deze bouweenheid was gedefinieerd dat er naar de juiste manier gezocht moest worden voor de initialisatie van de databaseklasse. Er mocht maar één instantie van de klasse worden aangemaakt, om niet meerdere connecties toe te staan vanaf één applicatie. Bovendien wordt de kans op dataverlies door de applicatie hierdoor gereduceerd.
Op basis hiervan is onderzocht op welke verschillende logische manieren Delphi toestaat om een klasse te implementeren. Vervolgens zijn de voordelen en nadelen opgesteld voor de resultaten.

Alternatief	Voordelen	Nadelen
DLL-bestand	De functies van de klasse zijn herbruikbaar De applicatie hoeft het bestand maar één keer te initialiseren	Het bestand neemt relatief veel geheugen in beslag. Het dll-bestand kan worden afgesloten zonder dat de applicatie dit kan controleren
Klasse binnen de applicatie	De functies verbruiken weinig extra geheugen Ze zijn onderdeel van de applicatie	Een globale variabele moet worden gedefinieerd binnen een bestaande unit De bestaande instantie mag nooit worden vrijgegeven of beëindigd. (eventueel verlies van data).
Nieuw bestand	De functies verbruiken weinig extra geheugen De unit staat een 'overload' van de functies 'create' en 'free' toe	Er zal een globale variabele moeten worden aangemaakt om één instantie van het object te garanderen.

Afbeelding 26: Alternatievenlijst implementatie databaseklasse

Uit deze afweging is gebleken dat het gebruik van een bestand toegewijd aan de klasse het beste alternatief was.
Vervolgens is het bestand database.pas aangemaakt en is de minimale code, die aanwezig moet zijn om een klasse in Delphi te compileren, geschreven. De testcase had na de uitvoering een positief resultaat. Vandaar dat er zich ook geen problemen hebben voorgedaan.

6.6.2. Implementeren van de databaseklasse

Na het vaststellen van de wijze waarop in Delphi de databaseklasse geïmplementeerd moest gaan worden, kon worden begonnen met de daadwerkelijke implementatie van de databaseklasse. Deze bouweenheid omvatte het opstellen van de functies die de databaseklasse moet ondersteunen en de implementatie van de variabelen.
Als eerste stap heb ik de variabelen aangemaakt. Deze variabelen zijn aangemaakt op grond van het onderzoek dat ik in het kader van de functionele structuur heb uitgevoerd. Hieruit is gebleken dat voor een correcte communicatie met de database er minimaal een mogelijkheid moest zijn ondergebracht voor:

- Het opzetten van de verbinding
- Het selecteren van een tabel
- Het uitvoeren van een query
- De resultaten van de tabel en query

Op basis hiervan zijn er vier variabelen ondergebracht in de klasse. Daarnaast zijn de functies ,beschreven in de functionele structuur, als statische functie opgenomen in de databaseklasse. De functies zijn statisch en moesten daarom een statisch resultaat leveren om de testcases succesvol te kunnen doorlopen.
Het testen van deze bouweenheid had vooral betrekking op de controle of de functies aangeroepen konden worden en of een eventuele meegeleverde variabelen weergegeven werd.
De testcases zijn verder succesvol doorlopen en er zijn bij het bouwen van deze bouweenheid verder geen problemen geconstateerd.

6.6.3. Ontwikkeling van de publieke functies.(public)

Nadat de basis van de databaseklasse was opgebouwd en getest, kon worden begonnen met het bouwen van de functies. Deze bouweenheid omvatte de bouw en het testen van de publieke functies zoals die in de databaseklasse waren opgenomen:

- Sql_query
- Select_table
- Create
- Free

Hoewel het herschrijven van functies vrij nieuw voor mij was, vormde dat na het doorlezen van enkele voorbeelden geen enkel probleem. Het gebruik van ‘overload’ achter de functiedefinitie in een klasse zorgt ervoor dat de compiler de herschreven

functie gebruikt in plaats van de originele. Om de ingewikkelde create en free functie niet geheel overnieuw te schrijven, zoals het zoeken en toewijzen van beschikbare geheugenruimte, kan de originele functie alsnog worden uitgevoerd door het gebruik van 'inherited'.
Waarbij 'inherited Create' ervoor zorgt dat de originele create functie wordt uitgevoerd.

Het bouwen van de functie sql_query zorgde echter voor veel problemen. In eerste instantie was de functie zo ontworpen dat de TDatasource en de TQuery werden geinitialiseerd binnen de functie. Hoewel dit een prima oplossing leek, volgden bij het testen vreemde problemen. Bij het herhaaldelijk uitvoeren van dezelfde query werd iedere keer een ander resultaat teruggegeven. Omdat de functie goed compileerde, leken er zich in de code syntactisch geen problemen voor te doen. Om tot de oorzaak van het probleem te komen is gebruik gemaakt van het debugprincipe "trial-and-error". Dit principe is gebaseerd op het toevoegen van een debugcode en verwijderen van de aanwezige code. Ter illustratie is een fragment hiervan opgenomen:

```
//ds := TDataSource.Create( nil );

showMessage('[debug] Initialisate tabel');

t := TTable.Create(nil);
//t.DatabaseName := db.DatabaseName;

//q := TQuery.Create( nil );
//ds.CleanupInstance;
```

Afbeelding 27: Een fragment van de debugcode.

Nadat alle operaties waren voorzien van het commentaarteken '//' kon worden geprobeerd om te achterhalen waar de fout zich voordeed. Na een aantal dagen debuggen bleek dat alle geraadpleegde objecten moesten zijn aangemaakt aan het begin van een functie. Bovendien kostte het aanmaken van een object relatief veel tijd, wat er toe leidde dat de functie een operatie uitvoerde op een object wat nog geen geheugenruimte toegewezen had gekregen. Dit verklaarde tevens de verschillende resultaten die de functie leverden.
Dit heeft onverwacht ontzettend veel tijd gekost en uiteindelijk geleid tot een alternatieve oplossing om het aanmaken van de objecten in de create-functie van de klasse uit te voeren. De objecten zullen nu eenmalig worden aangemaakt en worden vrijgegeven. Naast het aanmaken van de objecten moest ook het invullen van de variabelen, zoals 'Aliasname', hier worden opgenomen.

Na het alternatief geïmplementeerd te hebben, bleek er wederom een fout in de functies sql_query en select_table te zitten. Deze fout was te herleiden naar de functie Create, waarin het commando 'inherited Create' op de verkeerde positie was geplaatst.

De positie waar dit commando moest worden gegeven was cruciaal. Bij ‘create’ is het van belang dat de instantie van de klasse geheugenruimte toegewezen heeft gekregen voordat verdere operaties worden uitgevoerd. Hoewel dit bij mij bekend was, concludeerde ik hier niet uit dat de positie van het commando hier enige invloed op had.

Na het verplaatsen van het commando, zijn de testcases opnieuw doorlopen en dit keer met succes.

6.6.4. Ontwikkeling van de beschermde functies (private)

Na de ontwikkeling en de uitgebreide tests die uitgevoerd waren op de vorige bouweenheid, kon worden begonnen met het ontwerpen van de beschermde functies. De beschermde functies zijn de functies die alleen aangeroepen kunnen worden binnen het object. In deze bouweenheid zouden echter alleen de functies worden ontwikkeld die geen betrekking hadden op de foutafhandeling en / of -registratie.

De twee overgebleven functies waren ‘connect’ en ‘checkConnection’. Uit het onderzoek in de functionele structuur is gebleken dat objecten van de klasse ‘TDatabase’ de daadwerkelijke connectie opzetten. Hieruit bleek ook dat de variabele ‘connected’ van het type Boolean was en de verbinding konden worden opgezet door deze op true te zetten.

Het ontwikkelen van deze functies was daarom redelijk snel verlopen en zonder problemen. Het positieve resultaat van de uitgevoerde testcases was daar een onderdeel van.

6.6.5. Onderzoeken en aanpassen foutafhandeling functies en procedures.

Deze bouweenheid omvatte het onderzoek naar en de realisatie van de juiste foutafhandeling binnen de tot nu toe gemaakte functies en procedures.
Om een fout in Delphi af te vangen is hetzelfde principe als in Java gehanteerd. Dit principe houdt in dat de operaties binnen een ‘try—catch—finally’-loop worden uitgevoerd. Zo’n loop wordt begonnen met het commando ‘try’ (proberen), waarna alle operaties dienen te worden opgenomen. Zodra een fout zich voordoet binnen dit segment wordt een fout gecreëerd van een bepaald type dat in Delphi is vastgelegd. Waarbij voor iedere soort fout een bepaald type bestaat. Dit type moet vervolgens ‘gevangen’ (catch) worden in het volgende segment van de loop. In dit segment wordt de fout afgehandeld en worden de benodigde operaties uitgevoerd.
De loop kan worden afgesloten met een finally, waarin bepaalde operaties worden uitgevoerd. Deze operaties zijn vooral gericht op het opschonen van de fouten.

Bij het opstellen van deze foutafhandelingen is vooral uitgegaan van de fouten die zich voorgedaan hebben in bouweenheid 3: “Ontwikkeling van de publieke functies”. Omdat de fouten van die bouweenheid van dusdanig omvang waren, en meerdere soorten fouten omvatte, is besloten om een bij iedere procedure een algemene foutafhandeling op te stellen. Uit de testcases kon worden opgemaakt dat de bouweenheid correct was uitgevoerd.

6.6.6. Onderzoeken en ontwikkelen registratie foutafhandeling.

Na de realisatie en het testen van de vorige bouweenheid over foutafhandeling, moest de wijze van registratie worden onderzocht en gebouwd. De fouten die zich voor hadden gedaan in het kader van beveiligingsovertredingen of overige fouten moesten buiten de applicatie ook benaderbaar zijn.

De beste oplossing hiervoor was het opslaan van de fouten in een apart document op het werkstation. Om dit te kunnen realiseren moest eerst worden onderzocht hoe er in Delphi naar bestanden geschreven en hoe eruit gelezen kan worden. Hieruit bleek dat de klasse 'TextFile' in Delphi deze mogelijkheden biedt.

Daarnaast moest er in het tekstbestand onderscheid gemaakt worden tussen beveiligingsovertredingen en algemene applicatiefouten. De oplossing hiervoor was het schrijven van twee aparte functies. Eén functie die de fouten registreert en de ander die de overtredingen registreert. Om de registratie los van de databaseklasse ook te kunnen gebruiken was besloten om een aparte klasse te maken die deze operaties ondersteunt.

Deze klasse was niet eerder gedefinieerd en het besluit tot ontwikkeling van een aparte klasse is in deze pilot genomen. De resultaten van deze bouweenheid zijn daarom voor de overzichtelijkheid verwerkt in de eerdere ontwerpen van de bouweenheden.

Een instantie van de klasse TextFile is bij initialisatie een pointer naar een bestand op het werkstation. Om die reden moesten een tweetal extra variabelen worden gedefinieerd waarin de locatie en de bestandsnaam konden worden vastgelegd. Dit heeft geleid tot de volgende klassendefinitie.

```
Type
  TLog = class
    fileName: string;
    path: string;
    logFile: textfile;
    public
      procedure write( value: string );
      procedure error( value: string );
end;
```

Afbeelding 28: Klassendefinitie TLog

De verdere implementatie bracht verder geen problemen met zich mee. Hoewel het opzoeken van de geschikte operaties voor het schrijven naar een bestand redelijk wat tijd in beslag nam, was de implementatie verder goed verlopen. Bij het testen hebben zich ook verder geen problemen voorgedaan.

6.7. Invoeringsprocedure

De activiteit die na het bouwen van de software-bouweenheden is uitgevoerd was het opstellen van de invoeringsprocedure voor de pilot. Omdat de databaseklasse een onderdeel is van de applicatie, dient de applicatie opnieuw te worden gecompileerd en als geheel te worden ingevoerd. De enige verandering die in dit stadium had kunnen optreden, was een verandering aan de eerder geïnstalleerde software. Hieruit is het volgende stappenplan ontstaan.

1. Controle invoering hard- en software
   Voordat met de invoering kan worden begonen moet eerst worden gekeken of de onderstaande software nog operationeel is:
   a. MySQL
   b. MyODBC
2. Invoeren nieuwe applicatie
   Vervolgens wordt de opnieuw gecompileerde applicatie overgezet, aangezien de databaseklasse onderdeel is van de applicatie.

Vervolgens heb ik de testspecificaties opgesteld die de controle op de juiste invoering ondersteunen.

6.8. Testen en beoordelen

In deze activiteit, die in de planning is opgenomen na de invoeringsprocedure, wordt de gerealiseerde pilot getest en beoordeeld. Allereerst zijn alle testcases per bouweenheid doorlopen en zijn de resultaten daarvan toegelicht. Op basis daarvan is de pilot beoordeeld of het gewenste resultaat, gesteld in de definitiestudie, is behaald.

Naast de mislukte cases in de derde bouweenheid, is de ontwikkeling en het testen over het algemeen goed verlopen. Op dit moment is het mogelijk om via benadering van deze klasse een sql query uit te voeren op de database en het resultaat weer te geven. De verbinding met de database wordt opgezet en de klasse handelt alle fouten af en bovendien worden deze ook geregistreerd in een daarvoor bestemd bestand.

Het gewenste resultaat zoals die in de definitiestudie is gedefinieerd:

- Het gewenste resultaat van deze pilot is het afvangen van de communicatie tussen de database en de grafische user interface. De functies zullen geheel functioneel moeten zijn of in het uiterste geval een vergelijkbare testwaarde moeten teruggeven, zodat de subsystemen deze zonder problemen kunnen aanroepen

De interface heeft naast een functioneel karakter ook de registratie van de fouten afgevangen, wat resulteert in een pilot met het gewenste resultaat.

6.9. Afronding

Na de controle van het rapportopmaak, kon het document worden gemarkeerd als definitief. Het resultaat van dit document is als bijlage opgenomen in dit verslag.

Door de uitloop van bouweenheid 3 en 6 heb ik de pilot niet kunnen afronden in de geplande tijd. De bouw van deze pilot heeft anderhalve week meer in beslag genomen, als gevolg van de problemen bij bouweenheid drie. Gezien de geplande tijd voor de volgende pilot, was het in dit stadium moeilijk in te schatten of er nog tijd ingehaald kon worden.

7. Pilot 4: Subsystemen één tot en met drie

Na de afronding van de derde pilot kon worden begonnen met de ontwikkeling van de vierde en tevens laatste pilot in het pilotontwikkelingstraject. Tijdens de definitiefase is deze pilot ingedeeld als vierde pilot. De inhoud van deze pilot is het ontwikkelen en bouwen van de functionaliteiten van de subsystemen één, twee en drie.
De opbouw van dit hoofdstuk begint met de paragraaf over de activiteiten die zijn verricht in het kader van het plan van aanpak. In paragraaf 7.2 wordt de functionele structuur toegelicht en in de daaropvolgende paragraaf wordt het pilotontwikkelplan opgesteld. Het ontwerp van de bouweenheden gedefinieerd in het pilotontwikkelplan worden vervolgens in paragraaf 7.4 behandeld. De procesgang omtrent de bouw wordt in paragraaf 7.5 behandeld. De invoeringsprocedure en het beoordelen van de pilot zijn in respectievelijk paragraaf 7.6 en 7.7 toegelicht. Tot slot wordt de afronding van de pilot in paragraaf 7.8 besproken.

7.1. Plan van aanpak

Om de ontwikkeling van de pilot gestructureerd te kunnen aanpakken, is een plan van aanpak geschreven. Na het bestuderen van de activiteiten die in het kader van deze pilot moesten worden uitgevoerd is een planning gemaakt. De niet uitgevoerde activiteiten heb ik toegelicht met een reden van verval:

De overige activiteiten die IAD voorschrijft worden niet voor dit product uitgevoerd.

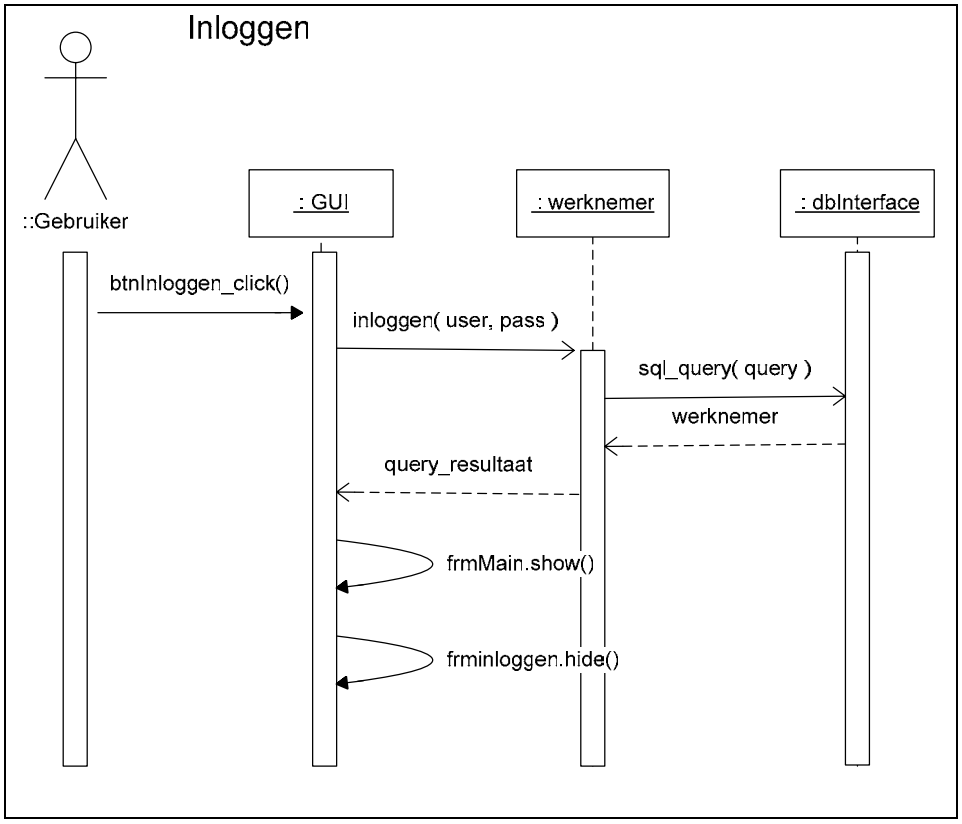
- Voorbereiden pilotontwerp-workshop
  Een workshop voor de pilot is reeds gehouden in de definitiefase.
- Specificeren globaal-technische structuur pilot
  De technische structuur van de pilot is onveranderd sinds de laatste pilot.
- Specificeren globaal-organisatorische inrichting
  De invloed op de organisatorische aspecten zijn beschreven in de definitiefase. Deze pilot levert geen extra veranderingen met zich mee..
- Aanpassen externe componenten
  Er hoeven geen externe componenten te worden aangepast.
- Wijzigen andere informatiesystemen
  Er is geen sprake van een ander informatiesysteem.
- Bouwen conversie-tools
  Deze pilot vereist geen conversie-tools
- Ontwerpen handmatige procedures
  De handmatige procedures zijn hier niet van toepassing.
- Integreren bouweenheden
  De bouweenheden worden bij implementatie direct geïntegreerd, doordat deze onderdeel zijn van de applicatie.
- Samenstellen opleidingsmateriaal pilot
  Er is geen behoefte aan opleidingsmateriaal voor enkel deze pilot.
- Samenstellen handleiding pilot
  Deze deelactiviteit is opgenomen als laatste bouweenheid van deze pilot

Door de uitloop van de eerdere pilots en de tijdsdruk van het project, heb ik besloten om de activiteiten in combinatie met de tijd voor het schrijven van het verslag in te plannen tot het einde van de afstudeerstage.

7.2. Functionele structuur

Na het opstellen van het plan van aanpak heb ik de functionele structuur van de pilot opgesteld. Het doel van deze activiteit was het beschrijven van de functionele inhoud en de reikwijdte van de pilot. Tijdens de fase definitiestudie had ik reeds geconcludeerd dat in deze pilot de essentiële deelsystemen van de applicatie ontwikkeld moesten worden. In het rapport definitiestudie had ik een aantal use-cases opgesteld. Die use-cases zijn vervolgens verdeeld in een aantal deelsystemen.

Aangezien de use-cases een globale indruk geven van de interactie met het systeem, heb ik gezocht naar een manier om dit gedetailleerder weer te geven binnen de technieken van UML. Dit kon worden gerealiseerd door het ontwerpen van sequencediagrammen. Voor iedere use-case is besloten om een sequencediagram op te stellen, als dat meer inzicht gaf op de interactie met het systeem. Het sequencediagram van de use-case “Inloggen”, onderdeel van het eerste subsysteem is opgenomen ter illustratie.



Afbeelding 29: Sequencediagram Inloggen

Het diagram geeft een overzicht van de interactie met de objecten die nodig zijn om een verloskundige (gebruiker van het systeem), in het geval van de juiste combinatie van de gebruikersnaam en wachtwoord, toegang te verlenen tot het systeem.
Met het ontwerpen van de sequencediagrammen en het toevoegen van een tekstuele toelichting leek de functionele structuur van de pilot voldoende duidelijk.

7.3. Pilotontwikkelplan

De volgende activiteit was het opstellen van een pilotontwikkelplan op basis van de opgestelde functionele specificaties. Dit pilotplan beschrijft op welke wijze de pilot gaat worden gebouwd. Om tot een pilotontwikkelplan te komen heb ik de werkwijze gehanteerd die in IAD wordt voorgeschreven. De pilot heb ik opgedeeld in een drietal pilotdelen. Voor ieder pilotdeel, heb ik bouweenheden opgesteld, die afzonderlijk van elkaar kunnen worden gebouwd. De volgende negen bouweenheden zijn gedefinieerd:

1. Definiëren basisstructuur
2. Definiëren constanten en globale variabelen
3. Definiëren standaard voor toelichting in de code
4. Omzetten klassen naar objecten
5. Ontwikkelen functies en procedures
6. Ontwikkelen functies Contactmoment cliënt
7. Ontwikkelen functies subsysteem Zwangerschap
8. Ontwikkelen functies subsysteem Administratie
9. Opstellen gebruikershandleiding subsystemen

Er was eerst besloten om het tweede pilotdeel: “Ontwikkelen klassen en bijbehorende operaties” als bouweenheid op te nemen en te ontwikkelen. Hoewel dit bij het opzetten van het pilotontwikkelplan geen slecht idee leek, was toch besloten bij het ontwerp om deze op te delen in meerdere bouweenheden. De verwachting was dat de omvang van de pilot relatief klein zou zijn. De klassen waren tenslotte al ontwikkeld en hoefden alleen nog te worden omgezet naar objecten. De operaties in de klassen moesten echter nog worden ontworpen en om alle bouweenheden van ongeveer gelijke omvang op te stellen, is besloten om dit pilotdeel alsnog op te delen. Hieruit zijn de bouweenheden 4 en 5 ontstaan.

Als laatste heb ik na de bouweenheden de ontwikkelaanpak gedefinieerd. In deze ontwikkelaanpak wordt tevens de teststrategie beschreven. Hierin is vastgelegd dat iedere bouweenheid na implementatie zal worden getest aan de hand van de testspecificaties.

7.4. Ontwerp software-bouweenheden

Na het definiëren van de bouweenheden was de volgende geplande activiteit het ontwerpen van de bouweenheden. Net als in de vorige pilots dient het ontwerp zodanig te zijn opgesteld, dat het mogelijk is om op basis van het ontwerp de bouweenheden te implementeren.

7.4.1. Ontwerp bouweenheden

Het ontwerp van de bouweenheden heb ik op ongeveer dezelfde wijze aangepakt als in de vorige pilots. Het ontwerp van de bouweenheden heb ik ook weer aangevuld tijdens de bouw op basis van opgedane kennis en inzicht. Op basis van de functionele specificaties en de kennis van de eerdere pilots is allereerst een tekstueel ontwerp gemaakt. Naast het tekstuele ontwerp was er voor gekozen om langer stil te staan bij het ontwerp. De omvang van de bouweenheden in deze pilot was groter dan de eerdere ontworpen bouweenheden. Bovendien moest na deze pilot de applicatie voldoen aan de functionele systeemeisen die waren opgesteld in de definitiefase.

Bij iedere bouweenheid is gekeken of er met behulp van een diagram meer detail kon worden aangebracht. De UML-diagrammen hadden echter geen toegevoegde waarde in dit stadium en daarom is besloten om voor iedere functie een pre- en postconditie op te stellen. Op deze manier kon er na de implementatie van de bouweenheid makkelijker getest worden of deze voldeed aan de verwachtingen.

Voor sommige bouweenheden kon worden volstaan met een tekstuele beschrijving. Dit was geconcludeerd uit het gewenste resultaat van de betreffende bouweenheid. Een voorbeeld hiervan is bouweenheid 3 “Standaardisering codecommentaar”:

Om de applicatie overdraagbaar te maken en duidelijk leesbaar voor eventuele toekomstige programmeurs, moet gekeken worden hoe code op de juiste wijze van commentaar moet worden voorzien. Er is geen vaste regelgeving omtrent het toevoegen van commentaar op programmeercode.

Door het Universitair Dienstencentrum voor Informatica en Telematica in Leuvens zijn er een aantal standaarden voor het geven van commentaar ontwikkeld voor de taal PHP. Hoewel deze taal in vele opzichten verschilt van Delphi, is het geven van commentaar onafhankelijk te beschouwen van de programmeertalen.

Naast deze bron is ook het document van de “Orange County Delphi Users Group” beschouwd. In dit document, dat ook online beschikbaar is, staan de standaarden voor het programmeren in Delphi. Naast een aantal programmeertips staat er ook een standaard in voor het commentaar geven in programmeercode.

De standaard die in het document is terug te vinden komt overeen met een van de standaarden die staat gedefinieerd in het universitaire onderzoek. Op basis van deze twee documenten wordt de standaard die in het document van “Orange County Delphi Users Group” is terug te vinden, gehanteerd als standaard voor dit project.

De standaard die daarin naar voren komt wordt als volgt geformuleerd:

1. Unit and Function Prologs

Each unit (Pascal source file) should begin with a comment block, or "prolog" that generally describes the contents of the unit. Each class definition and each function (procedure/method) within a unit should have a separate function prolog.

In deze bouweenheid is vastgesteld op welke manier het commentaar in de applicatie is geplaatst. De gekozen standaard en beredenering hiervoor is na de implementatie ook toegevoegd aan het ontwerp van de bouweenheid.

7.4.2. Opstellen testspecificaties

Voor bijna iedere bouweenheid heb ik testspecificaties ontworpen die voldoen aan de eisen die gesteld worden aan “black box”-testen. Testen met behulp van “black box” testsets had als voordeel dat het verwachte resultaat van een functie snel kon worden vastgesteld. Voor de bouweenheden waarvan het resultaat moeilijk te testen was, zijn niet de “black box” testsets opgesteld. Hiervoor zijn testen gespecificeerd om te controleren of het resultaat van de bouweenheid voldoet aan een aantal eisen.
Hieronder zijn ter illustratie de testspecificaties opgenomen van de bouweenheid 1 “Definiëren basisstructuur”. In deze bouweenheid worden de globale functies en de functies voor de MD5-encryptie ontwikkeld.

Case #	Test	Invoer	Verwachte uitvoer
1.0	getSelectedComboItem geeft een geselecteerd combo-item terug als string	Selecteer een veld uit een combobox en voer de functie uit met value 0	Het geselecteerde item wordt als string teruggegeven
1.1	getSelectedComboItem geeft een geselecteerd combo-item terug als index	Selecteer een veld uit een combobox en voer de functie uit met value 1	Het geselecteerde item wordt als index teruggegeven
1.2	getSelectedRadioItem geeft een geselecteerd radio-item terug als string	Selecteer een item uit een radiogroup en voer de functie uit met value 0	Het geselecteerde item wordt als string teruggegeven
1.3	getSelectedRadioItem geeft een geselecteerd radio-item terug als index	Selecteer een item uit een radiogroup en voer de functie uit met value 1	Het geselecteerde item wordt als index teruggegeven
1.4	groupActivate zet de gegeven groep op de voorgrond	groupActivate(grpClient)	De grpClient wordt op de voorgrond weergegeven
1.5	groupDeactivate verbergt de groep	groupDeactivate(grpClient)	De grpClient wordt verborgen
1.6	deactivateAllGroups verbergt alle groepen	deactivateAllGroups(alle groepen)	De groepen zijn allemaal verborgen
1.7	WindowExists zoekt naar een bepaalde applicatie in de proceslijst van Windows	WindowExists(‘W!se Delivery’, ‘TForm’)	Het window wordt gevonden.
1.8	isStrEmpty controleert of een string daadwerkelijk leeg is	testString := ‘’; isStrEmpty(testString)	Het resultaat van de functie is true.
1.9	strToArr converteert de gegeven string naar een gegeven array.	testString := ‘Henk’; voornaam: TNames; strToArr(testString, voornaam);	voornaam bevat de naam ‘Henk’
1.10	arrToStr converteert een gegeven array naar een string	testString := arrToSrr(voornaam)	testString is ‘Henk’
1.11	MD5Print() geeft een geldige MD5 encryptie	MD5Print(MD5String(‘hallo’))	MD5 ‘hallo’ = 598d4c200461b81522a3328565c25f7c

Afbeelding 30: Testspecificatie bouweenheid 1

Net als in de voorgaande pilots zijn de testspecificaties deels gedurende de bouw van de bouweenheden opgesteld. Tijdens de bouw kon het zich voordoen dat een bepaalde

vooraf ontworpen functie moest worden aangepast. Deze veranderingen moesten dan ook worden doorgevoerd in de testspecificaties. Een concreet voorbeeld hiervan deed zich voor in de eerste bouweenheid, waarin na het lezen van de RFC voor MD5-encryptie het aantal functies pas bekend was.

7.5. Bouw software-bouweenheden

Na het maken van het ontwerp van de bouweenheden ben ik begonnen met de bouw hiervan. In deze paragraaf wordt de bouw en het testen van iedere bouweenheid besproken.

7.5.1. Definiëren basisstructuur

In de eerste bouweenheid moesten aan de hand van het ontwerp een aantal functies worden geïmplementeerd. De globale functies / procedures die in de applicatie hergebruikt konden worden, waren in het ontwerp reeds beschreven door middel van pre- en postcondities. Bovendien was de structuur van de unit ook al ontworpen.

```
function getSelectedComboItem( cmbBox: TComboBox; value: integer ): string;
function getSelectedRadioItem( rdoBox: TRadioGroup; value: integer ): string;
procedure groupActivate( groupBox: TGroupBox );
procedure groupDeactivate( groupBox: TGroupBox );
procedure deactivateAllGroups( grp1, grp2, grp3, grp4, grp5, grp6, grp7, grp8,
grp9: TGroupBox );
function WindowExists(AppWindowName, AppClassName: string): Boolean;
function isStrEmpty( const testString: string ): Boolean;
function strToArr( const str: string; var typeArray : array of char ):
boolean;
function arrToStr( const typeArray: array of const ): string;
```

Afbeelding 31: Unitstructuur globale functies

Na de implementatie van deze unit structuur, konden de functies / procedures worden gebouwd. Om het commentaar toe te kunnen voegen aan de functies en het resultaat na de bouw te kunnen testen, zijn de pre- en postcondities opgesteld. De functie ‘getSelectedComboItem’ is hier als voorbeeld opgenomen.

Functienaam	getSelectedComboItem
Resultaat Type	String
Argumenten	cmbBox: TComboBox value: integer;
Preconditie	cmbBox not null value = [0 | 1]
Postconditie	Het geselecteerde item is teruggegeven als resultaat zoals in het argument value is aangegeven
Uitzonderingen	1. Er is geen item geselecteerd in de gegeven cmbBox. 2. Er zijn geen items aanwezig in de gegeven cmbBox.
Oplossing uitzonderingen	Het resultaat = -1

Afbeelding 32: Pre- en postconditie functie getSelectedComboItem

De beschrijving en de pre- en postcondities van iedere functie zijn vertaald naar het Engels en opgenomen als commentaar (‘prolog’).

Na de implementatie van de globale functies, moest een unit worden gebouwd die de MD5-encryptie realiseert. Tijdens het ontwerp van deze bouweenheid was de RFC die het MD5 algoritme beschrijft al bekeken. In deze standaard wordt een aantal standaard functies beschreven die geïmplementeerd moesten worden. Om aan de eisen te voldoen zijn de functies ontwikkeld die daarin worden voorgeschreven.
De functies zijn verder niet aangevuld op inhoud, aangezien de RFC hier voldoende aanvulling op geeft.

Het invoeren van een MD5-encryptie droeg bij aan de veiligheid bij het inloggen van de verloskundig. De bouw hiervan had aanzienlijk meer tijd in beslag genomen dan ingeschat. Er was wel overwogen om bestaande code te zoeken die de implementatie van het MD5-algoritme ondersteunde. Al snel bleek dat het gebruik van de ontwikkelde versies in Delphi alleen voor educatieve doeleinden waren bestemd. De broncode die ik nodig had om het te verwerken kon alleen worden verkregen tegen betaling.

Na het uitvoeren van de testcases had ik de planning aangepast, aangezien de bouw van deze bouweenheid al voor twee dagen uitloop had gezorgd.

7.5.2. Definiëren constanten en globale variabelen

Om terugkerende datatypen en variabelen niet bij iedere unit hoeven te definiëren, is ervoor gekozen om dat in deze bouweenheid te realiseren. In deze bouweenheid worden alle datatypen eenmalig gedefinieerd, zodat deze in de gehele applicatie kunnen worden gebruikt.
Naast het definiëren van de datatypen implementeert deze bouweenheid tevens een aantal constante waarden die met regelmaat in de functies worden geraadpleegd. Een fragment van de definitie van constanten en datatypen is hieronder opgenomen.

```
const
  C_SIZE_NAMES = 150;
  C_SIZE_DATES = 10;
  C_SIZE_ZIPCODE = 6;
  C_SIZE_TEL = 11;
  C_SIZE_SMALLNAME = 50;
  C_SIZE_ZWNR = 6;

type

  TNames = array[0..C_SIZE_NAMES] of char;
  TDates = array[0..C_SIZE_DATES] of char;
  TZipCode = array[0..C_SIZE_ZIPCODE] of char;
  TTel = array[0..C_SIZE_TEL] of char;
  TSmallName = array[0..C_SIZE_SMALLNAME] of char;
  TZwnr = array[0..C_SIZE_ZWNR] of char;
```

Afbeelding 33: Fragment definitie constanten en datatypen

Dit opgenomen fragment is een rechtstreekse kopie uit het pilotontwikkelrapport van deze pilot. De inhoud van dit fragment is opgebouwd door de maximale waarden voor ieder veld uit de database op te nemen als constante in de applicatie. Het datatype dat voor een cliëntnaam in de tweede pilot “Database” is gedefinieerd, is varchar(150). De constante C_SIZE_NAMES = 150 is het resultaat van de lengte van de varchar. Vervolgens wordt het datatype gedefinieerd als een array van ‘char’ met de maximale lengte zoals die in de constante is bepaald. Waaruit kan worden geconcludeerd dat velden met het datatype varchar(150) in de database, in Delphi gelijk zijn aan array[0..150] of char.
Naast deze constanten en datatypen is er een globale variabele aangemaakt voor het de harde return in een berichtvenster. Een harde return in een berichtvenster is in Windows een combinatie van de ASCII-Codes 10 en 13. Nummer 10 stelt een Carriage return voor en nummer 13 de Line Feed. Een carriage return, ook wel afgekort tot CR, is het terugbrengen van de cursor naar het begin van de regel. Een Line Feed, ook wel afgekort tot LF, houdt in dat er aan een nieuwe regel begonnen wordt. Om te voorkomen dat deze elke keer moeten worden ingevoerd bij een nieuwe regel, is hier constante voor aangemaakt.
In de derde pilot “GUI-Database interface” van dit project is bepaald dat er maar één instantie van het object TDb mag zijn. Vandaar dat een instantie bij de start van de applicatie wordt aangemaakt, die overal in de applicatie benaderbaar is.

Er was tijdens het ontwerp besloten om alleen dit fragment als toelichting op te nemen in het rapport en niet alle constanten en datatypen. Het opgenomen fragment bood voldoende inzicht in de definitie en opzet van de overige constanten en datatypen. Door de bouw van deze bouweenheid was de basis gelegd voor de invoering van de objecten en bijbehorende functies. De testspecificaties die zijn opgesteld zijn succesvol uitgevoerd.

7.5.3. Definiëren standaard voor toelichting in de code

Nadat ik de bouweenheid “Definiëren constanten en globale variabelen” gebouwd en getest had, moest een standaard worden vastgesteld voor het schrijven van commentaar in de programmeercode. Om de toekomstige applicatie overdraagbaar en leesbaar te maken voor eventuele toekomstige programmeurs, moet er in de code commentaar worden opgesteld. Deze tekstuele toelichting geeft een beschrijving van de functie / procedure voordat deze wordt aangeroepen.
Om de code beter leesbaar te maken is besloten om in deze bouweenheid te onderzoeken welke standaarden er gehanteerd worden voor het toevoegen van commentaar in de programmeercode. Het resultaat hiervan was vervolgens in het ontwerp van de bouweenheid opgenomen.

Na het doorlezen van een aantal documenten op internet, bleken het Universitair Dienstencentrum voor Informatica en Telematica in Leuvens en de “Orange County Delphi Users Group” de meest gewenste resultaten te geven. Aangezien het geraadpleegde document van de universiteit in Leuvens de taal PHP gebruikte, was het document van de Delphi Users Group hiermee vergeleken. Op basis van deze

vergelijking was te zien dat er veel vergelijkbare standaarden waren voor het opstellen van commentaar in de programmeercode.

> ***1. Unit and Function Prologs***
>
> *Each unit (Pascal source file) should begin with a comment block, or "prolog" that generally describes the contents of the unit. Each class definition and each function (procedure/method) within a unit should have a separate function prolog.*

Afbeelding 34: Gekozen standaard.voor toelichten programmeercode

Op basis van de standaard is de gehele applicatie voorzien van commentaar. Na het succesvol uitvoeren van de testcases, kon deze bouweenheid als afgerond worden beschouwd.

7.5.4. Omzetten klassen naar objecten

Om de klassen uit het klassendiagram als objecten in Delphi te kunnen gebruiken, moesten deze worden omgezet naar classes in Delphi. De datatypen van de attributen in de objecten waren reeds in de vorige bouweenheid geïmplementeerd. Om bij het implementeren van de objecten zoveel mogelijk aan de standaard te houden, zoals Borland Delphi die voorschrijft, is de eis dat iedere objectnaam begint met ‘T’.

- De naamgeving van de objecten beginnen met een ‘T’ gevolgd door de naamgeving vastgesteld in het klassendiagram.
- De attributen zijn publiek gedefinieerd, zodat alle units de attributen kunnen wijzigen en opvragen.
- Alle functies en procedures worden opgenomen als zogeheten ‘dummies’. Dummy-functies zijn functies die geen aanvullende code hebben, naast de definitie van de functie.

Afbeelding 35: Gestelde eisen aan de objecten

Verder zijn er twee andere eisen opgesteld die ervoor moeten zorgen dat in een later stadium er geen onvoorziene problemen kunnen optreden met de definitie van de objecten. Na het uitvoeren van de tests, bleken de resultaten succesvol.

7.5.5. Ontwikkelen functies en procedures

Na het testen van bouweenheid 4: “Omzetten klassen naar objecten”, konden de algemene functies en procedures worden ontwikkeld. Deze algemene functies hadden als doel om de velden in de objecten om te zetten naar een SQL query en deze vervolgens uit te voeren. Het doel van de functies in deze bouweenheid was het uitvoeren van de operaties op de database. Dit heeft geleid tot vier functies, die bij ieder object zijn geïmplementeerd:

- Vastleggen
- Wijzigen
- Verwijderen
- Opvragen

Deze functies zijn vervolgens toegelicht met behulp van pre- en postcondities. Deze functies moesten ervoor zorgen dat de database alleen door bepaalde functies kon worden benaderd. Bij het registreren van eventuele fouten in de database, kon de oorzaak snel worden gevonden.
De bouweenheid is vervolgens succesvol getest aan de hand van de testspecificaties.

7.5.6. Ontwikkelen functies Contactmoment cliënt

In deze bouweenheid zijn de functies ontwikkeld die de functionaliteiten van het subsysteem *Contactmoment cliënt* realiseren. De tekstuele toelichting tijdens de ontwikkeling van de bouweenheden is in dit stadium uitgebreid met beschrijvingen van de functies. Naast de beschrijvingen zijn ook de pre- en postcondities toegevoegd. Een voorbeeld van deze aanvulling op het ontwerp is hieronder van de functie inloggen.

Object	Werknemer
Functienaam	Inloggen
Resultaat Type	Boolean
Argumenten	Username : string Password : string
Preconditie	De username bevat een string met de voornaam, tussenvoegsel en de achternaam van de werknemer Het password is het wachtwoord van de werknemer
Postconditie	De gegevens van de werknemer zijn vastgelegd in het object werknemer
Beschrijving	1. Het argument Password wordt omgezet naar een MD5 encrypt wachtwoord. 2. Er wordt een query opgezet die de gegevens van de werknemer selecteert en voldoet aan de argumenten. 3. De query wordt uitgevoerd 4. Uit het aantal resultaten wordt opgemaakt of de query succesvol was. 5. De resultaten worden uit de datasource in het object geplaatst 6. True wordt als resultaat van de functie teruggegeven.
Uitzonderingen	Er is geen werknemer gevonden die voldoet aan criteria gegeven in de argumenten
Oplossing uitzonderingen	Resultaat uitzondering tonen aan de gebruiker en vastleggen in het logbestand (beveiligingsovertreding)

Afbeelding 36: Beschrijving functie *inloggen*

Voor iedere functie opgenomen in de sequencediagrammen is een beschrijving opgesteld. Deze beschrijving geeft naast de globale werking ook de uitzonderingen en het bijbehorende object aan.

Het implementeren van deze functies werd na het opstellen van deze beschrijvingen vergemakkelijkt. Na de implementatie zijn de testcases uitgevoerd. Er was besloten om

deze testcases niet per functie op te stellen, maar per use-case in het use-case diagram. De functionaliteiten die het subsysteem namelijk moest gaan ondersteunen waren de use-cases die in de definitiestudie waren opgesteld. Indien een van deze functionaliteiten niet aan de verwachte uitvoer zou leveren, kon alsnog gedetailleerder gekeken worden naar de opgestelde functies.
De resultaten van de tests waren positief. Geen enkele functionaliteit had een onverwachte uitvoer als resultaat. Na het registreren van deze resultaten, kon deze bouweenheid worden afgerond.

7.5.7. Ontwikkelen functies subsysteem Zwangerschap

In deze bouweenheid zijn op dezelfde manier als in de vorige bouweenheid de functies uit de sequencediagrammen beschreven. De functies zijn beschrijven en ontwikkeld zoals ze in het diagram van de functionele specificaties opgenomen waren. Voor de functionaliteiten *Babygegevens invoeren* en *Kraambedgegevens invoeren* zijn de sequencediagrammen niet ontwikkeld. De reden hiervoor was omdat dit geen extra inzicht in het proces gaf. Bovendien waren deze vergelijkbaar met de functionaliteiten waarvoor wel een diagram was gemaakt.
Tijdens de ontwikkeling van de functies en procedures van de bouweenheid waren verder geen problemen geconstateerd; de bouw verliep vlot

Net als in de vorige bouweenheid waren de testsets opgesteld per functionaliteit. Na uitvoeren van de tests konden wederom in de resultaten geen fouten worden geconstateerd. De verwachte uitvoer was bij iedere test ook de verkregen uitvoer.

7.5.8. Ontwikkelen functies subsysteem Administratie

De 8ste bouweenheid moest het derde subsysteem gaan implementeren. Aangezien de functionaliteit *LVR-Controle* was komen te vervallen, was *Rapporteren* de enige die in dit subsysteem zou worden ontwikkeld. De inhoud van deze functionaliteit was in de functionele specificaties in de vorm van een sequencediagram reeds toegelicht.

De functie rapporteren moest een lijst opleveren met cliënten die binnen een opgegeven termijn nog moeten bevallen. Door het bouwen van de functies *aantalZwangeren* en *geefClienten* kon dit worden gerealiseerd. Na de bouw van deze functies en de aanroep hiervan, kon worden overgegaan tot het testen van de bouweenheid.

Uit de resultaten van tests bleek dat de bouweenheid aan de verwachtingen voldeed.

7.5.9. Opstellen gebruikershandleiding subsystemen

Tot slot was in het kader van deze pilot de bouweenheid *opstellen gebruikershandleiding subsystemen* een belangrijke. Deze bouweenheid moest een gebruikershandleiding als resultaat opleveren in de vorm van een quick-reference guide.
In overleg met de opdrachtgever was besloten om geen uitgebreide handleiding te schrijven, maar één quick-reference guide die makkelijker te hanteren was en snel te raadplegen.

Om de handleiding op te stellen die voor de verloskundigen praktisch was, was in overleg bepaald wat de minimale inhoud moest zijn. In dit overleg was bepaald dat de handleiding aan de volgende eisen moest voldoen:

1. Bevat een overzicht van alle functies van de applicatie
2. Beschrijft de locatie en het doel van iedere functie
3. De schermen zijn opgenomen met een combinatie van een grafische en tekstuele toelichting
4. Een inhoudsopgave en een symbolenlijst is aanwezig.

Het opstellen van een gebruikershandleiding was niet nieuw voor mij, aangezien dit in mijn stageperiode ook aan de orde kwam. De gebruikershandleiding uit die periode is als basis gebruikt voor het schrijven van deze *quick-reference guide*.
Nadat deze was opgesteld, is deze getest door deze door de verloskundigen te laten lezen. Zij zouden in staat moeten zijn, om met behulp van deze handleiding de eindapplicatie te gebruiken.
Als gevolg van deze praktische test, bleek dit uitstekend te werken. Naast dat de verloskundigen tevreden waren, leken zij ook meer geïnteresseerd in het eindresultaat.
Het opstellen van deze handleiding was tevens de laatste bouweenheid van de pilot.

7.6. Invoeringsprocedure

De activiteit die ik heb uitgevoerd na het bouwen testen van de bouweenheden was het opstellen van een invoeringsprocedure. Hoewel de geïmplementeerde bouweenheden onderdeel waren van de applicatie, moest de invoeringsprocedure worden geschreven voor de applicatie. Deze activiteit had om die reden overgeslagen kunnen worden, ware het niet dat er een afspraak was gemaakt dat er in het nieuwe jaar een netwerk aangelegd zou zijn.
Deze invoeringsprocedure is praktisch identiek aan die van de vorige pilot, met uitzondering van de controle op het netwerk.

1. Controleren operationele software
   Software:
   - MySQL
   - MyODBC

2. Controleren of er een netwerk is aangelegd, zodat de applicatie correct kan opereren in de huidige omgeving.

3. Invoeren nieuwe applicatie met operationele subsystemen

Hierna heb ik de testspecificaties opgesteld die ondersteuning kunnen bieden aan de invoering.

7.7. Testen en beoordelen

In deze activiteit heb ik in samenwerking met de opdrachtgever de pilot beoordeeld. Nadat de testresultaten van de bouweenheden waren beschreven, kon op basis daarvan een beoordeling van de pilot worden opgesteld. Doordat dit tevens de laatste pilot in het ontwikkeltraject betreft, was tevens getest of was voldaan aan de gestelde functionele systeemeisen. Het resultaat van deze tests zijn hieronder opgenomen:

Code	Systeemeis	Voldaan
M1	Het systeem moet de gegevens op de oude zwangerschapskaart kunnen realiseren.	Ja
M2	De gegevens van cliënten die verwerkt worden in het systeem moeten grotendeels variabel zijn. (Doktoren, ziekenhuizen, ziektes, etc)	Ja
M3	Het moet mogelijk zijn om te zoeken naar de cliënten op eventueel unieke gegevens	Ja
M4	Het moet mogelijk zijn om ziekenhuizen bij te werken en er in te zoeken	Nee, onderdeel subsysteem 4
M5	Ziekenhuizen, verzekeraars, gynaecologen, doktoren, plaatsnamen, achternamen en voornamen. Deze gegevens moeten allemaal in een ‘zelfgroeiende’ lijst kunnen worden geselecteerd, zodra het systeem vereist dat deze worden ingevoerd.	Ja
S1	Het systeem zal middels een functie de cliëntdata moeten kunnen exporteren naar een locatie buiten het systeem.	Nee, onderdeel subsysteem 4
S2	De gebruiker en/of beheerder kan alleen toegang krijgen tot het systeem middels een gebruikersnaam en wachtwoord	Ja
S3	Er moet een mogelijkheid zijn om de gegevens te rapporteren	Ja
C1	De LVR (Landelijke Verloskunde Registratie) vereist dat er periodiek een formulier wordt toegezonden met het aantal cliënten en extra gegevens, deze moet door het systeem kunnen worden gegenereerd.	Nee, deze functie is komen te vervallen
C2	De instellingen binnen de applicatie moeten kunnen worden veranderd door een beheerder	Nee, onderdeel van subsysteem 4
C3	Gebruikers van het systeem moeten kunnen worden toegevoegd	Nee, onderdeel van subsysteem 4

Afbeelding 37: Resultaten dekking systeemeisen na pilot 4.

In dit stadium is het mogelijk om de dagelijkse handelingen die een verloskundige verricht te ondersteunen. De systeemeisen die niet behaald zijn in dit stadium, met uitzondering van C1, worden gedekt door het vierde en tevens laatste subsysteem.

De systeemeis C1 is komen te vervallen na de beslissing om het laatste subsysteem niet te ontwikkelen in het kader van dit project.

Ondanks dat binnen het kader van dit project het vierde subsysteem niet zal worden uitgevoerd, kan door het niet behalen van de systeemeis C1 deze pilot als onsuccesvol afgerond worden beschouwd.

Wellicht zal in een later stadium, die buiten het kader van dit project valt, de vierde pilot alsnog worden uitgevoerd. Waarna het mogelijk wordt om voor de systeemeis C1 alsnog de nodige functionaliteiten te ontwikkelen.

7.8. Afronding

Na het controleren van de opmaak, is het rapport afgerond. Het definitieve rapport is opgenomen als bijlage van dit verslag.

Door de uitloop van de vorige pilots, was de planning voor deze pilot ruimer opgesteld. Bovendien was besloten om de laatste pilot niet uit te voeren in het kader van dit project. Deze beslissing had ervoor gezorgd dat de uitloop van zowel de vorige als deze pilots kon worden afgevangen.

Ter afsluiting van het project heb ik een eindpresentatie gegeven over de applicatie en hierin aangegeven hoe de eisen in de workshop verwerkt waren in het eindresultaat. Na opleveren van de applicatie en de quick-reference guide, is de afstudeerperiode afgerond.

8. Evaluatie

In dit hoofdstuk wordt het uitgevoerde project geëvalueerd. In paragraaf 8.1 wordt het verloop van elke fase uit het project beoordeeld. Tot slot worden de producten in paragraaf 8.2 de opgeleverde mijlpaalproducten geëvalueerd.

8.1. Procesevaluatie

8.1.1. Opstellen plan van aanpak

Tijdens de opleiding informatica en informatiekunde aan de Haagse Hogeschool heb ik voor ieder project een plan van aanpak op moeten stellen. De ervaring die ik hierbij heb opgedaan kwam bij het opstellen van dit aanpak goed van pas. In de module, voorbereidend op het afstuderen, was besloten om zo compleet als mogelijk plan van aanpak op te stellen. In die module was daarom gekozen voor een ander soort plan van aanpak dan op de Haagse Hogeschool was geleerd. De afwegingen die toen zijn gemaakt, heb ik voor het plan van aanpak van dit project ook toegepast. Dit bleek een goed steunpunt te zijn om binnen de geplande tijd een voorlopig plan van aanpak op te leveren. Het voorlopige plan van aanpak zou tijdens het project worden aangevuld in de planning, zodra dit nodig was. Het opstellen van het plan van aanpak heeft verder geen problemen opgeleverd en kon binnen de geplande tijd worden opgeleverd. In een volgend project zou ik op dezelfde manier een plan van aanpak opstellen, zoals ik voor dit project heb gedaan.

8.1.2. Fase definitiestudie

Voor de aanvang van het ontwikkeltraject van IAD, had ik allereerst het verloskundeproces onderzocht. Om in te kunnen schatten hoe een verloskundige te werk gaat en tegen welke problemen ze kunnen oplopen leek dit mij een verstandige beslissing. Vervolgens heb ik onderzoek verricht naar de activiteiten die in het kader van de definitiestudie uitgevoerd konden worden. Hieruit bleek dat “Evalueren Pilot”, waarvan ik deze relevant achtte, helemaal niet relevant was. Hierdoor kon de geplande tijd voor de definitiestudie met acht uur worden ingekort.
Het uitvoeren van de workshops kostte echter veel tijd. Door gebrek aan ervaring met een workshop en het gebrek aan beschikbare tijd van de werknemers nam deze activiteit meer tijd in beslag dan gepland. Daarnaast was ook het opzetten van het klassendiagram en het use-case diagram een tijdrovende activiteit. De ervaring had mij geleerd dat het opstellen van een dergelijk diagram niet heel veel tijd in beslag moest nemen. Ik had echter nog geen ervaring met een diagram van deze omvang.

Tijdens de fase definitiestudie heb ik geleerd hoe een definitiestudie in een ontwikkeltraject van IAD wordt uitgevoerd. De definitiestudie die bij mij bekend was, was die van de methode SDM. Het doel van deze fase was vergelijkbaar, hoewel het vervolgtraject heel erg verschilt.

Ik ben vrij tevreden over de wijze waarop ik de definitiestudie heb uitgevoerd. Ik zou voor deze fase meer tijd in plannen, om meer detail in de definitiefase aan te kunnen

brengen. In een volgend project waar de ontwikkelmethode IAD wordt gehanteerd, zou ik deze fase uitvoeren als er meer tijd ingepland zou zijn.

8.1.3. Pilotontwikkeling eerste pilot

Voor de aanvang van het pilotontwikkeltraject van de eerste pilot heb ik wederom onderzoek verricht naar de activiteiten die volgens IAD kunnen worden verricht in deze fase. Uit dit onderzoek bleek dat een aantal activiteiten die ik eerder had ingepland niet relevant bleken.
Na bestudering van de activiteiten kon ik de planning voor de fase pilotontwikkeling beter inschatten doordat ik meer inzicht had in de duur van de activiteiten.

Het onderzoek dat ik heb verricht naar hoe de technieken van UML konden worden toegepast bij het ontwerpen van de functionele en technische structuur van de eerste pilot, kostte veel tijd. Uiteindelijk heb ik de technieken van UML slechts beperkt toegepast bij het ontwerp, in de vorm van activiteitendiagrammen. Het grotendeels tekstuele ontwerp van de bouweenheden voldeed echter wel om de pilot te kunnen bouwen.

In deze fase heb ik geleerd hoe binnen IAD een pilotontwikkeling moest worden uitgevoerd. Naast de leereffecten van IAD in combinatie met UML, heb ik in deze fase veel geleerd over het ontwerpen van Grafische Interfaces in Delphi. Door de ervaring met programmeertalen binnen de opleiding van informatica en informatiekunde, bleek het aanleren van een relatief onbekende taal geen enkel probleem.
Ik ben tevreden over de wijze waarop ik de fase pilotontwikkeling voor de eerste pilot heb uitgevoerd. In een volgend project waar de ontwikkelmethode IAD wordt gebruikt, zou ik deze fase op dezelfde manier uitvoeren.

8.1.4. Pilotontwikkeling tweede pilot

Met de ervaring die ik had opgedaan tijdens de ontwikkeling van de eerste pilot, kon de tweede pilot sneller worden uitgevoerd. Ik had meer inzicht gekregen in de inhoud van de activiteiten en de benodigde tijdsduur. Hierdoor was het mogelijk om de activiteiten beter te plannen. De technieken van UML heb ik bij het ontwerp van deze pilot beperkt gebruikt. De tabellen in de database waren naar mijn mening al voldoende gemodelleerd in het klassendiagram.
De ervaring met het opstellen van een database zoals aangeleerd in de opleiding Informatica en Informatiekunde, kon hier goed worden toegepast. Mede op basis van het diagram en de ervaring konden de bouweenheden worden opgesteld.
Het weglaten van de diagrammen van UML heeft achteraf geen problemen opgeleverd.

Naast de leereffecten op het gebied van IAD en UML, heb ik in deze fase veel geleerd over het omzetten van een klassendiagram naar een database. Dit proces bleek zeer complex en tijdrovend, maar tegelijkertijd zeer leerzaam.

Ik ben tevreden over de wijze waarop ik de fase pilotontwikkeling voor de tweede pilot heb uitgevoerd. In een volgend project waar de ontwikkelmethode IAD wordt gebruikt, zou ik deze fase op dezelfde manier uitvoeren.

8.1.5. Pilotontwikkeling derde pilot

De opgedane ervaring van de vorige twee pilots resulteerde in een vlotte ontwikkeling van de uit te voeren activiteiten. De technieken van UML heb ik bij het ontwerp van deze pilot wederom beperkt gebruikt. In een poging om de functionaliteiten te ontwerpen in een sequencediagram bleek vrij snel overbodig. De toegevoegde waarde van het sequencediagram was in deze pilot nihil. Het besluit om de sequencediagrammen te ontwikkelen voor use-cases in de laatste pilot leek daarom de beste oplossing.
De functionaliteiten zijn vervolgens ontworpen aan de hand van precondities en postcondities. Mede op basis van de functionele specificaties zijn de bouweenheden vastgesteld. De beslissing om de sequencediagrammen in de volgende pilot te ontwikkelen bleek achteraf geen verdere problemen te veroorzaken.

Naast de leereffecten op het gebied van IAD en UML, heb ik in deze fase veel geleerd over Delphi. Bijvoorbeeld hoe een database vanuit een applicatie met behulp van ODBC kon worden benaderd en hoe de resultaten van een query konden worden verwerkt. De combinatie van Delphi en MySQL bleek een goedkope en krachtige combinatie te zijn.

Tijdens de pilotontwikkelfase van de derde pilot was anderhalve week meer tijd nodig dan gepland. De extra benodigde tijd was vooral te wijten aan de problemen die ik bij de bouw van bouweenheid 3 tegenkwam. Het probleem van het programmeren in een taal als Delphi is dat er niet wordt gecontroleerd of een object al bestaat als deze wordt aangesproken. Dit leidt soms tot vreemde en complexe foutmeldingen. De applicatie bevat volgens de compiler dan geen fouten en deze zijn ook niet met de bijgeleverde debugger op te sporen. Na het oplossen van dit probleem was besloten om alle objecten die vanaf nu opgesteld moesten worden, voor de invoering in de applicatie eerst te compileren in een aparte testunit werden.

Ik ben redelijk tevreden over de wijze waarop ik de fase pilotontwikkeling voor de derde pilot heb uitgevoerd. In deze fase heb ik geleerd dat het verstandig is om nieuwe objecten eerst te testen in een aparte testunit voordat deze worden toegevoegd aan de applicatie.

8.1.6. Pilotontwikkeling vierde pilot

Met de ervaring die ik had opgedaan tijdens de fase pilotontwikkeling voor de eerdere pilots, kon ik de fase pilotontwikkeling voor de vierde pilot in enkele opzichten sneller uitvoeren. Hierdoor kon ik de activiteiten voor deze fase beter plannen.
De technieken van UML heb ik bij het ontwerp van deze pilot toegepast in de vorm van sequencediagrammen. Voor iedere functionaliteit is een sequencediagram opgesteld om meer inzicht te krijgen in de diepgang. Mede op basis van dit ontwerp kon ik de bouweenheden implementeren. Tijdens het ontwerp van de bouweenheden was gebleken dat het bouwen van de bouweenheden veel tijd kostte. Vooral het ontwerp en de

implementatie van de globale functies en MD5 encryptie nam meer tijd in beslag dan vooraf was ingepland.
Tijdens de definitiestudie was ervan uitgegaan dat het gebruik van de MD5-encryptie een unit zou zijn, die voor Delphi gratis beschikbaar zou zijn. Na het zoeken op internet bleken echter dat de meeste units een bepaald limiet hadden, zoals een tijdslimiet of alleen te gebruiken voor informatieve doeleinden.

Naast de leereffecten op het gebied van IAD en UML, heb ik in deze fase op het gebied van Delphi het meeste geleerd. Bijvoorbeeld de implementatie van RFC en het gebruik van complexe functionaliteiten binnen Delphi. Daarnaast heb ik ook meer geleerd over encryptie en MD5-encryptie in het algemeen.

Voor de pilotontwikkelfase was voldoende tijd gepland, door het laten vervallen van de vijfde pilot. Door de toepassing van de juiste diagrammen in UML en het ontwerp van de functionaliteiten vormden de bouw van deze pilot geen enkel probleem. Op enkele programmeerfouten nagelaten, was de uitvoering van de activiteiten in het kader van pilotontwikkeling succesvol te noemen.

Ik ben tevreden over de wijze waarop ik de fase pilotontwikkeling voor de vierde pilot heb uitgevoerd. In een volgend project waar de ontwikkelmethode IAD wordt gebruikt, zou ik deze fase op dezelfde manier uitvoeren.

8.2. Productevaluatie

8.2.1. Plan van aanpak

Het doel van het plan van aanpak, overeenstemming bereiken over de opdracht en over de uitvoering daarvan, is bereikt. Na oplevering van de definitieve versie van het document is in overleg met de opdrachtgever een akkoord bereikt over de inhoud ervan.

Het document is op meerdere punten tijdens het project nuttig gebleken. De opgestelde planning bood houvast om het project goed te kunnen beheersen. De schattingen van de benodigde tijd per fase bleken in de praktijk in sommige gevallen correct, maar in veel gevallen niet. Het gebrek aan ervaring met de methode zal daaraan bijgedragen hebben. Daarnaast heeft een vrij uitgebreide beschrijving van de standaarden en richtlijnen met betrekking tot de rapportopmaak bijgedragen aan een consistente opmaak van de opgeleverde documentatie.

Door het toepassen van het risicomanagement had ik besloten om alle documenten op drie fysiek verschillende locaties op te slaan. Het tweede risico vond in de vierde week plaats. De harde schijf waarop de rapporten waren opgeslagen was onaangekondigd op onherstelbare wijze beschadigd. De keuze om de recente documenten op drie verschillende locaties op te slaan, was verstandig gebleken.
In het risicomanagement waren tevens de stappen opgenomen die moesten worden ondernomen indien het project niet binnen de gestelde tijd kon worden afgerond.

Omdat het plan van aanpak tijdens dit project zijn waarde heeft bewezen, zou ik dit plan van aanpak als uitgangspunt kunnen gebruiken voor het opstellen van een volgend plan van aanpak.

8.2.2. Rapport definitiestudie

Het rapport definitiestudie bevat de resultaten van de activiteiten die zijn uitgevoerd tijdens de fase definitiestudie. Het doel van het rapport was om de eisen aan de te ontwikkelen applicatie vast te stellen en een basis te vormen voor de ontwikkeling van deze applicatie. Het doel van het rapport is bereikt aangezien het rapport tenminste de volgende onderdelen bevat:

- Een systeemconcept met daarin de eisen die gesteld zijn aan de applicatie
- Een pilotplan waarin de basis voor de ontwikkeling is gevormd.

8.2.3. Eerste Pilotontwikkelrapport

Het eerste pilotontwikkelrapport had als doel om ondersteuning te bieden aan de bouw van de Grafische User Interface voor de applicatie. Dit doel is bereikt doordat het rapport ondermeer de volgende onderdelen bevat:

- Bouweenheden op basis waarvan de bouw kan worden uitgevoerd
- Testspecificaties waarmee het gewenste resultaat van iedere bouweenheid kan worden gegarandeerd.

8.2.4. Tweede Pilotontwikkelrapport

Het tweede pilotontwikkelrapport heeft als doel om ondersteuning te bieden aan de bouw van de database. De opbouw van het tweede ontwikkelrapport is hetzelfde als die van de eerste pilot en heeft daarom ook het gestelde doel bereikt. De eisen, gesteld in het plan van aanpak, aan de opmaak van het rapport voldoen ook.

8.2.5. Derde Pilotontwikkelrapport

Het doel van het derde pilotontwikkelrapport is ondersteuning bieden aan de bouw van de GUI-database interface. De opbouw en de eisen aan de opmaak zijn hetzelfde als die van de voorgaande pilots en daaraan wordt voldaan. Het doel van dit rapport is daarom bereikt.

8.2.6. Vierde Pilotontwikkelrapport

Het laatste ontwikkelrapport heeft als doel ondersteuning te bieden aan de bouw van de subsystemen één tot en met drie. Het rapport was in opbouw hetzelfde als de voorgaande pilots en heeft daarom ook het gestelde doel bereikt. De eisen aan de opmaak van het rapport voldoen ook.

8.2.7. Gebruikershandleiding

De opgeleverde quick-reference guide voldoet aan de gestelde eisen, zoals die zijn overeengekomen met de opdrachtgever. Bovendien is dit document na het opstellen

ingezien en beoordeeld door de verloskundigen. Het resultaat van dit product is daarom als succesvol te beschouwen aangezien het doel is bereikt.

8.2.8. Applicatie

De applicatie ‘W!se Delivery’ voldoet niet aan alle gestelde systeemeisen. Door de beslissing om de vijfde pilot niet uit te voeren, zijn de functionaliteiten van de beheerder niet aanwezig. De functionele systeemeisen die betrekking hebben op het gedeelte dat wel ontwikkeld is, voldoet. Aan de hand van de tests die zijn uitgevoerd in de laatste pilot, is gebleken dat alle use-cases beschreven in de definitiestudie functioneel zijn.

De oplevering van de applicatie kan als succesvol worden beschouwd, gelet op de functionele systeemeisen die binnen de ontwikkelde pilots worden gedekt. Indien de volledigheid van de applicatie wordt bekeken dan is het resultaat onsuccesvol.

Literatuurlijst

Hieronder is de lijst op alfabetische volgorde opgesteld van de geraadpleegde literatuur.

- Binzinger, Thomas, Snel leren programmeren Delphi 7, 1e druk, 2003.
- Ghezzi, C., e.a., Fundamentals of Software Engineering, Prentice-Hall, januari 1991
- Online documentatie over 'Coding Standards' van Orange County Delphi Users Group, http://www.ocdelphi.org/standard.htm
- Online medisch woordenboek, http://www.ziekenhuis.nl/index.php?cat=woordenboek
- Online onderzoek over 'Coding Standards in PHP' van het Leuvens Universitair Dienstencentrum voor Informatica en Telematica, http://ludit.kuleuven.be/
- Reader AV-03 Haagse Hogeschool, nr. 908: Rapportagetechniek
- Tolido, R.J.H., IAD - Het evolutionair ontwikkelen van informatiesystemen, Academic Service, 6e oplage., 2003;
- Warmer, J., en Kleppe A., Praktisch UML, Addison Wesley, 2e editie, 2001.

Bijlagen

Vanwege de grootte van de bijlagen, zijn deze opgenomen in de bijgesloten bundel. Deze bundel bevat de volgende bijlagen:

1. Plan van aanpak gehele project
2. Rapport definitiestudie
3. Pilotontwikkelrapport pilot 1: Grafische User Interface
4. Pilotontwikkelrapport pilot 2: Database
5. Pilotontwikkelrapport pilot 3: GUI-Database Interface
6. Pilotontwikkelrapport pilot 4: Subsystemen één tot en met drie.

Bijlagen
Project Verloskundeapplicatie

Datum	14 januari 2004
Student / Auteur	C. Kruijf, 99008079
Afstudeerperiode	30 augustus 2004 – 14 januari 2004
Opleiding	Haagse Hogeschool Informatica en Informatiekunde
Studiepad	Ontwikkeling van Software en Technische Infrastructuren (OSTI)
Organisatie	Verloskundeprakijk de Maatschap Lammenschans Bloemistenlaan 45/A 2313BB Leiden Tel: 071 5124040
Bedrijfsmentor	Mevr. van Iterson
Examinatoren	Dhr. N.J.J. Groot Dhr. F. Bögels

Voorwoord

Voor u ligt de bundel die de bijlagen bevat die in het kader van de afstudeeropdracht *Verloskundeapplicatie* zijn ontwikkeld. Deze bijlagen zijn onderdeel van het eindverslag dat dient te worden geschreven ter afronding van de opleiding Informatica en Informatiekunde aan de Haagse Hogeschool. De opdracht is uitgevoerd bij de verloskundepraktijk de Maatschap te Leiden.

De bijlagen bestaan uit de documentatie die ik heb opgeleverd gedurende de uitvoering van de afstudeeropdracht.

Leiden, 14 januari 2005

Cees Kruijf
Verloskundepraktijk de Maatschap

Inhoudsopgave

Bijlage 1
Plan van aanpak gehele project

Voorwoord

Voor u ligt het plan van aanpak van het verloskunde afstudeerproject. Dit rapport is bestemd voor de opdrachtnemer en betrokkenen. Dit rapport zal tijdens het project gebruikt worden om de uit te voeren processen op de juiste wijze plaats te laten vinden. Tevens worden de eisen en wensen van het project en het proces in kaart gebracht. Het is aan te raden voor alle betrokkenen dit rapport aandachtig door te lezen. Wijzigingen kunnen worden aangeleverd aan de documentbeheerder in een toevoeging op het plan van aanpak.

Cees Kruijf
Den Haag, 06-09-2004

Inhoudsopgave

1. Introductie

Het plan van aanpak wordt gebruikt om het ontwikkelen van een informatiesysteem in goede banen te leiden. Als eerste wordt de aanleiding van het project gegeven. Vervolgens wordt de wijze van accorderen van dit plan van aanpak beschreven. In paragraaf 1.3 vindt een beschrijving van de rapportstructuur plaats.

1.1. Aanleiding

De verloskundepraktijk Maatschap Lammenschans, werkt al tijden met een applicatie die de primaire en secundaire processen ondersteuning verleend. Waaronder het registreren van cliënten, afhandelen van betalingen, versiebeheer en de complicaties die kunnen optreden bij de cliënten en/of zwangerschap. De bestaande applicatie genaamd Orfeus, voldoet niet meer aan de eisen van de praktijk. Foutmeldingen zijn ‘slordig’ en onvolledig en het gebruik van de applicatie laat vaak te wensen over.

1.2. Accorderen en bijstelling

Na het opstellen van het plan van aanpak, zal deze worden gevalideerd door de opdrachtnemer. Dit zal gebeuren door te kijken of eventuele zaken niet zijn vergeten en of op basis van dit plan van aanpak het project tot een succes zou kunnen worden uitgevoerd. Na eventuele aanpassingen zal het plan van aanpak niet meer worden gewijzigd, echter is het onderdeel ‘de planning’ hier een uitzondering op.

1.3. Toelichting op de opbouw

Allereerst wordt er een basis gelegd voor de opdracht. Onderwerpen als opdrachtomschrijving en op te leveren producten worden hier besproken. In hoofdstuk 3 wordt de aanpak besproken die zal leiden tot een goede uitwerking van de besproken opdracht. Hoofdstuk 4 beschrijft de indeling van het project en zullen de taken van de betrokkenen worden toegelicht. De globale en gedetailleerde planning zijn opgenomen in hoofdstuk 5. Er zal worden afgesloten met de wijze waarop de kwaliteit wordt gewaarborgd van het project, hieronder valt ook het risicomanagement.

2. Projectopdracht

In overleg met de opdrachtgever wordt het project afgebakend. Als eerste wordt verklaard waar en waarom het project is opgestart. Vervolgens wordt de doelstelling van het project bepaald. Uit deze doelstelling komt een opdrachtformulering voort. Vervolgens wordt beschreven wat de opdrachtgever wenst te ontvangen om aan de hand van vastgestelde eisen en beperkingen de uitvoering van de opdracht kan beoordelen. Vervolgens wordt bekeken waar de opdrachtgever de nadruk van de opdracht legt.

2.1. Projectomgeving

Er is reeds een informatiesysteem om onder andere het registeren van cliënten en afhandelen van de betalingen mogelijk te maken. Voor de praktijk is dit informatiesysteem van essentieel belang om de primaire taken goed uit te kunnen voeren. Binnen de praktijk is geen technische ondersteuning aanwezig.

2.2. Doelstelling

De doelstelling van de opdracht is het ontwerpen en realiseren van de verloskunde applicatie.

2.3. Nadrukken

De verloskundeapplicatie zal voldoen aan de systeemeisen die in de definitiestudie vermeld zijn. Deze eisen zullen op basis van de eisen van de opdrachtgever en uit het reverse engineeren van de bestaande applicatie moeten blijken.

2.4. Opdrachtformulering

Er wordt een informatiesysteem ontworpen en ontwikkeld ter ondersteuning van het primaire proces. Elk aspect van begin tot eind in het cliënttraject zal worden ondersteund door het informatiesysteem.
Aangezien er binnen de organisatie geen ICT-deskundigen aanwezig zijn, zal de gebruikersvriendelijkheid hoog moeten zijn. Ook het beheer en de documentatie zullen uitgebreid en begrijpelijk moeten zijn.

2.5. Op te leveren producten

De producten die opgeleverd gaan worden, zijn de mijlpaalproducten zoals die in de fasering van IAD ter sprake komen, zoals hieronder weergegeven:

- Plan van aanpak
- Rapport Definitiefase
- Ontwikkelde pilots
- Documentatie
- Informatiesysteem

Naast deze op te leveren producten, zal ook een verslag van het reverse engineeren van de huidige applicatie en de interviews met de opdrachtgever worden opgeleverd.

- Verslag reverse engineering
- Samenvatting interview(s)

2.6. Eisen en beperkingen

Het eindproduct zal door de opdrachtgever worden beoordeeld op de aanwezigheid van de in de doelstellingen en opdracht geformuleerde eisen. Tussentijds zal de opdrachtgever op de hoogte gehouden worden van de voortgang van het project. Het overleggen van tussentijdse producten zal plaatsvinden op de data die in de gedetailleerde planning zijn opgenomen.
Tevens wordt de opdracht beoordeeld op de aanwezigheid van ondersteunende documentatie en de begrijpbaarheid en werkbaarheid van het systeem. Het systeem moet betrouwbaar en voldoende beschikbaar zijn.

2.7. Cruciale succesfactoren

De Cruciale succesfactoren geven aan, welke door de opdrachtnemer beïnvloedbare zaken er vanuit de opdrachtgever gezien essentieel zijn om het resultaat zo goed mogelijk te laten aansluiten bij de te bereiken doelstelling.

Voor dit project gelden de volgende factoren:

- Het informatiesysteem moet voorzien in de behoefte van de opdrachtgever.
- Het informatiesysteem zal de huidige applicatie moeten kunnen vervangen
- De gebruikershandleiding moet concreet en begrijpelijk zijn.
- Het systeem moeten worden opgeleverd binnen de afstudeerperiode die door de Haagse hogeschool is gesteld.

3. Aanpak

Er moet een gestructureerde aanpak worden gevolgd om tot een juiste uitwerking van de doelstelling op de juiste manier om te zetten in een oplossing.

3.1. Gekozen Methode

Voor dit project is daarom gekozen voor de ontwikkelmethode IAD. Deze methode is zeer geschikt voor het ontwikkelen van software waar de behoefte aan een snelle oplevering groot is. Verder zullen de technieken van Yourdon worden gebruikt in samenwerking met deze methode.

3.2. Fasering

Iedere methode heeft een bepaald aantal fasen, hieronder is de IAD-fasering uitgezet.

Fase 1: Definitiestudie

- Stel een plan van aanpak op
- Definieer ontwikkelscenario
- Bereid pilotplan-workshop voor
- Evalueer pilot
- Definieer systeemeisen
- Bepaal systeemconcept
- Beschouw technische structuur
- Beschouw organisatorische inrichting
- Stel pilotplan op

Fase 2: Pilotontwikkeling

- Stel plan van aanpak op
- Bereid pilotontwerp-workshop voor
- Specificeer globaal-functionele structuur pilot
- Specificeer globaal-technische structuur pilot
- Specificeer globaal-organisatorische inrichting
- Stel pilotontwikkelplan op
- Ontwerp software-bouweenheden
- Bouw software-bouweenheden
- Pas externe componenten aan
- Wijzig andere informatiesystemen
- Ontwerp handmatige procedures pilot
- Stel opleidingsmateriaal pilot samen
- Stel handleidingen pilot samen
- Stel invoeringsprocedures pilot op
- Integreer bouweenheden
- Beoordeel en test pilotdeel

Fase 3:

- Stel plan aanpak op
- Voer pilot in
- Verzorg opleidingen
- Voer pilotacceptatie uit
- Draag pilot over
- Verzamel feedback
- Ondersteun pilot

Elke fase wordt voorafgegaan door een aanvulling op het plan van aanpak. In deze aanvulling wordt de detailplanning waar nodig aangepast/aangevuld voor de betreffende fase.

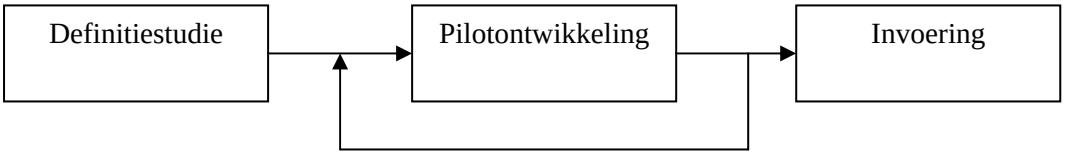
3.3. Ontwikkelstrategieën

Binnen de IAD-methode kan worden gekozen uit de vier hoofd ontwikkelstrategieën:

- Evolutionair ontwikkelen
- Incrementeel opleveren
- Incrementeel ontwikkelen (‘RAD’)
- Big-bang-invoeren

Er is voor gekozen om de definitiestudie en projectinvoering eenmalig en de projectontwikkeling iteratief uit te voeren. Het project dient binnen een tijdsbestek van 600 uur te worden opgeleverd vandaar dat er is gekozen voor een eenmalige pilotinvoering.

Bij IAD is bij een pilotinvoering sprake van het operationeel maken van een pilot, om vervolgens na een aantal weken feedback te ontvangen over deze pilot. Aangezien één van de randvoorwaarden aan het project is dat het binnen 600 uur vervuld moet zijn, is er geen tijd om te ‘wachten’ op feedback.
Om toch een systeem op te leveren dat voldoet aan de eisen en verwachting van de opdrachtgever, zal hier tijdens de ontwikkeling om meer feedback gevraagd worden. Daarom is er gekozen voor het iteratief ontwikkelen, de RAD-methode.



4. Projectinrichting en voorwaarden

4.1. Projectinrichting

Projectmanager: Dhr. C. Kruijf
Technische ondersteuning: Dhr A. Nederend
Opdrachtgever: Mevr. S. van Iterson.

IAD onderscheid in de methode drie verschillende teams, het A-team, het U-team en QA-team.
Het A-team is het team waarin de systeemontwikkelaars zijn ondergebracht. In een groot project bestaat zo'n team uit ongeveer vier à vijf personen, in dit geval één, namelijk de Projectmanager.
Het U-team is het team van 'users' (toekomstige gebruikers, ook wel G-Team genaamd). In dit geval is dat de opdrachtgever en eventueel de collega's van de opdrachtgever. Dit team zal worden betrokken bij de aankomende workshops en de benodigde feedback op de applicatie.
Dan is er nog een adviserend orgaan (het QA-Team) binnen het project aanwezig dat binnen IAD de standaarden zou moeten bewaken en in een groot project nog een aantal taken heeft. In dit geval zal het QA-Team bestaan uit Arno Nederend en alleen de functie van technische ondersteuning vervullen.

4.2. Voorwaarden aan de opdrachtnemer

De opdracht dient te worden uitgevoerd in een tijdsbestek van 15 weken. Hierbij inbegrepen is de documentatie die geschreven zal moeten worden voor het gebruik van de applicatie. Afgerond zal worden met het geven van een eindpresentatie.

4.3. Voorwaarden aan de opdrachtgever

De opdrachtgever stelt zich beschikbaar voor het leveren van commentaar. Dit commentaar zal voorzover mogelijk 'life' gebeuren.
De opdrachtgever zal de ontvangst van de ingediende tussenproducten binnen drie dagen na ontvangst bevestigen en geeft hierbij de goedkeuring voor voorzetting van het project of afkeuring van de geleverde stukken met commentaar ter verbetering.
In de eerste twee weken van het project zal de opdrachtgever zich drie maal een uur beschikbaar stellen voor het beantwoorden van vragen met betrekking tot de te verwerken informatie. De opdrachtgever kent geen verantwoordelijkheid richting de Haagse Hogeschool.

4.4. Voorwaarden aan derden

De aangewezen begeleidend docent stelt zich beschikbaar voor het beantwoorden van technisch inhoudelijke vragen. De begeleidende docent zal indien nodig ingezonden stukken doorlezen en becommentariëren.

5. Plannen

Voor de voortgangscontrole wordt een globale planning en een gedetailleerde planning gemaakt. De globale planning is vooral opgesteld om de opdrachtgever een indruk te geven, wanneer een tussenproduct zal worden opgeleverd. De gedetailleerde planning is een richtlijn voor de opdrachtnemer.

5.1. Activiteitenplan

De globale planning geeft de planning ingedeeld in weken. Per week besteed elk projectlid minimaal 40 uur.

Mijlpaalproduct	Weken*
Plan van aanpak	1
Definitiestudie	2
• Reverse Engineering	0.5
Pilotontwikkeling	9
Pilotinvoering	2
Afronden documentatie	0.5

* tienden in de weken worden als volgt berekend: U = 40x. Hierbij is x het aantal weken en U het aantal uren.

5.2. Detailplanning

De detailplanning is ingedeeld in uren, met de activiteiten / producten uitgezet aan de linkerkant van de tabel en de uren aan de rechterkant.

Product/Proces	Uren
Voorbereiding	
Opstellen plan van aanpak	40
Aantal uren per onderdeel	**40**
Definitiestudie	
Opstellen plan van aanpak	5
Definiëren ontwikkelscenario	4
Bereid pilotplan-workshop voor	13
Evalueer pilot	8
Definieer systeemeisen	35
Bepaal systeemconcept	8
Beschouw technische structuur	8
Beschouw organisatorische inrichting	4
Stel pilotplan op	20
Aantal uren per onderdeel	**105**
Pilotontwikkeling	
Opstellen plan van aanpak	11
Bereid pilotontwerp-workshop voor	16
Specificeer globaal-functionele structuur pilot	24
Specificeer globaal-technische structuur pilot	24

Specificeer globaal-organisatorische inrichting	24
Stel pilotontwikkelplan op	24
Ontwerp software-bouweenheden	105
Bouw software-bouweenheden	105
Stel handleidingen pilot samen	24
Stel invoeringsprocedures pilot op	12
Integreer bouweenheden	12
Beoordeel en test pilotdeel	12
Aantal uren per onderdeel	**393**
Pilotinvoering	
Opstellen plan van aanpak	16
Invoeren pilot	20
Uitvoeren pilotacceptatie	12
Aantal uren per onderdeel	**48**
Afronding	
Verzamelen handleidingen	2
Opstellen gebruikershandleiding	5
Aantal uren per onderdeel	**7**
Totaal	
Totaal aantal uren	**591**

5.3. Resourceplan

De projectgroep bestaat uit één persoon. Deze dient zich minimaal 40 uur per week te bezigen met het project. De werktijden zijn vastgesteld op 8:00 – 17:00. De opdrachtnemer werkt gedeeltelijk zelfstandig. Een docent aan de Haagse Hogeschool zal dienen als technische ondersteuning van het gehele project.

5.4. Financieel plan

Er zijn geen kosten verbonden aan de uitvoering van het project. Er zullen ook geen kosten gemaakt ten behoeve van het project zonder uitdrukkelijke toestemming van de opdrachtgever. Wanneer de opdrachtgever toestemming geeft voor het maken van kosten, worden deze kosten volledig door de opdrachtgever vergoed.
De opdrachtgever kan op geen enkele wijze de opdrachtnemer aansprakelijk stellen voor gemaakte kosten ten behoeve van het project. Ook kan de opdrachtgever de opdrachtnemer geen kosten laten maken zonder deze volledig te vergoeden.

6. Kwaliteitsborging

Om aan te tonen dat dit project op HBO niveau wordt uitgevoerd, moet het product en proces aan vastgestelde eisen voldoen. Tevens zal er richting de opdrachtgever naast een professionele houding ook een HBO kwalitatief product opgeleverd worden.

6.1. Productkwaliteit

De opgeleverde rapporten moeten voldoen aan de rapportage-eisen zoals die tijdens de module AV-03 aan de Haagse Hogeschool gesteld worden. Daarnaast moeten de rapporten naast technische informatie ook een uitleg voor de opdrachtgever bevatten. Het informatiesysteem moet met een beperkte uitleg te begrijpen zijn, omdat de gebruikers van het systeem geen technische achtergrond zullen hebben, zal de functionaliteit voor zich moeten spreken..
Naast deze functionaliteit zal tevens een gebruikershandleiding opgesteld worden, die bij eventuele vragen als ondersteuning zal dienen.

6.2. Proceskwaliteit

Iedere dag zal een kleine evaluatie geschreven worden van de taken die vervuld zijn en over de producten waaraan gewerkt is. Deze kleine evaluaties zullen uiteindelijk de basis vormen voor het afstudeerverslag, waarin in een aantal weken voor het definitieve oplevermoment een concept van moet worden ingeleverd.

De opdrachtgever zal uitsluitend de product gerelateerde rapporten ontvangen. Deze zullen indien noodzakelijk worden toegezonden op een door met de opdrachtgever overeengekomen wijze.
Tussentijdse producten die door de begeleidende docent becommentarieerd moeten worden, worden een dag voor de bespreking, vòòr 12:00 toegezonden.

6.3. Testaanpak

Om de gerealiseerde producten te kunnen testen op correcte implementatie, zullen de onderdelen van de producten moeten worden getest. De test zal bestaan uit het opstellen van testspecificaties (cases) die voldoen aan de eisen van het “black box”-testen. Testen aan de hand van black-box, houdt in dat een bepaalde invoer een verwachte uitvoer geeft. Om de resultaten van de testcase succesvol te kunnen benoemen moet de verwachte uitvoer gelijk zijn aan de verkregen uitvoer.
Indien het niet mogelijk of niet relevant is om testspecificaties op te stellen volgens dit principe, zal zijn aangegeven op welke andere manier er getest is. Het uitvoeren van deze tests, zal bijdragen aan de kwaliteit van het product.

6.4. Voorgestelde maatregelen

De gebruikte methode is IAD, deze methode is door de opdrachtnemer nog nooit in de praktijk toegepast. Het boek van de dhr R.J.H. Tolido zal hierbij worden gebruikt als richtlijn.
In dit project zullen een aantal onbekende factoren zich kunnen voordoen, zoals knelpunten bij het toepassen van de methode IAD. Ook bij het coderen kunnen complicaties optreden. Vandaar dat er in de volgende paragraaf wordt ingegaan op risicomanagement.
De programmeertaal Delphi die gebruikt gaat worden is bij de opdrachtnemer redelijk bekend, de kennis van deze taal zal worden ondersteund door een naslagwerk.

6.5. Risicomanagement

Risicomanagement, een manier waarmee in een vroeg stadium nagedacht is over de eventuele knelpunten die zich kunnen voordoen op elk tijdstip binnen het project. Onderstaand is een matrix opgenomen die de knelpunten opsomt met daartegenover één of meerdere oplossingen. Daarnaast worden de kans en het gewicht in de matrix opgenomen. De kans geeft met een geschat getal - variërend van 0.0 – 1.0 - aan hoe groot de kans is op dat het risico zich voordoet. Daarnaast is de impact opgenomen waarbij met een geschat getal – variërend van 1 tot 10 – de invloed van het risico wordt aangegeven indien deze zich binnen het project zou voordoen. Hoe kleiner de score (Kans * Impact), hoe kleiner het risico.

Risicomanagement Matrix			
Risico	Oplossing(en)	Kans	Impact
Project			
Afstudeeropdracht niet op tijd af te ronden	Opdrachtgever informeren over huidige situatie en alternatieve mogelijkheden. Bijstellen van de planning en afstudeerverslag uitleggen waarom dit niet voorkomen had kunnen worden.	0.4	6
Hardwaremankement	Zo snel als mogelijk repareren en meest recente data ophalen van extern medium. Indien het niet mogelijk is de hardware te repareren, deze vervangen en tussentijds een ander werkstation gebruiken voor het bewerken van de data. In alle gevallen bijstellen van de planning	0.2	5
Onvoldoende kennis van Delphi 7	Online website raadplegen voor advies Naslagwerk: “Advanced programming with Delphi 7” raadplegen	0.4	8
Onvoldoende kennis van IAD	IAD-boek van de heer Tolido raadplegen. De heer Nederend, docent Haagse hogeschool, raadplegen voor advies	0.6	8
Organisatie failliet	Aan de hand van het stadium van faillissement ten opzichte van de afstudeerperiode beslissingen nemen in samenwerking met school	0.1	10
Softwaremankement (Besturingssysteem)	Indien mogelijk verhelpen, anders opnieuw installeren	0.2	5
Ziekte Arno Nederend	Er is een lijst opgesteld met overige docenten die ook vragen over IAD zouden kunnen beantwoorden	0.2	1
Ziekte van opdrachtgever	Overige werknemers raadplegen voor dringende vragen Vragen desnoods via e-mail of een ander medium	0.2	4
Ziekte van opdrachtnemer	Planning bijstellen en (indien langdurig) de heer Schipper op de hoogte stellen	0.2	10

Bijlage A: Beoordeling rapportagetechniek

Onderdeel	Element		Aanwezig/correct
Titelpagina	Titel (en evt. ondertitel)		ja/nee
	Naam student(en)		ja/nee
	Naam opdrachtgever/docent		ja/nee
	Modulenummer en -naam		ja/nee
	Plaats en datum		ja/nee
	(Naam bedrijf)		ja/nee
Voorwoord	Tussen titelpagina en inhoudsopgave		ja/nee
	Persoonlijk		ja/nee
	Kader/doelgroep/leeswijzer		ja/nee
	Dankbetuiging(en)		ja/nee
	Plaats, datum en namen studenten		ja/nee
Inhoudsopgave	Inhoudsopgave en voorwoord afwezig		ja/nee
	Inleiding (= hoofdstuk 1)		ja/nee
	Niveaus in hoofdstukken (max. 3) aangebracht		ja/nee
	Verband tussen hoofdstukken en paragrafen		ja/nee
	Inspringen per niveau		ja/nee
	Typografisch onderscheid per niveau		ja/nee
	Titels: informatief		ja/nee
	consequent/gelijkvormig		ja/nee
	kernachtig geformuleerd		ja/nee
	Literatuurlijst (géén hoofdstuk)		ja/nee
	Bijlagen met nummer/letter en titel		ja/nee
	Paginanummering		ja/nee
Inleiding	Aanleiding (achtergrond/probleem//belang)		ja/nee
	Doelstelling rapport:	hoofdvraag	ja/nee
		werkwijze	ja/nee
		randvoorw./uitgangspunten	ja/nee
	Structuurbeschrijving/samenvatting hoofdstukken		ja/nee
	Alinea-indeling		ja/nee
	Goede volgorde alinea's		ja/nee
Kern	Elk hoofdstuk begint met inleiding		ja/nee
	Heldere structuur:	één onderwerp per alinea	ja/nee
		alinea's beginnen met kernzin	ja/nee
		alineagroepen	ja/nee
		puntsgewijze opsommingen	ja/nee
	Conclusies en aanbevelingen		ja/nee
	Rapportindeling overeenkomstig inhoudsopgave		ja/nee
	Verwijzingen naar de bijlagen		ja/nee
Literatuurlijst	Volgens richtlijnen voor titelbeschrijvingen		ja/nee
Bijlagen	Nummer/letter en titel		ja/nee
	Zelfstandig leesbaar		ja/nee
	Paginanummering loopt door in de bijlagen		ja/nee
	(Uitwerking interview volgens de eisen)		ja/nee
	(Notulen: zakelijk verslag/besluitenlijst)		ja/nee
Publieksgerichtheid/	Leesbaar/begrijpelijk voor opdrachtgever/belanghebbenden		ja/nee
Verzorgdheid	Spelling correct/ typefouten ontbreken		ja/nee
	Taalkundig correct		ja/nee
Deadline	Tijdig opgeleverd		ja/nee

Bijlage 2
Rapport Definitiestudie

Voorwoord

Voor u ligt de definitiestudie van de applicatie W!se Delivery. Dit rapport is bestemd voor zowel de projectgroep als voor de opdrachtgever. Dit rapport zal tijdens het project gebruikt worden als basis voor de aankomende ontwikkelingen.
Het is in het belang van het project dat alle betrokkenen het rapport aandachtig doorlezen en bij eventuele vragen dit kunnen terugkoppelen aan de projectgroep.

Cees Kruijf
Leiden, 22–09–2004

Samenvatting

De functionele systeemeisen zijn vertaald naar een aantal te ontwikkelen subsystemen. Deze subsystemen kunnen onafhankelijk van elkaar worden ontwikkeld. Voordat de subsystemen ontwikkeld kunnen worden zijn er eerst een aantal functionaliteiten nodig die de applicatie ondersteunen. Deze functionaliteiten zijn onderverdeeld in zogenoemde kernel-pilots:

- GUI (Grafische User Interface)
- Database
- GUI-database interface

Deze zogeheten kernel-pilots, zorgen voor de basis van de applicatie. Naast de basis zullen de systeemeisen, die zijn vertaald naar use-cases, worden ontwikkeld in de ‘normale’ pilots

- Subsystemen 1 tot en met 3
- Subsysteem 4

De prioriteiten in het ontwikkelen van de pilots is verwerkt in de planning. De ontwikkeling zal incrementeel zijn, wat inhoudt dat de pilots na elkaar worden ontwikkeld. De eerste pilot zal echter na de ontwikkeling ook direct worden ingevoerd, zodat feedback van het U-team (User/ gebruikerteam) kan worden verzameld).

Inhoudsopgave

1. Inleiding

De definitiestudie wordt ontwikkeld om de basis van de toekomstige goed doordacht in kaart te brengen. De wensen en eisen van zowel de opdrachtgever als de opdrachtnemer worden verwerkt in dit rapport. Uit dit eisenpakket groeien modellen die volgens een bepaalde methodiek ontwikkeld moeten worden. Deze methodiek geeft de opdrachtnemer en eventueel de opdrachtgever een helder overzicht van de acties die in de toekomst uitgevoerd gaan worden.

In de aankomende hoofdstukken zal duidelijk worden welke functionaliteiten de toekomstige applicaties zal gaan ondersteunen, op welke wijze deze ontwikkeld gaan worden en hoeveel tijd dat in beslag gaat nemen. In de hoofdstukken 2, 3 en 4 zal worden uitgelegd welke functies de applicatie zal gaan ondersteunen en hoe de systeemeisen zijn vertaald naar logische functies. De twee daaropvolgende hoofdstukken brengen respectievelijk de technische eisen en de organisatorische gevolgen in kaart. Het rapport wordt afgesloten met een pilotplan en de validatie van de onderdelen in dit rapport.

2. Systeemeisen

De systeemeisen zijn de eisen en mogelijkheden van het toekomstige systeem. Deze systeem zullen naast de pilot-workshop ook worden vastgesteld door het reverse-engineeren van de huidige applicatie.

Dit hoofdstuk bestaat uit de twee paragrafen: functionele en niet-functionele eisen. In de functionele eisen zullen de eisen beschreven worden in de vorm van events waarop het systeem op een voorspelbare, afgesproken manier moet reageren.
De niet-functionele eisen, bevatten de overige minder makkelijk te toetsen eisen, zoals de performance-eisen en integriteiteisen.

Dit hoofdstuk zal worden afgesloten met een opsomming van de eisen met een prioriteit daaraan gekoppeld. Aan de hand van de prioriteitenlijst zal in het pilotplan worden bepaald in welke volgorde de functies geïmplementeerd gaan worden/

2.1. Functionele systeemeisen

Hieronder is de lijst met geprioriteerde systeemeisen. De eisen zijn geprioriteerd aan de hand van de MoSCoW analyse. De MoSCoW analyse kent de volgende indeling in prioriteiten:

- Must have (M)
- Should have (S)
- Could have (C)
- Would like to but probably could not have (W)

Systeemeis	Prioriteit
Het systeem moet de gegevens op de oude zwangerschapskaart kunnen realiseren.	M
De gegevens van cliënten die verwerkt worden in het systeem moeten grotendeels variabel zijn. (Doktoren, ziekenhuizen, ziektes, etc)	M
Het moet mogelijk zijn om te zoeken naar de cliënten op eventueel unieke gegevens	M
Het moet mogelijk zijn om ziekenhuizen bij te werken en er in te zoeken	M
Ziekenhuizen, verzekeraars, gynaecologen, doktoren, plaatsnamen, achternamen en voornamen. Deze gegevens moeten allemaal in een ‘zelfgroeiende’ lijst kunnen worden geselecteerd, zodra het systeem vereist dat deze worden ingevoerd.	M
Het systeem zal middels een functie de cliëntdata moeten kunnen exporteren naar een locatie buiten het systeem.	S
De gebruiker en/of beheerder kan alleen toegang krijgen tot het systeem middels een gebruikersnaam en wachtwoord	S
Er moet een mogelijkheid zijn om de gegevens te rapporteren	S
De LVR (Landelijke Verloskunde Registratie) vereist dat er periodiek een formulier wordt toegezonden met het aantal cliënten en extra gegevens, deze moet door het systeem kunnen worden gegenereerd.	C
De instellingen binnen de applicatie moeten kunnen worden veranderd	C

door een beheerder	
Gebruikers van het systeem moeten kunnen worden toegevoegd	C
De cliëntgegevens moeten worden gecontroleerd aan de hand van de voorwaarden van het LVR	W
Alle kosten die gemaakt worden in het cliënttraject moeten kunnen worden gerapporteerd	W

2.2. Niet-functionele systeemeisen

Naast de functionele eisen, zijn er ook de niet-functionele eisen. Deze laatste groep van eisen zijn de eisen die ervoor zorgen dat de applicatie naar verwachting zal gaan functioneren.

Systeemeis
De grafische user interface moet overzichtelijk zijn
Er mogen niet meerdere gebruikers tegelijkertijd met het systeem werken
Het systeem moet werken op een werkstation met een Windows operating systeem
Het systeem zal dag en nacht moeten kunnen draaien.
Registratie van eerste gesprek met client mag maximaal een half uur duren
Foutmeldingen moeten concreet en begrijpelijk zijn

3. Systeemconcept

In dit hoofdstuk zal het concept van het systeem in kaart gebracht worden. Dit concept zal een globale beschrijving zijn van het toekomstige systeem, dit zal worden gerealiseerd met behulp van de technieken van UML Naast het vaststellen van de actoren en functionaliteiten zullen de functionaliteiten worden opgedeeld in diverse subsystemen.

3.1. Use-case diagram

Het use-case diagram geeft in één oogopslag een overzicht van de actoren en de taken die uitgevoerd moeten kunnen worden in de toekomstige situatie.
In dit use-case diagram zijn er twee actoren, namelijk de gebruiker en de beheerder, zoals in het onderstaande figuur te zien is.


Uitvoeren LVR-controle
Rapporteren
Client toevoegen
anamnese bijwerken
Onderzoek vastleggen
baring invoeren
uitkomst bepalen
Gegevens kraambed invoeren
Babygegevens invoeren
Zoeken
Inloggen
ziekenhuizen bijwerken
Medici bewerken
Praktijk bewerken
Verzekeraars bewerken
Begeleiders bewerken
Gebruiker toevoegen
Wegschrjven data
Contactmoment bijwerken
Voornemens registreren
Toxische facotren bewerken
Termijnen instellen
«uses»
«uses»
«uses»
«uses»
«uses»
Gebruiker
Beheerder

3.2. Use-case beschrijvingen

Naast het use-case diagram is elke use-case in deze paragraaf beschreven. Deze beschrijving zal een beter inzicht geven in de diepgang en voorwaarden aan de functionaliteit van de use-cases. De onderstaande beschrijvingen gaan uit van de gewenste situatie. Dit houdt in dat de uitzonderingen die zouden kunnen optreden niet in dit stadium worden vermeld. Allereerst worden de use-case beschrijvingen van de gebruiker beschreven, daaropvolgend vanzelfsprekend de beheerder.

3.2.1. Gebruiker

Voor de gebruiker geldt dat voor alle use-cases deze ingelogd dient te zijn.

Use-case	Uitvoeren LVR-controle
Preconditie	De controleren data is beschikbaar en volledig
Postconditie	Data is gecontroleerd
Beschrijving	Deze use-case controleert de gegevens die vereist zijn voor LVR (Landelijke Verloskunde Registratie). Deze gegevens moeten volgens een bepaald protocol worden vastgelegd voordat deze kunnen worden verzonden naar de LVR.

Use-case	Rapporteren
Preconditie	Gegevens voor rapportage zijn beschikbaar
Postconditie	Gewenste rapport is in het juiste formaat geleverd
Beschrijving	1. De gebruiker dient in te loggen. 2. De gebruiker kiest het gewenste rapport 3. De gebruiker vult de eventuele ‘voorwaarden’ in, zoals cliëntnummer of datum. 4. Het systeem levert de uitvoer van het juiste rapport.

Use-case	Client toevoegen
Preconditie	-
Postconditie	De client is ingevoerd en de bijbehorende sub-cases zijn ook uitgevoerd.
Beschrijving	1. De gebruiker dient in te loggen 2. De gebruiker vult de personalia in 3. De gebruiker werkt het contactmoment bij 4. De gebruiker registreert de voornemens 5. De gebruiker registreert de toxische factoren 6. De gebruiker stelt de ‘a terme’ in, (bevallingstermijn) 7. De gebruiker vult de anamnese in

Use-case	Anamnese bijwerken

Preconditie	Een client is geselecteerd
Postconditie	De anamnese van de client is bijgewerkt
Beschrijving	1. De vorige (succesvolle) zwangerschappen kunnen worden bijgewerkt 2. De ‘a terme’ kan worden bijgewerkt 3. De eventuele ziektes / aandoeningen kunnen worden bijgewerkt 4. De sociale achtergrond kan worden bijgewerkt

Use-case	Baring invoeren
Preconditie	De client is bekend in het systeem De client heeft een lopende zwangerschap De client kan theoretisch bevallen
Postconditie	De bevalling is volledig ingevoerd
Beschrijving	1. Invoeren partusgegevens 2. Invoeren ontsluitingsgegevens 3. Invoeren uitdrijvingsgegevens 4. Invoeren placentagegevens 5. Invoeren eventuele opgetreden trauma 6. Invoeren vastgestelde tijden 7. Invoeren medische aanwezigen 8. Invoeren babygegevens

Use-case	Uitkomst bepalen
Preconditie	De client is bekend in het systeem
Postconditie	De uitkomst van de zwangerschap van de client is in het systeem vastgelegd
Beschrijving	1. De uitkomst wordt vastgelegd 2. De einddatum en de eventuele eindstatus van de zorg worden vastgelegd 3. Aantal vruchten / kinderen wordt vastgelegd, afhankelijk van de uitkomst. 4. Eventuele medische indicatie wordt bepaald

Use-case	Inloggen
Preconditie	De gebruiker is bekend in het systeem
Postconditie	Het systeem heeft toegang verleend aan de gebruiker
Beschrijving	1. De gebruiker vult een combinatie van gebruikersnaam en wachtwoord in 2. Het systeem bevestigt de juiste combinatie

Use-case	Voornemens registreren
Preconditie	De client is bekend in het systeem

Postconditie	De voornemens van de client zijn vastgelegd
Beschrijving	1. De gebruiker selecteert de plaats van bevalling 2. De gebruiker legt de kraamzorg vast 3. De gebruiker legt de soort babyvoeding vast 4. De gebruiker legt de bevallingsvoorbereiding vast

Use-case	Contactmoment bijwerken
Preconditie	De client is bekend in het systeem
Postconditie	Het contactmoment met de client is bijgewerkt
Beschrijving	1. De contactdatum wordt door het systeem bepaald 2. De verwijzing wordt ingevoerd (eventueel de praktijk en verwijzer) 3. Het stadium van de zwangerschap wordt bepaald en vastgelegd 4. De zorgvraag wordt ingevoerd 5. De reden van verwijzing wordt ingevoerd 6. Datum van overname zorg wordt ingevoerd 7. Eventuele reden van zorgovername wordt ingevoerd

Use-case	Toxische factoren bewerken
Preconditie	De client is bekend in het systeem De client heeft toxische factoren gebruikt
Postconditie	Het gebruik van toxische factoren zijn vastgelegd
Beschrijving	1. De gebruiker bepaalt het soort ‘drug’ en de mate van gebruik

Use-case	Termijnen instellen
Preconditie	Client is bekend in het systeem Client heeft een lopende zwangerschap
Postconditie	Zwangerschaptermijn is vastgelegd in het systeem
Beschrijving	1. Datum laatste menstruatie bepalen 2. Cyclus vastleggen 3. Eventuele zwangerschaptest invoegen 4. Eventuele gegevens van echo invoegen 5. Vastleggen conceptiedatum 6. Vastleggen fertifiteitsbehandeling 7. Vastleggen voorafgaande anticonceptie 8. Bepalen ‘A terme’ (datum uitgerekend + weken) 9. Vastleggen zekerheid ‘A terme’.

Use-case	Onderzoek vastleggen

Preconditie	Client is bekend in het systeem
Postconditie	Onderzoeksgegevens zijn vastgelegd in het systeem
Beschrijving	1. Lichaamslengte vastleggen 2. Gewicht voor zwangerschap vastleggen 3. Gewicht bij eerste onderzoek vastleggen 4. Maximaal verwachte gewicht vastleggen 5. Bloedgroep vastleggen 6. Rhesusfactor vastleggen 7. Hoogste tensie vastleggen 8. Proteïnurie niveau vastleggen 9. Eventuele bijzonderheden vastleggen

Use-case	Gegevens kraambed invoeren
Preconditie	Client is bekend in het systeem Uitkomst zwangerschap van client is positief Zorgvraag van client is inclusief het kraambed
Postconditie	Gegevens kraambed zijn ingevoerd
Beschrijving	Eerste controle: 1. Vastleggen lactatieremming 2. Vastleggen kraamzorg 3. Bepalen van Anti D. 4. Eventuele bijzonderheden invoegen Nacontrole: 1. Gebruiker bepaalt de datum 2. Gebruiker legt babyvoeding vast 3. Gebruiker legt anticonceptie vast 4. Gebruiker legt perineum vast 5. Gebruiker bepaalt mate van genezing 6. Gebruiker bepaalt continentie 7. Gebruiker legt eventuele bijzonderheden vast van de kraamvrouw.

Use-case	Zoeken
Preconditie	-
Postconditie	Resultaat wordt weergegeven
Beschrijving	1. Bepalen zoekcategorie 2. Invoeren criteria

Use-case	Babygegevens invoeren
Preconditie	Client is bekend in het systeem Uitkomst zwangerschap van client is positief
Postconditie	Gegevens van baby is vastgelegd in het systeem
Beschrijving	1. Vastleggen conditie baby 2. Voedingswijze bepalen 3. Eventuele opname en ontslag vastleggen 4. Eventuele bijzonderheden omtrent sterfte vermelden

3.2.2. Beheerder

Use-case	Ziekenhuizen bijwerken
Preconditie	Beheerder heeft toegang tot het beheerdermenu
Postconditie	Ziekenhuizen zijn bijgewerkt
Beschrijving	1a Ziekenhuis selecteren 1b Maak nieuw ziekenhuis aan 2 Vul gegevens in 3. Sla de gegevens van het ziekenhuis op

Use-case	Medici bewerken
Preconditie	Beheerder heeft toegang tot het beheerdermenu
Postconditie	Medici zijn bijgewerkt
Beschrijving	1a Medici selecteren 1b Maak nieuwe medicus aan 2 Vul gegevens in 3. Sla de gegevens van de medicus op

Use-case	Praktijk bewerken
Preconditie	Beheerder heeft toegang tot het beheerdermenu
Postconditie	Praktijk is bijgewerkt
Beschrijving	1a Praktijk selecteren 1b Selecteer nieuwe Praktijk 2 Vul gegevens in 3. Sla de gegevens van de Praktijk op

Use-case	Verzekeraars bewerken
Preconditie	Beheerder heeft toegang tot het beheerdermenu
Postconditie	Verzekeraar zijn bijgewerkt
Beschrijving	1a Verzekeraar selecteren 1b Selecteer nieuwe Verzekeraar 2 Vul gegevens in 3. Sla de gegevens van de Verzekeraar op

Use-case	Begeleiders bewerken
Preconditie	Beheerder heeft toegang tot het beheerdermenu
Postconditie	Begeleiders zijn bijgewerkt
Beschrijving	1a Begeleider selecteren 1b Selecteer nieuwe Begeleider 2 Vul gegevens in 3. Sla de gegevens van de Begeleider op

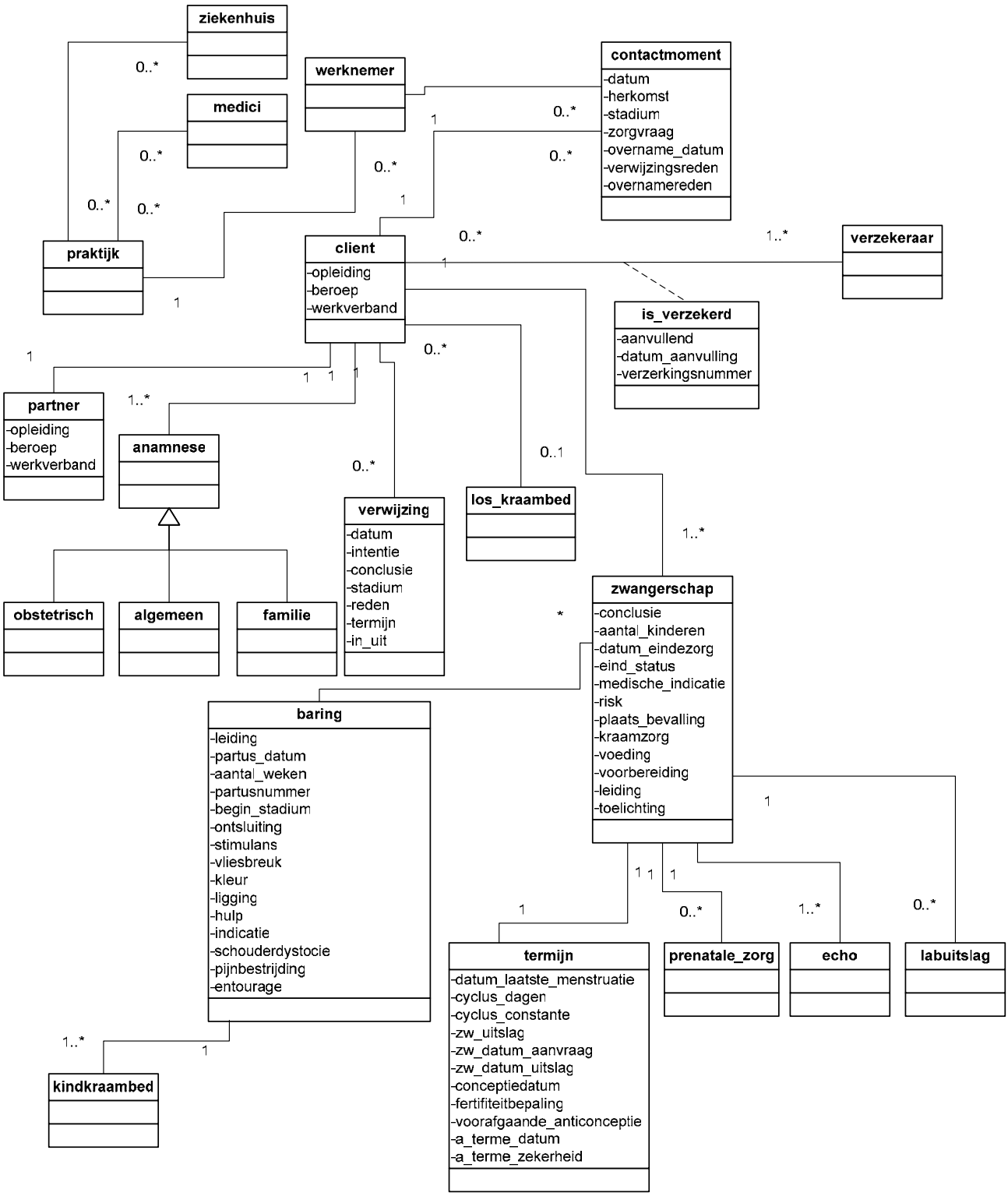
Use-case	Gebruiker toevoegen
Preconditie	Beheerder heeft toegang tot het beheerdermenu Gebruiker is nog niet bekend in het systeem
Postconditie	Gebruiker is toegevoegd
Beschrijving	1. Maak gebruikersnaam aan 2. Vul overige gegevens in 3. Sla gebruikergegevens op

Use-case	Wegschrijven data
Preconditie	Data is beschikbaar voor wegschrijven Geen gebruiker meer actief in het systeem
Postconditie	Data is weggeschreven naar geselecteerd medium
Beschrijving	1. Selecteer medium 2. Selecteer weg te schrijven data 3. Volg aangekondigde stappen 4. Klik op wegschrijven

3.3. Klasse diagram

In deze paragraaf worden de relaties tussen de verschillende klassen duidelijk middels een klassendiagram. In het volgende klassendiagram ontbreekt het grootste gedeelte van de attributen en zijn de procedures weggelaten. In het onderstaande diagram is om de overzichtelijkheid te bewaren ervoor gekozen om alleen de relaties en enkele belangrijke attributen in kaart te brengen.

De overige klassen zullen worden uitgewerkt in het pilotplan. In het pilotplan zullen de klassen, met bijbehorende attributen en procedures, bij pilots ondergebracht worden, die later deze klassen gaan aanspreken. In deze pilots zullen die klassen ook verder worden uitgewerkt.


ziekenhuis
werknemer
contactmoment
-datum
-herkomst
-stadium
-zorgvraag
-overname_datum
-verwijzingsreden
-overnamereden
medici
praktijk
client
-opleiding
-beroep
-werkverband
verzekeraar
is_verzekerd
-aanvullend
-datum_aanvulling
-verzerkingsnummer
partner
-opleiding
-beroep
-werkverband
anamnese
verwijzing
-datum
-intentie
-conclusie
-stadium
-reden
-termijn
-in_uit
los_kraambed
obstetrisch
algemeen
familie
zwangerschap
-conclusie
-aantal_kinderen
-datum_eindezorg
-eind_status
-medische_indicatie
-risk
-plaats_bevalling
-kraamzorg
-voeding
-voorbereiding
-leiding
-toelichting
baring
-leiding
-partus_datum
-aantal_weken
-partusnummer
-begin_stadium
-ontsluiting
-stimulans
-vliesbreuk
-kleur
-ligging
-hulp
-indicatie
-schouderdystocie
-pijnbestrijding
-entourage
kindkraambed
termijn
-datum_laatste_menstruatie
-cyclus_dagen
-cyclus_constante
-zw_uitslag
-zw_datum_aanvraag
-zw_datum_uitslag
-conceptiedatum
-fertifiteitbepaling
-voorafgaande_anticonceptie
-a_terme_datum
-a_terme_zekerheid
prenatale_zorg
echo
labuitslag
0..*
1
1..*
0..1
*

3.4. Subsystemen

Subsystemen, dit zijn kleine systemen binnen het gehele informatiesystemen die onafhankelijk van elkaar kunnen worden gerealiseerd en geïmplementeerd. Deze subsystemen worden gedefinieerd zodat deze wellicht in het pilotplan kunnen dienen als separate pilot. Ieder subsysteem zal in de aankomende subparagrafen worden gedefinieerd op basis van het - in dit document verwerkte - use-case diagram.

3.4.1. Eerste contactmoment cliënt

Bij het eerste contactmoment met een cliënt moeten heel veel gegevens in één keer worden vastgelegd. Dit proces mag echter maar maximaal maar een half uur duren. Alle procedures die binnen dit half uur uitgevoerd moeten worden vorm daarom dit subsysteem. Dit zijn de volgende:

- Inloggen
- Cliënt toevoegen
    - Contactmoment bijwerken
    - Voornemens registreren
    - Toxische factoren bijwerken
    - Termijn instellen
- Anamnese bijwerken
- Uitkomst bepalen
- Onderzoek vastleggen

3.4.2. Zwangerschap

Onder zwangerschap valt alles omtrent de zwangerschap na het eerste contactmoment tot de baring en het verzorgen van het kraambed. In dit subsysteem zullen alle procedures die hiermee te maken hebben worden uitgevoerd. Dit zijn de volgende:

- Baring invoeren
- Gegevens kraambed invoeren
- Babygegevens invoeren
- Zoeken

3.4.3. Administratie

Dit subsysteem behandelt alle procedures die voor de praktijk van belang zijn en waar geen direct contact met cliënten vereist is. Dat zijn de volgende use-cases:

- Rapporteren
- Uitvoeren LVR-Controle

3.4.4. Beheren

Het beheren van de gegevens en instellingen van de toekomstige applicatie vallen onder dit subsysteem. De volgende use-cases vallen onder dit subsysteem:

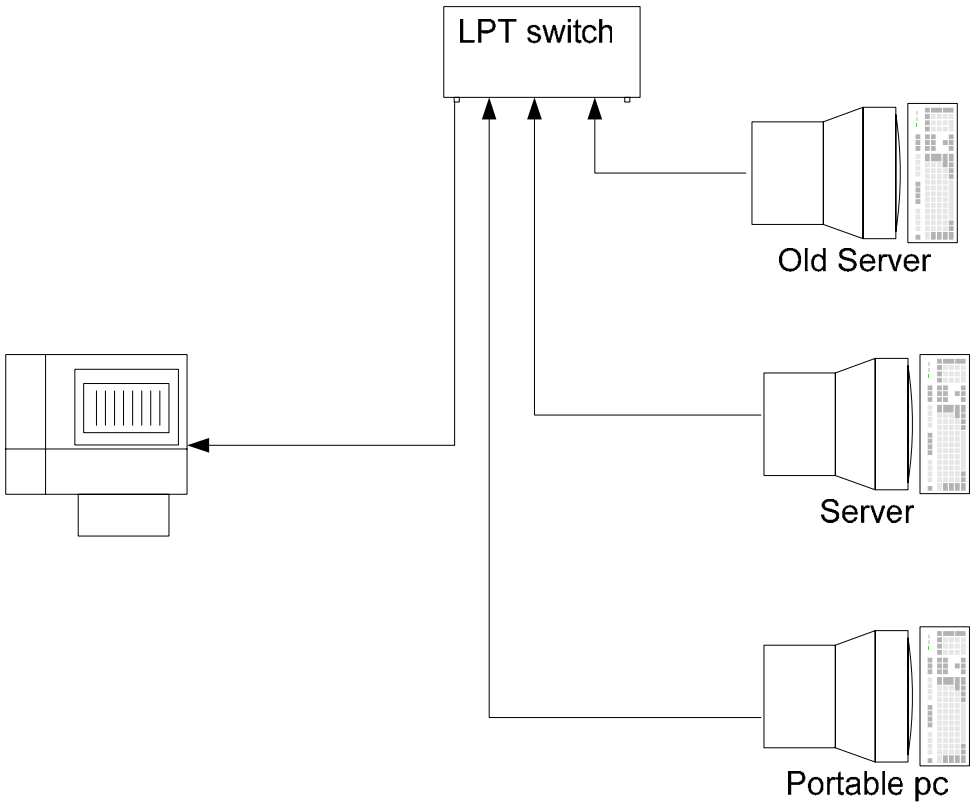
- Ziekenhuizen bijwerken
- Medici bijwerken
- Praktijk bijwerken
- Verzekeraars bijwerken
- Begeleiders bijwerken
- Gebruiker toevoegen
- Wegschrijven data

4. Technische structuur

In dit hoofdstuk wordt ingegaan op de technische infrastructuur. Hierbij moet worden gedacht aan de huidige en de toekomstige (vereiste) omstandigheden. Er zal worden getoetst of het huidige systeemconcept volledig zal kunnen functioneren in de huidige infrastructuur. De mogelijkheden voor eventuele benodigde veranderingen aan hard- en software zullen worden toegelicht.

4.1. Beschouw technische architectuur

De huidige infrastructuur is recent gemoderniseerd. Deze bestaat op het moment uit twee nieuwe werkstations, een oude server en een faxmodem/printer/kopieerapparaat combinatie, zie figuur 4.1.



Figuur 4.1

Het systeemconcept zal draaien op een werkstation dat in contact staat met een printer en bijvoorkeur dag en nacht ingeschakeld staat. Dit werkstation zal moeten beschikken over een beschrijfbaar medium waar de data veilig kan worden weggeschreven.
Onder de huidige omstandigheden zijn om die reden geen wijzigingen noodzakelijk aan de architectuur.

Een aanbeveling is echter wel dat de huidige architectuur wordt heringericht met ondersteuning van een netwerk, zodat de werkstations constant met elkaar in verbinding staan, zodra deze zijn ingeschakeld. De huidige printer zal alleen toegankelijk zijn voor het werkstation waar de LPT switch op staat ingesteld.

Naast de verbeterde printertoegankelijkheid, wordt ook communicatie tussen de werkstations mogelijk. Uitwisselen van data tussen de werkstations hoeft niet meer middels een beschrijfbaar medium te gebeuren.

4.2. Beschouw technologische basis

De technologische basis is de benodigde hard- en software die het huidige systeemconcept vereist om goed te kunnen functioneren. Vandaar dat in deze paragraaf tussen deze twee onderscheid gemaakt zal worden.

4.2.1. Hardware

In de vorige paragraaf werd ingegaan op de infrastructuur van de aanwezige randapparatuur, maar niet de hardware. Ieder component genoemd in de vorige paragraaf zal hier worden toegelicht met de technische specificaties.

Randapparatuur	Technische specificaties
Faxmodem/printer	[Productnaam] Faxmogelijkheid Printmogelijkheid Kopieermogelijkheid LPT1-aansluiting
Portable PC	Pentium IV 3 Gigahertz 256 MB DDR Ram 80 Gigabyte Harddisk
Server	Pentium IV 3 Gigahertz 256 MD DDR Ram 80 Gigabyte Harddisk
Old Server	Pentium II 333 Megahertz 64 MB RAM 6 Gigabyte Harddisk
LPT Switch	4 ports

De benodigde hardware voldoet aan de eisen van het huidige systeemconcept. Het systeemconcept zal namelijk de eisen overnemen van de taal waarin het systeem zal worden ontwikkeld.

4.2.2. Software

Het besturingssysteem wat op de huidige werkstations is geïnstalleerd, is één van de Windows-varianten; Windows XP Home Edition of Windows 98 Second Edition. Het systeem zal opereren op beide varianten, dus aanschaf van een nieuw besturingssysteem is niet noodzakelijk.

Echter de taal waarin het systeem wordt gemaakt vereist zekere bibliotheekbestanden, die niet standaard bij een Windows-besturingssysteem worden geleverd. Dit zal naar alle waarschijnlijkheid worden afgevangen bij de installatie van het systeem, zo niet, dan zullen de bibliotheekbestanden worden geleverd met het systeem.

4.3. Kosten/baten analyse

Er zijn aan het project geen kosten verbonden, noch zullen er kosten gemaakt kunnen gaan worden. Hoewel aanschaf van nieuwe producten ook niet noodzakelijk is, is er wel een analyse uitgevoerd voor het advies omtrent het aanleggen van een klein LAN (Local Area Network).

Aanleg nieuw netwerk:

Randapparatuur	Kosten in euro
Sweex Switch 10/100mbps	€ 13,75
15 Meter Cat 5e UTP kabel	€ 3,98
15 Meter Cat 5e UTP-kabel	€ 3,98
3 Meter Cat 5e UTP-kabel	€ 2,00
Aanleg	€ 25,00
Totale kosten	€ 48,71

5. Organisatorische inrichting

5.1. Inrichting

De organisatie, de verloskundigenpraktijk de Maatschap, is een organisatie bestaande uit de volgende vier medewerkers:

- Trees van het Kaar
- Barbara Lapperre
- Viola Vlogtman
- Susanne van Iterson

Volgens de typologie van Mintzberg is de organisatie een professionele organisatie met een uitzondering.
De professionele organisatie wordt vooral gekenmerkt doordat de werknemers tijdens de uitvoering van het werk eigen beslissingen nemen, zonder de 'chef' te raadplegen. De operationele kern van de organisatie bevat de professionals, maar in het geval van de verloskundepraktijk zijn de professionals tevens het management.
Dit laatste kenmerk lijkt veel op een ondernemersorganisatie, echter is van dit type geen sprake omdat deze gekenmerkt wordt door één en maximaal twee stafleden.

5.2. Opleidingsdoelstellingen

Er zullen in de toekomstige situatie twee typen gebruikers zijn, de 'normale' gebruiker en de beheerder. Voor deze twee typen zal een korte introductie worden gegeven over de applicatie middels een presentatie. Deze introductie zal naast de opgestelde gebruikersdocumentatie voldoende moeten zijn voor het gebruik van het toekomstige systeem.

5.3. Vereiste gebruikersdocumentatie

Het toekomstige systeem zal worden ondersteund met drie verschillende documenten:

- één quick-reference guide (een korte opsomming van de functies en hoe deze te bereiken zijn binnen de applicatie)
- een uitgebreide gebruikershandleiding
- en een uitgebreide beheerdocumentatie

De wijze van oplevering van de handleidingen zal in een later stadium pas bekend worden.

6. Het pilotplan

6.1. Definieer pilotstructuur

Voor het ontwikkelen van de applicatie is gekozen voor drie kernel-pilots en twee normale pilots.

Kernel pilots:
1. GUI
2. Database
3. GUI-database interface

Normale pilots:
1. Subsystemen 1 tot en met 3
2. Subsysteem 4

De kernel-pilots zullen als eerste ontwikkeld worden in de volgorde dat ze zijn opgesomd. De GUI zal vrij snel afgerond moeten worden, zodat van het U-team voldoende feedback kan worden verkregen en de applicatie in de toekomstige situatie goed zal aansluiten bij de wensen van de opdrachtgever.

Tijdens het verzamelen van feedback op de GUI-pilot, kunnen de overige twee kernel-pilots ontwikkeld worden. Na het afronden van de laatste kernel-pilot kan worden gekeken of er tijd en ruimte is om de verzamelde feedback direct te realiseren, of te gebruiken in een later stadium.

Na de gerealiseerde kernel-pilots, kan de ontwikkeling van de normale pilots beginnen. In het systeemconcept is reeds naar voren gekomen hoe de globale structuur eruitziet. In de volgende paragrafen zal per pilot worden beschreven wat de structuur is, hoelang de ontwikkeling gaat duren, wat het gewenste resultaat is en welke personen zich met de pilot bezighouden. Tevens wordt er verwezen naar dit hoofdstuk dat per pilot de structuur in detail wordt uitgewerkt, daarom is bij elke pilot een subparagraaf inhoud opgenomen waarin dit wordt toegelicht.

6.2. Grafische User Interface

6.2.1. Pilotinhoud

De grafische user-interface heeft geen verdere toevoeging op het klassendiagram. Zo zal de grafische user-interface worden gemaakt aan de hand van de systeemeisen in combinatie met het use-case diagram.
De prioriteiten die aan de systeemeisen zijn toegekend zullen als basis dienen voor de toegankelijkheid waarmee een functie te bereiken is.

Naast de gestelde eisen door de opdrachtgever en de use-cases, zal aan de hand van een online handleiding over gebruikersvriendelijkheid een professionele en ‘vriendelijke’ grafische user interface worden gerealiseerd.

6.2.2. Geschatte ontwikkeltijd

De ontwikkeltijd voor deze pilot zal ongeveer 40 uur bedragen. In deze 40 uur zullen alle grafische user interfaces worden gerealiseerd inclusief het fictief gebruik van de applicatie. Dit fictieve gebruik houdt in dat iedere actie een vaste actie oplevert

6.2.3. Gewenste resultaat

Het realiseren van deze pilot zal moeten leiden tot een volledig functionele grafische user interface, die aan het U-team kan worden voorgelegd opdat hier feedback over verzameld kan worden.
De achterliggende functies zijn in deze pilot niet aanwezig, maar een vergelijkbaar resultaat zal door het systeem worden gegenereerd.

6.2.4. Toegewezen teams

Het U-team en A-team zullen in dit geval nauw samenwerken. De grafische user interface zal tenslotte in de behoefte van het U-team moeten gaan voorzien, aangezien dat de medewerkers van de verloskundepraktijk zijn.

6.3. Database

6.3.1. Pilotinhoud

De inhoud van deze pilot is het vertalen van het klassendiagram naar een implementatiemodel. Deze zal vervolgens in de daarvoor bestemde MySQL-database worden geïmplementeerd.
In de definitiefase is reeds aangegeven dat de attributen en operaties niet zijn opgenomen. Deze zijn in deze paragraaf opgenomen in een tabel met de klassen en attributen.

Klassenaam	Attributen
Algemeen	cystitis herpes_labialis foliumzuur operatie bloedtransfusie thrombose psychiatrische_aandoening SOA vaginisme candida specialist alcohol drugs roken allergie medicatie bijzonderheden
Client	Voornaam tussenvoegsel

	meisjesnaam geboortedatum burgerlijke_staat straatnaam huisnummer postcode woonplaats telefoonnummer opleiding beroep werkverband
Contactmoment	Datum Herkomst Stadium Zorgvraag Overname_datum Verwijzingsreden Overnamereden
Echo	Datum Plaats Aanvullende_informatie Indicatie
Familie	diabetes hypertensie congenitale_afwijkingen heupdysplasie bijzonderheden
Familie_partner	congenitale_afwijkingen heupdysplasie bijzonderheden
Is_verzekerd	datum_aanvullend
Kind	Geslacht Voornaam Tussenvoegsel Achternaam Geslacht Gewicht Gewogen_apparaat Aspect Proporties Opname Apgar_1_min Apgar_5_min Apgar_10_min Conditie Reanimatie Testes_rechts Testes_links
Kraambed	Voeding Opname Conditie

	Ontslag Lactatieremming Kraamzorg Anti_D Bijzonderheden Aanvullend
Labuitslag	glucose_1 glucose_2 glucose_3 bloedgroep Hemoglobine MCV varchar(150) Antistoffen Hepatitis_b Rubella HIV Rhesusfactor Rhesusfactor_antid MCV_28 Hemoglobine_28
Los_kraambed	a_terme_datum zekerheid gravida para spontane_abortus abortus_provocatus EUG foetal_loss in_leven bloedgroep rhesusfactor partus_datum partus_tijd leiding plaats voorgenomen_plaats
Medici	Voornaam Tussenvoegsel Achternaam Titel Specialisme Straatnaam Huisnummer Postcode Woonplaats Telefoonnummer Actief
Obs_abortus	Ingreep Bijzonderheden
Obs_zwangerschap	Geslacht

	ligging hulp perineum bloedverlies conditie apgar_1 apgar_5 kind_voornaam kind_tussenvoegsel kind_achternaam geleid_door verantwoordelijke bijzonderheden toelichting
Obstetrisch	Datum Uitkomst Tijd_weken Tijd_dagen Anti_D Toelichting
Onderzoek	Lichaamslengte Gewicht_voor_zwangerschap Gewicht_bij_onderzoek Prenatale_diagnostiek Bijzonderheden
Partner	Voornaam Tussenvoegsel Achternaam Opleiding Beroep Werkverband
Partus	leiding plaats datum aantal_weken begin_stadium_a begin_stadium_b ontsluiting pijnbestrijding_baring stimulans vliesbreuk kleur problematiek_ontsluiting ligging hulp indicatie_uitdrijving schouderdystocie entourage pijnbestrijding_uitdrijving problematiek_uitdrijving

	leidingswijze geboortewijze bloedverlies medicatie ziektebeeld_nageboorte compleet eindbestemming aantal_vaten problematiek_nageboorte datum_vliezen tijd_vliezen tijd_meepersen datum_geboorte tijd_geboorte tijd_placenta ruptuur episiotomie type indicatie_trauma ziektebeeld_trauma overig_trauma gehecht_door problematiek_trauma aanvullend_commentaar
Praktijk	Naam Straatnaam Huisnummer Postcode Woonplaats Telefoonnummer Faxnummer
Termijn	Datum_laatste_menstruatie Menstruatie_is_onbekend Cyclus_dagen Zekerheid Fertifiteit Anticonceptie A_terme_datum A_terme_termijn A_terme_zekerheid
Verzekeraar	Naam Straatnaam Huisnummer Postcode Woonplaats Telefoonnummer Faxnummer
Werknemer	Voornaam Tussenvoegsel Achternaam

	Straatnaam Huisnummer Postcode Woonplaats Telefoonnummer Geboortedatum Wachtwoord
Ziekenhuis	Naam Straatnaam Huisnummer Postcode Woonplaats Telefoonnummer Faxnummer
Zwangerschap	datum_begin plaats_bevalling kraamzorg voeding voorbereiding aanvullend_commentaar

Dit zijn de klassen en hun bijbehorende attributen in tabelvorm. Dit domeinmodel zal zoals in UML verklaard wordt, bijna één op één kunnen worden vertaald. Waar nodig zullen extra tabellen aangemaakt en toegelicht worden.

6.3.2. Geschatte ontwikkeltijd

De ontwikkeltijd van deze pilot bevat het vertalen van het domeinmodel naar een implementatiemodel die geschikt is voor MySQL. Deze vertaling zal inclusief het controleren van de consistentie ongeveer 24 uur in beslag nemen.

6.3.3. Gewenste resultaat

Het gewenste resultaat is een volledige functionele database waarin alle benodigde data van de applicatie kan worden opgeslagen. De database kan in de nog te ontwikkelen pilots onvolledig blijken, indien nodig zal deze daar worden aangevuld. De vulling van de database, zal in de testfase van de derde kernel-pilot gebeuren.

6.3.4. Toegewezen teams

De gebruikers (U-team) hebben geen inbreng bij deze pilot, daarom zal in deze pilot alleen het A-team zich bezighouden met de ontwikkeling. Het U-team zal zich in deze fase bezighouden met ervaring opdoen met de eerste kernel-pilot.

6.4. GUI-database Interface

6.4.1. Pilotinhoud

De inhoud van deze pilot bevat het ontwikkelen van de functies die de database aanspreken en de benodigde acties uitvoeren om de juiste data terug te geven aan de GUI. Onderstaand zijn een aantal globale functies en bijbehorende omschrijving

opgenomen die minimaal in deze pilot ontwikkeld moeten worden ontwikkeld. In de pilot zal het verwachte resultaat en de daadwerkelijke functies worden gedetailleerd.

Functie	Omschrijving
Inloggen	Controle van gebruikersnaam en wachtwoord in de database.
Invoegen gegevens	Globale functie die het invoegen van een nieuwe rij in een tabel met bijbehorende gegevens realiseert.
Database_connect	Maakt connectie met de database, met statische gebruikersnaam en wachtwoord
Database_query	Voert een query uit op de database
Opvragen gegevens	Deze functie vraagt gegevens op van een bepaalde tabel die in het klassendiagram in kaart zijn gebracht. (Bijvoorbeeld: ziekenhuizen, praktijken, cliënten, enzovoorts)
Bijwerken gegevens	Deze functie werkt de gegevens bij die via de functie zijn verkregen en worden zo verwerkt in de database.

De bovenstaande functies zijn niet de functies die in de pilot gebruikt gaan worden, maar geven een beschrijving van de mogelijkheden van de interface. De taak van de interface is het afvangen van de communicatie tussen de database en de gebruiker. In de pilot zullen de daadwerkelijke functies nader worden toegelicht.

6.4.2. Geschatte ontwikkeltijd

De ontwikkeltijd van deze pilot zal ongeveer 80 uur bedragen. In deze pilot zal de communicatie met de database worden gerealiseerd, zodat de toekomstige pilots direct deze functies aan kunnen roepen.

6.4.3. Gewenste resultaat

Het gewenste resultaat van deze pilot is het afvangen van de communicatie tussen de database en de grafische user interface. De functies zullen geheel functioneel moeten zijn of in het uiterste geval een vergelijkbare testwaarde moeten teruggeven, zodat de subsystemen deze zonder problemen kunnen aanroepen .

6.4.4. Toegewezen teams

Zoals ook in de vorige pilot, is in dit stadium alleen het A-team belangrijk. Het U-team is nog steeds bezig met het verzamelen van feedback. Het U-team heeft tenslotte geen belang bij hoe de gegevens worden aangeroepen, alleen dat deze juist zijn op het moment dat deze ze nodig hebben.

6.5. Subsystemen 1-3

6.5.1. Pilotinhoud

De subsystemen één tot en met drie zijn gedefinieerd in hoofdstuk drie "Systeemconcept". Deze subsystemen bevatten voornamelijk de use-cases die de gebruiker in de toekomstige applicatie zal kunnen gaan gebruiken. Deze globale use-cases zullen in deze pilot worden gedetailleerd door te onderzoeken welke gegevens benodigd zijn en hoe deze via de interface verkregen worden.
Deze functies worden ondergebracht in de eerste pilot, de grafische user interface..

6.5.2. Geschatte ontwikkeltijd

De geschatte ontwikkeltijd van deze pilot is ongeveer 160 uur. Aangezien dit een van de grootste pilots is, neemt deze logischerwijs ook de meeste tijd in beslag.

6.5.3. Gewenste resultaat

Het gewenste resultaat is dat de functies voor de gebruiker moeten werken. Indien het resultaat niet binnen de toegewezen tijd bereikt kan worden, zullen de niet-operationele functies met een testwaarde worden aangevuld.

6.5.4. Toegewezen teams

Deze pilot zal grotendeels door het A-team worden uitgevoerd, hoewel de eventuele feedback van de eerste pilot misschien een rol kan gaan spelen. Dit zal moeten blijken uit de feedback.

6.6. Subsysteem 4

6.6.1. Pilotinhoud

Deze pilot zal het laatste subsysteem (de beheerder) implementeren. Dit subsysteem zal alle use-cases in het use-case diagram moeten gaan ondersteunen. In deze pilot zal ook worden gelet op de feedback op de grafische user interface die in de eerste pilot naar voren is gekomen.

6.6.2. Geschatte ontwikkeltijd

De geschatte ontwikkeltijd zal ongeveer 80 uur bedragen. Bij deze pilots, net als de overige pilots, wordt er gebruik gemaakt van time-boxing. Time-boxing wil zeggen dat er na de verstreken geplande uren er niet meer verder gewerkt wordt aan de pilot.

6.6.3. Gewenste resultaat

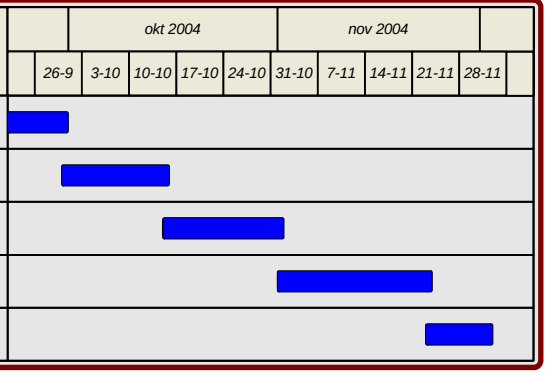
Het gewenste resultaat is een volledig operationeel subsysteem, voorzover de geplande tijd dat toelaat. Indien de geplande tijd onvoldoende is, zullen de onafgemaakte functies opgevuld worden met testwaarden.

6.6.4. Toegewezen teams

Het A-team zal zich vooral bezighouden met deze pilot, de door het U-team verzamelde feedback van de eerste pilot zal hier wellicht nog worden toegepast. Toch heeft het U-team over het algemeen in deze pilot geen rol.

6.7. Pilot planning

ID	*Pilotnaam*	*Begindatum*	*Einddatum*	*Tijdsduur (dagen)*
1	Grafische User-Interface	22-9-2004	30-9-2004	7d
2	Database	30-9-2004	15-10-2004	12d
3	GUI-Database Interface	15-10-2004	1-11-2004	12d
4	Subsystemen 1 tot en met 3	1-11-2004	23-11-2004	17d
5	Subsysteem 4	23-11-2004	2-12-2004	8d



7. Verificatie en validatie

Verificatie is het beoordelen van de resultaten van de definitiestudie, om de correctheid en de overeenstemming met de resultaten van de gemaakte documentatie in de voorgaande fases te verzekeren.
Validatie is het beoordelen van de definitiestudie om zeker te zijn dat aan de vastgelegde en aan de vanzelfsprekende eisen wordt voldaan.

7.1. Verificatie samenhang use-case diagram en functionele systeemeisen

Hieronder volgt een tabel met verticaal de use-cases en horizontaal de functionele systeemeisen. Aangezien er redelijk wat systeemeisen zijn, zijn deze hieronder opgesomd en hebben een code toegewezen gekregen, om de overzichtelijkheid te bewaren.

Code	Systeemeis
S1	Het systeem moet de gegevens op de oude zwangerschapskaart kunnen realiseren.
S2	De gegevens van cliënten die verwerkt worden in het systeem moeten grotendeels variabel zijn. (Doktoren, ziekenhuizen, ziektes, etc)
S3	Het moet mogelijk zijn om te zoeken naar de cliënten op eventueel unieke gegevens
S4	Het moet mogelijk zijn om ziekenhuizen bij te werken en er in te zoeken
S5	Ziekenhuizen, verzekeraars, gynaecologen, doktoren, plaatsnamen, achternamen en voornamen. Deze gegevens moeten allemaal in een ‘zelfgroeiende’ lijst kunnen worden geselecteerd, zodra het systeem vereist dat deze worden ingevoerd.
S6	Het systeem zal middels een functie de cliëntdata moeten kunnen exporteren naar een locatie buiten het systeem.
S7	De gebruiker en/of beheerder kan alleen toegang krijgen tot het systeem middels een gebruikersnaam en wachtwoord
S8	Er moet een mogelijkheid zijn om de gegevens te rapporteren
S9	De LVR (Landelijke Verloskunde Registratie) vereist dat er periodiek een formulier wordt toegezonden met het aantal cliënten en extra gegevens, deze moet door het systeem kunnen worden gegenereerd.
S10	De instellingen binnen de applicatie moeten kunnen worden veranderd door een beheerder
S11	Gebruikers van het systeem moeten kunnen worden toegevoegd
S12	De cliëntgegevens moeten worden gecontroleerd aan de hand van de voorwaarden van het LVR
S13	Alle kosten die gemaakt kunnen worden in het cliënttraject moeten kunnen gerapporteerd

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13
Anamnese bijwerken	**X**												
Babygegevens invoeren	**X**												
Baring invoeren	**X**												
Begeleiders bewerken		**X**											
Client toevoegen	**X**												
Contactmoment bijwerken	**X**												
Gebruiker toevoegen							**X**				**X**		
Gegevens kraambed invoeren	**X**												
Inloggen							**X**						
Medici bewerken		**X**											
Onderzoek vastleggen	**X**												
Praktijk bewerken		**X**											
Rapporteren								**X**					**X**
Termijn instellen	**X**												
Toxische factoren bewerken	**X**												
Uitkomst bepalen	**X**												
Uitvoeren LVR-controle									**X**			**X**	
Verzekeraars bewerken		**X**											
Voornemens registreren	**X**												
Wegschrijven data						**X**							
Ziekenhuizen bijwerken		**X**		**X**									
Zoeken			**X**	**X**									

7.2. Verificatie samenhang use-case diagram en subsystemen

In de onderstaande tabel worden use-cases gecontroleerd aan de hand van de opgestelde subsystemen. Zijn alle use-cases ondergebracht in een subsysteem.

	Eerste contactmoment cliënt	Zwangerschap	Administratie	Beheren
Anamnese bijwerken	**X**			
Babygegevens invoeren		**X**		
Baring invoeren		**X**		
Begeleiders bewerken				**X**
Cliënt toevoegen	**X**			
Contactmoment bijwerken	**X**			
Gebruiker toevoegen				**X**
Gegevens kraambed invoeren		**X**		
Inloggen	**X**			
Medici bewerken				**X**
Onderzoek vastleggen	**X**			
Praktijk bewerken				**X**
Rapporteren			**X**	
Termijn instellen	**X**			
Toxische factoren bewerken	**X**			
Uitkomst bepalen	**X**			
Uitvoeren LVR-controle			**X**	
Verzekeraars bewerken				**X**
Voornemens registreren	**X**			
Wegschrijven data				**X**
Ziekenhuizen bijwerken				**X**
Zoeken		**X**		

7.3. Verificatie samenhang functionele eisen en pilots

Hieronder is een tabel opgenomen die de samenhang tussen de functionele eisen en de pilots in kaart brengt. In dit geval zullen de systeemeisen aan de linkerkant uitgezet worden met een code zoals deze ook in paragraaf 1 is gebruikt.

	GUI	Database	GUI-database Interface	Subsystemen1 tot en met 3	Subsysteem 4
S1			**X**	**X**	
S2		**X**			**X**
S3	**X**		**X**	**X**	
S4			**X**	**X**	
S5	**X**	**X**			
S6			**X**		**X**
S7	**X**		**X**	**X**	
S8			**X**	**X**	
S9	**X**			**X**	
S10		**X**	**X**		**X**
S11	**X**	**X**	**X**		**X**
S12		**X**	**X**	**X**	
S13	**X**		**X**	**X**	

7.4. Verificatie samenhang use-case diagram en klassendiagram

Deze paragraaf is de laatste in de serie van vier controles. Deze controleert de samenhang tussen het klassendiagram en het use-case diagram. Wat hier vooral van belang is, is dat de klassen die in het klassendiagram zijn opgesteld, ook daadwerkelijk gebruikt worden door de use-cases.
Om de overzichtelijkheid te bewaren, zijn de klassen aangegeven met een code. Deze code wordt in onderstaande tabel toegelicht.

Code	Klassenaam	Code	Klassenaam
K1	Werknemer	K15	Sociaal
K2	Client	K16	Baring
K3	Praktijk	K17	Voornemen
K4	Ziekenhuis	K18	Uitkomst
K5	Medici	K19	Ziektebeeld
K6	Contact	K20	Placenta
K7	Labuitslag	K21	Trauma
K8	Echo	K22	Tijd
K9	Onderzoek	K23	Aanwezig
K10	Aandoening	K24	Kind
K11	Anamnese	K25	Kraambed
K12	Obstetrisch	K26	Nacontrole
K13	Zwangerschap	K27	Verzekeraar
K14	Termijn		

	k1	k2	k3	k4	k5	k6	k7	k8	k9	k10	k11	k12	k13	k14
Anamnese bijwerken							X	X			X	X		
Babygegevens invoeren								X		X				
Baring invoeren										X				
Begeleiders bewerken	X													
Client toevoegen		X												
Contactmoment bijwerken						X							X	
Gebruiker toevoegen	X													
Gegevens kraambed invoeren										X				
Inloggen	X													
Medici bewerken					X									
Onderzoek vastleggen									X					
Praktijk bewerken			X											
Rapporteren														
Termijn instellen														X
Toxische factoren bewerken										X				
Uitkomst bepalen														
Uitvoeren LVR-controle														
Verzekeraars bewerken														
Voornemens registreren														
Wegschrijven data	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Ziekenhuizen bijwerken				X										
Zoeken		X		X	X									

	k15	k16	k17	k18	k19	K20	K21	K22	K23	k24	k25	k26	k27
Anamnese bijwerken	X												
Babygegevens invoeren										X			
Baring invoeren		X			X	X	X	X	X				
Begeleiders bewerken													
Client toevoegen													
Contactmoment bijwerken													
Gebruiker toevoegen													
Gegevens kraambed invoeren											X	X	
Inloggen													
Medici bewerken													
Onderzoek vastleggen													
Praktijk bewerken													
Rapporteren													
Termijn instellen													
Toxische factoren bewerken													
Uitkomst bepalen				X	X								
Uitvoeren LVR-controle													
Verzekeraars bewerken													X
Voornemens registreren			X										
Wegschrijven data	X	X	X	X	X	X	X	X	X	X	X	X	
Ziekenhuizen bijwerken													
Zoeken													

Bij controleren van de samenhang tussen het klassendiagram en de use-cases, valt direct op dat de use-cases: “Rapporteren” en “Uitvoeren LVR-Controle”, volgens de tabel geen enkele klassen gebruiken. Dit is uiteraard niet het geval. Deze twee use-cases gebruiken wel degelijk klassen, echter is ervoor gekozen om dit in een later stadium pas invulling te geven.

7.5. Validatie niet functionele systeemeisen

In deze paragraaf wordt er gekeken of de niet functionele systeemeisen haalbaar en reëel zijn.

- De grafische user interface moet overzichtelijk zijn
  Door het raadplegen van een online handleiding over gebruikersvriendelijkheid en het frequente contact met de opdrachtgever is deze systeemeis wel degelijk haalbaar.

- Er mogen niet meerdere gebruikers tegelijkertijd met het systeem werken
  Omdat het systeem afhankelijk is van de correctheid van de gegevens uit de database, is het niet toegestaan om met meerdere gebruikers tegelijk actief

bezig te zijn. Dit is wel degelijk haalbaar door te controleren of een applicatie met dezelfde naam al actief is.

- Het systeem moet werken op een werkstation met een Windows operating systeem
  De technische omgeving voldoet op dit moment aan deze eis.

- Het systeem zal dag en nacht moeten kunnen draaien.
  Dit is afhankelijk van de werknemers binnen de verloskundepraktijk. Deze eis zal voorgelegd worden met de voordelen hiervan. Deze eis zal niet kunnen worden nageleefd binnen dit project als er geen medewerking is.

- Registratie van eerste gesprek met client mag maximaal een half uur duren
  Door de grafische user interface en de acties zo te ontwikkelen dat deze snel kunnen worden toegepast, moet dit haalbaar zijn.

- Foutmeldingen moeten concreet en begrijpelijk zijn
  Hier zal bij de vormgeving van de subsystemen goed op gelet moeten worden.

Bijlage A: Plan van aanpak IAD Definitiestudie

Afbakening

Doelstellingen
Om te kunnen controleren of deze fase goed is uitgevoerd, zullen de onderstaand gedefinieerde doelstellingen bereikt zijn of is er toegelicht waarom een doelstelling niet geëvenaard is.

De doelstellingen van de definitiefase zijn de volgende, gedefinieerd door IAD:

- Overeenstemming bereiken over het te volgen ontwikkelscenario.
- Ervaring met eerdere pilots evalueren.
- De systeemeisen bepalen
- Deze eisen prioriteren
- Een globale oplossing in functionele termen opstellen in de vorm van een systeemconcept
- De technische en organisatieaspecten analyseren die met de systeemeisen en het systeemconcept verband houden.
- Het systeemconcept onderverdelen in componenten die als afzonderlijke, coherente pilots ontwikkeld kunnen worden.
- Een pilotplan definiëren, dat gebruikt wordt voor het besturen van het iteratieve ontwikkelproces.
- Dit plan valideren via risicoanalyse en kosten/batenanalyse
- De haalbaarheid van het project (her)bevestigen
- De volgende fase gedetailleerd plannen

Randvoorwaarden

Er is in de subparagraaf sprake van drie verschillende soorten randvoorwaarden, namelijk die van de opdrachtnemer, van de opdrachtgever en de methode IAD. In deze volgorde zullen ze hier ook behandeld worden.

Opdrachtnemer
De randvoorwaarden die gesteld zijn aan deze fase door de opdrachtnemer, zijn een manier om de kwaliteit van het eindproduct te kunnen valideren. Naast de validatie is het, samen met de doelstellingen, een richtlijn voor de juiste uitvoer van deze fase.

- Alle doelstellingen zijn geëvenaard, zo niet is er een onderbouwing aanwezig waarom deze doelstellingen niet behaald zijn.
- De definitiefase moet volledig doorlopen zijn, voordat met de volgende fase aangevangen mag worden.

Opdrachtgever
Naast de voorwaarden van de opdrachtgever over het gehele project, zijn er voor de definitiefase twee voorwaarden gesteld:

- De samenvattingen van de gesprekken zijn in de definitiefase verwerkt
- De definitiefase moet in de vorm van een rapport beschikbaar zijn voor de organisatie, een digitale kopie is toegestaan.

IAD
Om de definitiefase tot een methodisch correct resultaat te maken, en om aan het eind van het stadium te kunnen valideren is de voorwaarde dat de doelstellingen indien mogelijk bereikt zijn.

Risico's

Door het toepassen van risicomanagement tijdens het gehele project, zijn de risico's al in het globale plan van aanpak opgenomen. Voor een beschrijving van het risicomanagement wordt verwezen naar globale plan van aanpak.

Productoplevering

Resultaten
De resultaten van de definitiefase zijn hier opgesomd en toegelicht met de bijbehorende activiteiten om tot het gewenste product te komen. Deze zullen in de planning in hoofdstuk 4 worden toegelicht met de benodigde tijd per activiteit.

Op te leveren producten en de bijbehorende activiteiten:

- Het ontwikkelscenario
    - Beschouwing projectkenmerken
    - Bevestiging ontwikkelproces
    - Bevestiging projectorganisatie
    - Beschrijving globale pilotstrategie
- Pilotworkshop
    - Zet workshop-accoumodatie/administratie op
    - Bevestig deelnemers workshop
    - Installeer workshopomgeving
    - Prepareer conceptmateriaal workshop
- De definitie van systeemeisen
    - Bevestig reikwijdte workshop
    - Definieer basissysteemeisen
    - Definieer interface-eisen
    - Definieer integriteiteisen
    - Definieer performance-eisen
    - Definieer operationele eisen
    - Consolideer systeemeisen
    - Beoordeel acceptatiecriteria
    - Houd workshop-checkpoint
- Het systeemconcept
    - Specificeer globaal actoren
    - Specificeer globaal events
    - Specificeer globaal gegevensmodel
    - Beschrijf systeeminterfaces
    - Beschrijf operationele kenmerken
    - Maak prototype systeemconcept
    - Houd workshop-checkpoint
- Technische structuur
    - Beschouw technische architectuur
    - Beschouw technologische basis
    - Beschouw systeemconfiguratie
    - Onderzoek haalbaarheid technische aspecten
    - Definieer technische veranderingen
    - Stel herbruikbare componenten vast
    - Stel aanschaflijst op
    - Houd workshop-checkpoint
- Organisatorische inrichting
    - Beschouw werkstroom
    - Onderzoek organisatorische inrichting
    - Definieer opleidingsdoelstellingen
    - Definieer vereiste gebruikersdocumentatie

    - Houd workshop-checkpoint
- Het pilotplan
    - Definieer pilotstructuur
    - Structureer pilotontwikkeling
    - Prioriteer pilots
    - Stel pilotacceptatieplan op
    - Stel globaal invoeringsplan op
    - Stel globaal pilotontwikkelplan op
    - Houd workshop-checkpoint

Vervallen activiteiten:

- Evalueren Pilotplan

Naast de eerder genoemde activiteiten, is de activiteit evalueren pilotplan moedwillig weggelaten. In dit stadium is er nog geen pilot gemaakt en is het evalueren hiervan niet mogelijk.

Kwaliteitsborging

Om de kwaliteit van de op te leveren producten en het uit te voeren proces te kunnen verzekeren wordt in deze paragraaf de wijze beschreven waarop dit kan worden behaald. Allereerst zal de wijze beschreven worden waarop de kwaliteit van de producten kan worden gegarandeerd.

Product

Om de kwaliteit van de mijlpaalproducten te kunnen garanderen zullen alle bijbehorende activiteiten uitvoerig bestudeerd en correct uitgevoerd worden. De kwaliteit zal vervolgens bij de afronding voor ieder product worden getoetst, door te controleren of het product voldoet aan de eisen die gesteld zijn in het boek van IAD en door de diepgang van de activiteiten na te gaan.

Proces

Om de kwaliteit van de uit te voeren processen te kunnen verzekeren, zullen de procedures die beschreven staan in het boek van IAD worden uitgevoerd. Bovendien zal ieder proces goed ingepland worden, zodat voorkom worden dat deze niet / onvolledig worden uitgevoerd.

Uitvoering

Projectstructuur
De structuur van de projectgroep, de betrokkenen en de taken en verantwoordelijkheden zijn reeds verklaard in het globale plan van aanpak. In deze projectstructuur zal daarom alleen de interne communicatiemomenten worden toegelicht.

Interne communicatiemomenten
Er zal tussen het A-Team en het U-Team bij de start van ieder nieuw mijlpaalproduct een contactmoment plaats hebben. Dit zal in het stadium van de projectontwikkeling meerdere malen zijn.
Bij deze contactmomenten zal om feedback worden gevraagd over het opgeleverde mijlpaalproduct en zullen er indien nodig aanpassingen worden gemaakt.

De contactmomenten tussen het A-Team en het QA-Team kunnen op ieder tijdstip in het project plaats hebben. Indien een contactmoment nodig is zal de detailplanning in het plan van aanpak worden gewijzigd.

Planning

Hieronder volgt de planning, met de activiteiten van de definitiefase uitgewerkt.

Product / Proces	Tijdsduur in minuten
Het ontwikkelscenario	140
Pilotworkshop	880
De definitie van systeemeisen	2180
Het systeemconcept	480
Technische structuur	480
Organisatorische inrichting	160
Het pilotplan	1200
Verificatie en Validatie	700
Totaal in minuten	6220
Totaal in uren	+- 104

Bijlage 3
Pilotontwikkelrapport
Pilot 1: Grafische User Interface

Voorwoord

Voor u ligt het pilotontwikkelrapport dat ter ondersteuning dient bij de implementatie van de eerste pilot uit het project de verloskunde applicatie. Tijdens deze pilot wordt de grafische user interface ontwikkeld die de basis vormt voor de applicatie.

In het IAD ontwikkeltraject is dit de fase na de definitiefase. Deze fase is de pilotontwikkelingsfase, waarin alle gedefinieerde pilots uit de definitiefase worden gedetailleerd, gedetailleerd ontworpen en gebouwd.

Dit document is tot stand gekomen in overleg met een verloskundige van de praktijk, Mevr S. van Iterson.

Leiden,

Cees Kruijf
Verloskundepraktijk de Maatschap

Samenvatting

De eerste pilot in het ontwikkeltraject van de pilots, is het ontwerpen en implementeren van de Grafische User Interface. Het doel van dit rapport is om de gedefinieerde pilot in de definitiestudie in dit document dusdanig te ontwerpen, dat bouw hiervan mogelijk moet zijn. Daarnaast worden er testspecificaties opgesteld, die na implementatie de delen kunnen valideren en verifiëren. Op basis van die resultaten kan worden bepaald hoe de ontwikkeling van de pilot is verlopen.

Deze pilot is opgedeeld in enkele pilotdelen. Deze pilotdelen zijn daarna onderverdeeld in bouweenheden. Bouweenheden kunnen worden beschouwd als opzichzelfstaande componenten, die onafhankelijk kunnen worden ontwikkeld en geïmplementeerd. De volgende bouweenheden zijn gedefinieerd:

1. Opzetten ontwikkelomgeving en bepalen naamgeving
2. Ontwikkelen scherm voor inloggen
3. Ontwikkelen van het hoofdscherm
4. Ontwikkelen van het clientscherm

Deze bouweenheden zijn ontwikkeld en geïmplementeerd. Na implementatie zijn de tests uitgevoerd. Het resultaat van dit document

Inhoudsopgave

1. Inleiding

Tijdens de definitiefase is het project ‘W!se delivery’ verdeeld in vijf pilots. Deze pilots worden tijdens de pilotontwikkelfase sequentieel ontwikkeld. De eerste pilot bestaat uit het ontwikkelen en implementeren van de Grafische User Interface voor de applicatie.

Het doel van dit rapport is om het systeemconcept in het rapport definitiestudie op het gebied van de user interface in detail uit te werken. Daarnaast wordt er in dit document de samenhang tussen de diagrammen gecontroleerd. Hiermee kunnen de geïmplementeerde bouweenheden worden getest op het gewenste resultaat. Op basis van deze resultaten is een eindoordeel over de pilot uitgesproken

In hoofdstuk 2 wordt de functionele structuur toegelicht. De technische structuur wordt in het daaropvolgende hoofdstuk beschreven. Vervolgens wordt het pilotontwikkelplan opgesteld, waarna de gedefinieerde bouweenheden worden ontworpen in hoofdstuk 5. Na dit hoofdstuk is een hoofdstuk opgenomen die de gebruikersvriendelijkheid beschrijft. In hoofdstuk 6 wordt de invoeringsprocedure van de pilot toegelicht. Het laatste hoofdstuk, 8, beschrijft de resultaten van pilot..

2. Functionele structuur

Het doel van dit hoofdstuk is om de functionele inhoud dusdanig te ontwerpen dat deze als basis kan dienen voor het ontwerp van de pilot. Deze pilot betreft het ontwerp van de GUI. De GUI zal zelf functionele toevoeging hebben op het systeemconcept en in de volgende paragrafen zullen daarom een aantal activiteitsdiagrammen worden opgenomen die een indruk moeten geven over hoe de interactie met het systeem zal gaan verlopen. Allereerst zal de functionele reikwijdte van de pilot worden toegelicht.

2.1. Functionele reikwijdte

De functionele reikwijdte bepaald in hoeverre de gestelde functionele systeemeisen worden behandeld. Er zal bij de kernel-pilot gelet worden op dat met het ontwerp alle functionele systeemeisen geïmplementeerd kunnen worden in latere pilots, dat wil zeggen dat de interface voor deze functies aanwezig is.

Naast de systeemeisen, zal er bij het ontwerp gelet worden op de niet-functionele systeemeisen.

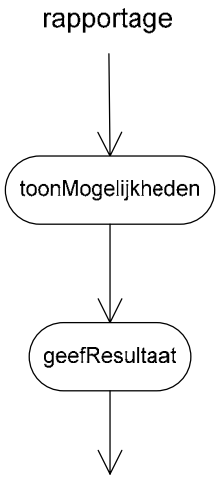
- *De grafische user interface moet overzichtelijk zijn*
- *Er mogen niet meerdere gebruikers tegelijkertijd met het systeem werken*
- *Het systeem moet werken op een werkstation met een Windows operating systeem*
- *Het systeem zal dag en nacht moeten kunnen draaien.*
- *Registratie van eerste gesprek met cliënt mag maximaal een half uur duren*
- *Foutmeldingen moeten concreet en begrijpelijk zijn*

2.2. Activiteitsdiagrammen

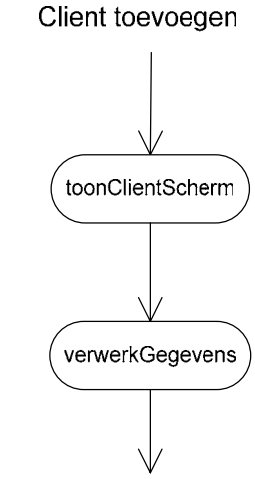
In het rapport van de definitiestudie zijn use-cases opgenomen, waarmee de interactie tussen de gebruiker en het systeem is gemodelleerd. Deze use-cases vormen het uitgangspunt van het modelleren van de UML-activiteitsdiagrammen.
Het activiteitsdiagram geeft inzicht in hoe een proces verloopt en welke subprocessen en /of beslissingen daarbij moeten worden uitgevoerd.
Voor iedere use-case van de gebruiker, behalve uitvoeren van LVR-Controle, is een activiteitsdiagram opgenomen. De reden hiervoor is dat de LVR-Controle geen apart scherm heeft. Deze diagrammen zullen in de desbetreffende pilots verder worden uitgewerkt.
De activiteitsdiagrammen van de use-cases van het beheer zijn bewust niet opgenomen, omdat deze geen toegevoegde waarde hebben voor het ontwerp. Dit zal namelijk bestaan uit één scherm.

Rapportage

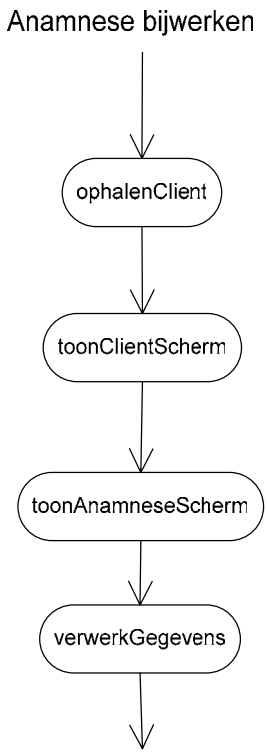
Hieronder is het activiteitsdiagram van rapportage opgenomen.



Client toevoegen

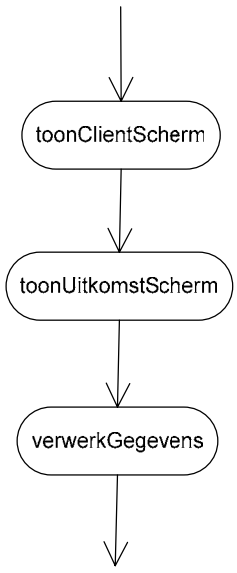


Anamnese bijwerken

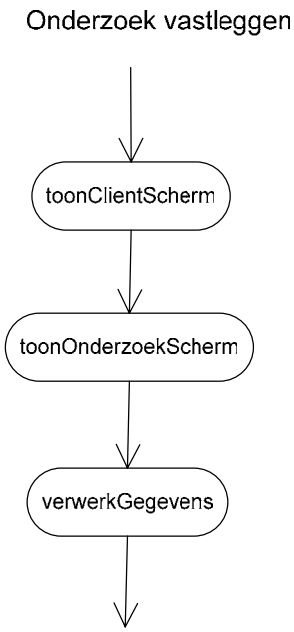


Uitkomst bepalen

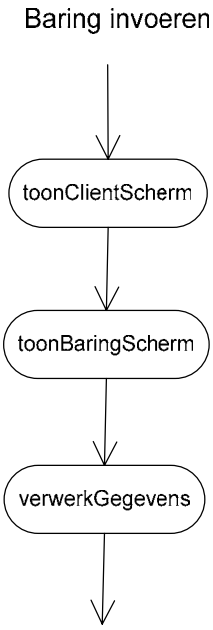
Uitkomst bepalen



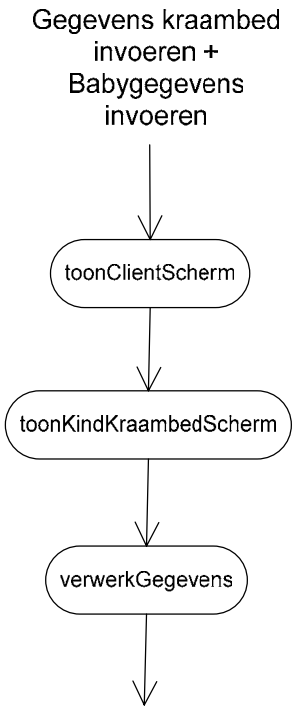
Onderzoek vastleggen



Baring invoeren

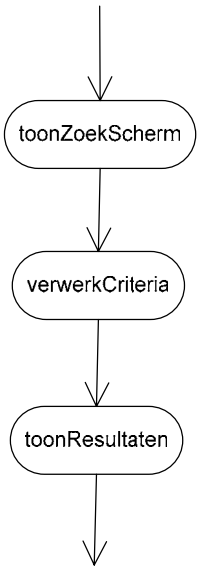


Gegevens kraambed en babygegevens invoeren

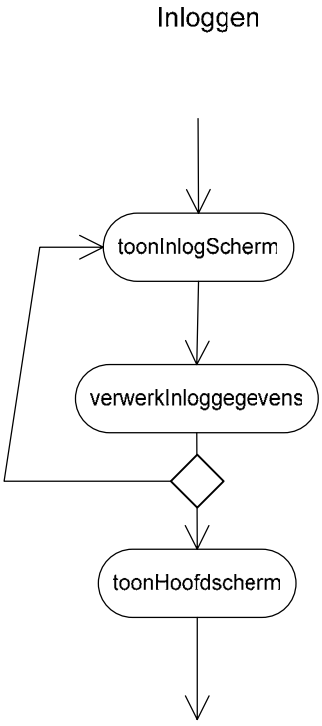


Zoeken

Zoeken



Inloggen


Inloggen
toonInlogScherm
verwerkInloggegevens
toonHoofdscherm

3. Technische structuur

Dit hoofdstuk beschrijft de technische aspecten met betrekking tot de ontwikkeling van de eerste pilot. Deze aspecten zijn reeds globaal gedefinieerd in de definitiestudie. Voor deze pilot zullen de technische aspecten vooral beperkt worden tot het beschrijven van de manier waarop de diagrammen in het voorgaande hoofdstuk invulling gaan krijgen in de applicatie. Deze invulling zal betrekking hebben op de schermen voor de gebruiker en de naamgeving hiervan.

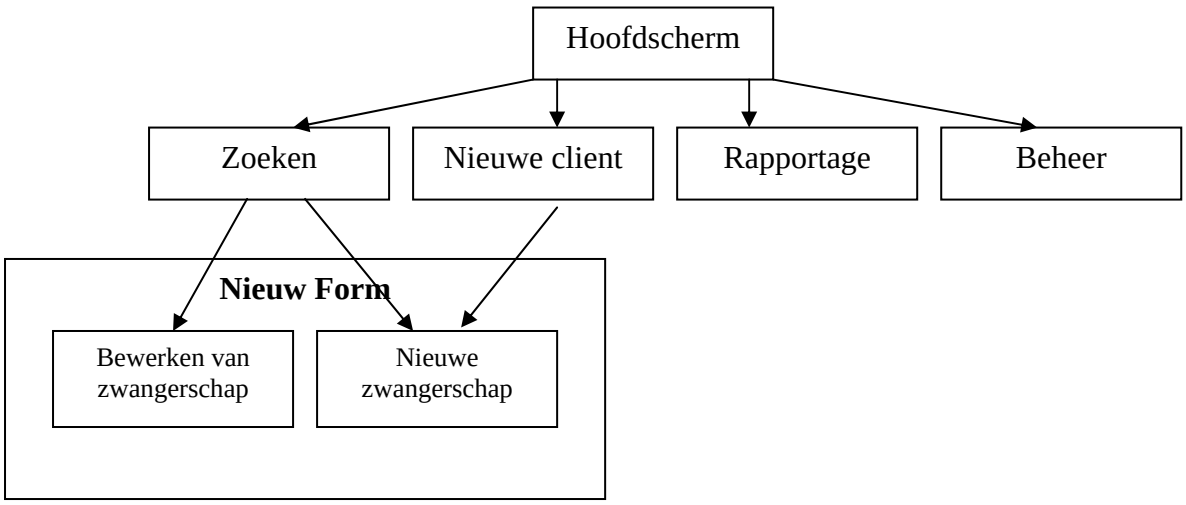
3.1. Vormgeving

Het ontwerp in deze pilot houdt in dat er gekeken wordt naar het klassendiagram, de activiteitsdiagrammen en de gebruikervriendelijkheid. Op basis hiervan wordt een structuur en uiteindelijk de GUI gemaakt.
De eventuele verbeteringen in het klassendiagram zullen in de volgende pilot worden gedocumenteerd, omdat in die pilot de database ontworpen gaat worden en deze bijna gelijk is aan het klassendiagram.

Door de structuur van de Grafische User Interface vast te leggen, wordt bepaald hoe bepaalde schermen aangeroepen gaan worden en welke gegevens er op de schermen zullen moeten komen te staan.

3.2. Basisstructuur

De GUI zal bestaan uit een hoofdscherm waarin de navigatie wordt ondergebracht. Deze navigatie zal het beheer, het zoeken, het aanmaken van cliënten en de rapportage omvatten.



Zoals hierboven te zien is, moeten de zwangerschappen en bijbehorende aspecten in een nieuw scherm (form in Delphi, ClientForm vanaf nu) worden vastgelegd. Deze optie zal het mogelijk maken om meerdere cliënten tegelijkertijd te kunnen bijwerken, zonder de huidige cliënt te sluiten.

ClientForm

Het cliëntscherm zal een nieuw navigatiemenu bevatten door middel van een menu. Alle onderdelen van de cliënt zullen verdeeld worden onder een aantal gegroepeerde frames. Deze frames worden verborgen en getoond afhankelijk van de optie die in het menu wordt aangeklikt. De groepsnamen zijn allereerst in secties verdeeld en hebben bepaalde code toegekend gekregen, zodat de tabellen overzichtelijk blijven

Code	Groepsnaam
G1	Anamnese
G2	Partus
G3	Client
G4	Contact
G5	EchoLab
G6	LosKraambed
G7	Verwijzingen
G8	PrenataleZorg

Klassenaam	G1	G2	G3	G4	G5	G6	G7	G8
Algemeen	**X**							
Client			**X**					
Contactmoment				**X**				
Echo					**X**			
Familie	**X**							
KindKraambed		**X**						
Labuitslag					**X**			
Los Kraambed						**X**		
Obstetrisch	**X**							
Onderzoek				**X**				
Partner	**X**		**X**					
Partus		**X**						
Prenatale_zorg								**X**
Termijn				**X**				
Verwijzingen							**X**	
Verzekeraar			**X**					
Zwangerschap				**X**			**X**	

Deze groepen met een bepaalde code zullen terugkomen in de GUI van de cliënt

4. Pilotontwikkelplan

Het pilotontwikkelplan beschrijft hoe de pilot gaat worden gebouwd. Het pilotplan dat tijdens de definitiestudie is gemaakt, wordt hier uitgebreid en eventueel aangepast. Per pilotdeel worden één of meer bouweenheden gedefinieerd. Bouweenheden zijn componenten van de pilot die door hun aard en reikwijdte achtereenvolgens of parallel kunnen worden ontwikkeld.
In paragraaf 4.1 worden er pilotdelen opgesteld en onderverdeeld in bouweenheden en wordt aangegeven in welke volgorde de bouweenheden worden ontwikkeld. Paragraaf 4.2 beschrijft de ontwikkelaanpak.

4.1. Pilotdelen en bouweenheden

Om de pilot gestructureerd te kunnen ontwikkelen, zijn deze onderverdeeld in pilotdelen. Om vast te kunnen stellen wat precies een pilotdeel is en welke onderdelen daarmee gepaard gaan, is er gekeken naar de verschillende vensters..

In het vorige hoofdstuk is te zien hoe de basisstructuur van het hoofdscherm en hoe het clientscherm eruit moet gaan zien. Aan de hand hiervan zijn de drie onderstaande pilotdelen vastgesteld:

5. Opzetten ontwikkelomgeving en bepalen naamgeving
6. Ontwikkelen scherm voor inloggen
7. Ontwikkelen van het hoofdscherm
8. Ontwikkelen van het clientscherm

In de aankomende subparagrafen worden de bouweenheden van de pilotdelen gedefinieerd.

4.1.1. Opzetten ontwikkelomgeving

1. Onderzoeken ontwikkelomgeving
2. Bepalen programmeeromgevingen
3. Bepalen naamgeving objecten

4.1.2. Ontwikkelen scherm voor inloggen

1. Bepalen schermattributen
2. Ontwerp inlogForm

4.1.3. Ontwikkelen van het hoofdscherm

1. Bepalen schermattributen
2. Ontwerp hoofdForm

4.1.4. Ontwikkelen van het clientscherm

1. Bepalen schermattributen
2. Ontwerp clientForm
3. Ontwerpen groepen gedefinieerd in basisstructuur

4.2. Ontwikkelaanpak

Om de GUI gestructureerd te kunnen ontwikkelen, worden de bouweenheden zoals gedefinieerd ontwikkeld. Per pilotdeel worden eerst alle bouweenheden geïmplementeerd, voordat er met de volgende bouweenheid begonnen mag worden. Deze wijze van ontwikkeling zorgt ervoor dat niets wordt overgeslagen en eventuele knelpunten direct worden afgehandeld.

Na het ontwikkelen van alle pilotdelen zal het in een later stadium kunnen voorkomen dat er aanpassingen gedaan moeten worden aan een of meerdere pilotdelen.

5. Ontwerp software-bouweenheden

De bouweenheden gedefinieerd in het vorige hoofdstuk, worden in de hoofdstuk uitgewerkt op inhoud. De eerste paragraaf zal de bouweenheden detailleren en paragraaf 5.2 zal ingaan op de testspecificaties.

5.1. Bouweenheden

In de aankomende subparagrafen zijn de bouweenheden gespecificeerd, echter is er voor gekozen om de bouweenheden voor het ontwikkelen van een scherm te groeperen tot een subparagraaf. Aangezien de werkwijze bij deze bouweenheden ongeveer hetzelfde zal zijn.

5.1.1. Onderzoek ontwikkelomgeving

Voor het opstellen van de opdrachtomschrijving in het beginstadium is gekozen voor het gebruik van Borland Delphi 7. Om in dit stadium te verzekeren dat deze taal de best toepasbare taal is om de applicatie in te ontwikkelen is een klein onderzoek gedaan naar de mogelijkheden.
Het resultaat van dit onderzoek zal onderbouwen waarom de gekozen programmeeromgeving het beste alternatief is ten opzichte van de rest.

Vaststellen alternatieven
Allereerst is een lijst opgesteld met talen waarin de applicatie in geschreven kon worden. Deze talen moesten aan enkele voorwaarden voldoen:

- Stand-alone kunnen draaien
- Ondersteuning van een Grafische User Interface
- Mogelijkheid tot het aanspreken van een DBMS.

Programmeertaal
Delphi
C++
C
Visual Basic
Turbo Pascal

Deze alternatieven zijn aan de hand van een aantal criteriapunten vergelijken en door per taal te bekijken in welke mate deze voldeed aan de punten is gebleken welke taal het meest geschikt is. De moderatie loop van -- (zeer slecht) tot ++ (zeer goed)

	Delphi	C++	C	Visual Basic	Turbo Pascal
Ondersteund Object georiënteerd programmeren	++	++	+	--	--
Kan zonder extra bibliotheekbestanden op iedere pc functioneren	+	++	++	-	++
Is bekend bij de projectgroep	++	--	++	++	+
Kan een Windows-GUI opleveren zonder extra	++	++	++	++	--

bibliotheekbestanden					
Heeft de mogelijk tot het gebruik ODBC	++	++	-	-	-

5.1.2. Bepalen programmeeromgevingen

Delphi komt als geschikte programmeertaal het beste uit de vergelijkingen. Daardoor is de lijn met Borland snel getrokken. Borland is de beste leverancier op het gebied van een programmeeromgeving voor Delphi. Dit komt omdat zij een studenten regeling hebben. Deze regeling betreft het gratis gebruik maken van de Borland Delphi programmeeromgevingen zolang de tot stand gekomen producten niet voor commerciële doeleinden worden gebruikt.

5.1.3. Bepalen naamgeving objecten

Om de applicatie overdraagbaar te maken, is er voor gekozen om de objecten een bepaalde naamgeving te geven. Deze naamgeving houdt in dat de eerste drie letters van ieder nieuw object refereren naar dat object. Na deze drie letters volgt een woord dat aangeeft in welk deel van het programma het zich bevindt. Hieronder staan enkele voorbeelden hiervan:

- btnKraamBed_vastleggen.
  Dit object is van het type btn (TButton) en legt de gegevens vast van het kraambed.
- grpOnderzoek
  Dit object is van het type grp (TGroup) en is een groep met een aantal gegevens met betrekking tot onderzoek

5.1.4. Bepalen schermattributen

Om de schermattributen te kunnen bepalen, moet eerst worden vastgelegd wat schermattributen zijn. Met schermattributen worden attributen zoals hoogte, breedte, kleur, titel, functionaliteit op de titelbalk en positie van de buttons (knoppen).

De hoogte en breedte is voor bijna ieder scherm gelijk, behalve voor het inlogscherm. Dit scherm is een klein scherm en heeft de hoogte en breedte van de huidige applicatie.
De overige schermen zijn na een klein onderzoekje naar de gebruikte resolutie en de mogelijkheden van de systemen vastgesteld op 1024 breedte en 768 hoogte maximaal.

Voor ieder scherm in de applicatie geldt dat de buttons voor bevestiging zich zoveel als mogelijk links onderin bevinden en voor annulering bevinden deze zich rechts onderin.

Om de functies van de applicatie overzichtelijker te maken, is ervoor gekozen om de groepen een kleur te geven. Deze kleur bepaalt dan uiteindelijk welke functionaliteit deze behuisd. Dit zal in een later stadium ook makkelijker zijn voor de gebruikershandleiding.

5.1.5. Ontwerpen groepen gedefinieerd in basisstructuur

Met het ontwerpen van de groepen wordt bedoeld dat deze worden omgezet naar een TGroupBox in Delphi. Deze groepen bevatten, zoals eerder gedefinieerd een aantal klassen uit het klassendiagram.
Het voordeel van deze groepering is dat het overzichtelijk is.
De volgende groepen zullen worden ontwikkeld:

- grpAnamnese
- grpPartus
- grpClient
- grpContact
- grpEchoLab
- grpKindKraambed
- grpLosKraambed
- grpPrenataleZorg
- grpVerwijzingen

De kleuren die bij iedere groep gebruikt zullen worden zijn hieronder in tabelvorm opgenomen:

Groepsnaam	Kleurcode
grpAnamnese	clSkyBlue
grpPartus	clLime
grpClient	clInactiveCaptionText
grpContact	clTeal
grpEchoLab	$00FF008A (Purple)
grpKindKraambed	clOlive
grpLosKraambed	clButtonFace
grpPrenataleZorg	clWhite
grpVerwijzingen	clCream

De attributen die in de klassen zijn opgenomen en betrekking hebben op de desbetreffende groep zullen moeten kunnen worden opgevraagd of verantwoord. Dit houdt in dat al deze attributen in logische volgorde ingedeeld moeten worden in de applicatie. Deze logische volgorde wordt aan de hand van prototyping in overleg met de opdrachtgever bepaald.

5.2. Testspecificaties

Normaliter zijn de testspecificaties gedefinieerd aan de hand van de bouweenheden. Omdat het bij deze pilot een GUI betreft is ervoor gekozen om de tests te specificeren als een soort feedbacktests. Een feedbacktest zal inhouden dat de GUI wordt voorgelegd aan het U-team en er op deze manier feedback verkregen wordt.

5.2.1. Look & Feel – test

De look&feel test is een serie tests die uit moeten wijzen of er met de GUI fijn gewerkt kan gaan worden. Dit fijn werken zal inhouden dat samen met het U-team de tests zullen worden uitgevoerd, het U-team is tenslotte het team dat met de toekomstige applicatie zal moeten gaan werken.

De onderstaande tests dienen te worden uitgevoerd op alle geïmplementeerde pilotdelen.

Testspecificatie Look&Feel	
Case #	**Test**
1.0	De buttons zitten voor de gebruiker op de juiste plek
1.1	De schermopvolging is logisch en makkelijk bereikbaar
1.2	De schermen zijn overzichtelijk en er is makkelijk mee te werken
1.3	De naamgeving van de labels is logisch
1.4	De groepen hebben een juiste kleur toegewezen gekregen

5.2.2. Gebruikersvriendelijkheid – test

Deze test zal uitwijzen of de toekomstige applicatie gebruikersvriendelijk is, in de ogen van de gebruiker. Dit is een verschil met gebruikersvriendelijkheid volgens de bouwer.

De onderstaande tests dienen te worden uitgevoerd op de geïmplementeerde GUI en in samenwerking met het U-team.

Testspecificatie Gebruikersvriendelijkheid	
Case #	**Test**
2.0	Het taalgebruik is helder en correct ABN
2.1	Het aantal vereiste muisklikken om bij een scherm te komen is tot een minimum beperkt.
2.2	Er is altijd een navigatie aanwezig
2.3	De naamgeving van de navigatie is veelal hetzelfde en de achterliggende functies hebben een logisch resultaat
2.4	Op ieder scherm is duidelijk aangegeven op welke gegevens het scherm betrekking heeft

6. Gebruikersvriendelijkheid

Om het gebruik van de applicatie te stimuleren en het gemak hiervan te bevorderen is hiervoor documentatie geraadpleegd. Naast deze documentatie is de opdrachtgever geïnterviewd over waar rekening mee moet worden gehouden in de toekomstige applicatie, waar de huidige applicatie tekortschiet.

6.1. Richtlijnen documentatie

Volgens deze documentatie moet bij het programmeren van iedere applicatie enkele richtlijnen worden nagestreefd:

- Gebruik heldere taal (dat wil zeggen, de taal voor de gebruiker)
- Wissel zo min mogelijk van schermen
- Zorg voor een vaste navigatie
- Gebruik zoveel mogelijk standaarden (voor zowel navigatie als interface)
    - Voor de navigatie gebruik Help in plaats van Hulp
    - Voor de interface, plaats de buttons zoveel mogelijk op een vaste plek.
- Foutmeldingen moeten niet te wensen overlaten en moeten in het uiterste noodgeval worden getoond. Teveel foutmeldingen wekt irritatie op.
- Zorg voor veel terugkoppeling, indien de applicatie bezig is laat dit dan ergens zien.
- Bij het afsluiten vraag dan altijd om bevestiging, een gebruiker kan per ongeluk ergens op hebben geklikt.
- Een GUI moet overzichtelijk zijn. Overzichtelijk wil zeggen dat de gebruiker weet wat deze aan moet met de gegevens.

6.2. Knelpunten huidige applicatie

De richtlijnen volgens de GUI zijn redelijk allesomvattend. Naast deze richtlijnen zijn er ook een aantal aandachtspunten in het interview naar voren gekomen. Deze aandachtspunten zijn vooral irritatiepunten in de huidige applicatie.

- Foutmeldingen laten te wensen over.
- Bij elk verplicht veld verschijnt een ‘messagebox’ indien de ingevulde data niet voldoet, dit wekt irritatie op.
- De terugkoppeling aan de gebruiker is in een onprofessionele taal geschreven
- Er moet heel veel ‘geklikt’ worden in het scherm om ergens te kunnen komen.

7. Invoeringsprocedure pilot

Wanneer de pilot functioneert volgens de opgestelde eisen en de testgevallen succesvol heeft doorlopen, kan worden overgegaan tot het operationeel maken van de pilot. Dit hoofdstuk beschrijft de procedure die dient te worden gevolgd wanneer de pilot wordt ingevoerd. In paragraaf 7.1 wordt een plan opgesteld dat als uitgangspunt dient bij de invoering van de pilot. Vervolgens worden in paragraaf 7.2 testspecificaties opgesteld. De testgevallen kunnen worden uitgevoerd nadat de pilot is ingevoerd en dienen ter controle op een correcte invoering.

7.1. Invoeringsplan

In deze subparagraaf is een invoeringsplan geschreven die toelichting geeft op de correcte invoering van de pilot:

- Controleren aanwezigheid benodigde hardware en software
  Voordat de pilot kan worden ingevoerd, moet eerst worden gekeken of:
  - De bibliotheekbestanden van Delphi reeds aanwezig zijn.
  - Het werkstation voldoende harde schijf ruimte heeft
  - Het DBMS MySQL geïnstalleerd is.
- Controleren werking applicatie
  Vervolgens zal de werking gecontroleerd worden, dit zal gedaan worden door de in de volgende paragraaf gedefinieerde testspecificaties

7.2. Testspecificaties

Hieronder is een lijst met testspecificaties opgenomen die betrekking hebben op het invoeringsplan.

Testspecificatie Invoeringsplan			
Case #	**Test**	**Invoer**	**Verwachte uitvoer**
3.0	Bibliotheekbestanden zijn aanwezig	Controleren of de bibliotheekbestanden aanwezig zijn op het werkstation	De bestanden zijn aanwezig
3.1	Voldoende harde schijf ruimte	Controleren of het werkstation voldoende ruimte heeft	Het werkstation heeft voldoende ruimte
3.2	MySQL is aanwezig	Controleren of het DBMS MySQL reeds geïnstalleerd is.	MySQL is aanwezig en functioneert naar behoren

8. Testen en beoordelen Pilot

Na de implementatie van iedere bouweenheid zijn deze getest aan de hand van een aantal testspecificaties uit hoofdstuk 5. Dit hoofdstuk beschrijft deze tests en diens resultaten en licht waar nodig de mislukte cases toe.

8.1. Look&Feel

Case #	Test	Resultaat
1.0	De buttons zitten voor de gebruiker op de juiste plek	OK
1.1	De schermopvolging is logisch en makkelijk bereikbaar	OK
1.2	De schermen zijn overzichtelijk en er is makkelijk mee te werken	OK
1.3	De naamgeving van de labels is logisch	Test mislukt
1.4	De groepen hebben een juiste kleur toegewezen gekregen	Test mislukt

Na het uitvoeren van de eerste vijf testcases, waren de eerste drie zonder verdere problemen doorlopen. De laatste twee hadden echter een negatief resultaat. De naamgeving voor de labels was in sommige gevallen onduidelijk voor de verloskundigen en gaven geen concrete toelichting op wat er in het invoerveld ingevuld moest worden.

De laatste twee cases kunnen door middel van een korte termijn oplossing worden afgehandeld.

Mislukte Case # 1.3
De naamgeving voor de labels is aangepast om een concretere beschrijving te geven van de ingevulde waarde. Dit heeft er toe geleid dat sommige groepen opnieuw moesten worden ingedeeld.

Mislukte Case # 1.4
De kleur van de groep “Anamnese” was op een beeldscherm met lage resolutie onleesbaar geworden. De oplossing hiervoor was de kleur aanpassen tot de kleur clSkyBlue.

8.2. Gebruikersvriendelijkheid

Het resultaat van de cases omtrent de gebruikersvriendelijkheid zijn in deze paragraaf verwerkt.

Case #	Test	Resultaat
2.0	Het taalgebruik is helder en correct ABN	OK
2.1	Het aantal vereiste muisklikken om bij een scherm te komen is tot een minimum beperkt.	OK
2.2	Er is altijd een navigatie aanwezig	OK
2.3	De naamgeving van de navigatie is veelal hetzelfde en de achterliggende functies hebben een logisch resultaat	OK
2.4	Op ieder scherm is duidelijk aangegeven op welke gegevens het scherm betrekking heeft	OK

De resultaten van de testspecificaties voor de gebruikersvriendelijkheid zijn uitgevoerd in overleg met de opdrachtgever. Het resultaat voor iedere case is positief.

8.3. Invoeringsplan

De resultaten van de cases van het invoeringsplan zijn hier verwerkt.

Case #	Test	Invoer	Verwachte uitvoer	Resultaat
3.0	Bibliotheekbestanden zijn aanwezig	Controleren of de bibliotheekbestanden aanwezig zijn op het werkstation	De bestanden zijn aanwezig	OK
3.1	Voldoende harde schijf ruimte	Controleren of het werkstation voldoende ruimte heeft	Het werkstation heeft voldoende ruimte	OK
3.2	MySQL is aanwezig	Controleren of het DBMS MySQL reeds geïnstalleerd is.	MySQL is aanwezig en functioneert naar behoren	Mislukt

Het resultaat van de eerste twee cases voor het invoeringsplan zijn positief.

Mislukte Case # 3.2
Er was reeds een versie van MySQL aanwezig op het werkstation, echter was dit een verouderde versie en ondersteunde het opslagtype InnoDB niet. Daarom is besloten om een gratis versie te installeren die deze mogelijkheid wel ondersteund.

8.4. Beoordeling pilot

Naast het mislukken van de testcases #1.3, #1.4 en #3.2, kon na het invoeren van de oplossing, de pilot als succesvol worden beoordeeld. Het gewenste resultaat zoals vastgesteld in het rapport van de definitiefase was behaald. Met deze beoordeling is tevens de ontwikkeling van deze pilot afgerond en kan het pilotontwikkeltraject worden vervolgd.

Conclusie

In dit document is een ontwerp gemaakt dat heeft gediend ter ondersteuning van de implementatie van de eerste pilot in het pilotontwikkeltraject.
De resultaten van de tests hebben uitgewezen dat de pilot voldoet aan de gestelde eisen. Dit document zal verder ondersteuning bieden aan de aankomende documenten in het kader van het pilotonwikkeltraject.

Bijlage A: Plan van aanpak pilot 1 'GUI-design'

Inleiding

In dit document worden de onderdelen van de eerste pilot toegelicht en in een planning opgenomen. Dit plan van aanpak is een detailplanning voor de eerste pilot en toevoeging op het plan van aanpak voor het gehele project.
Allereerst zullen de resultaten die pilot oplevert worden beschreven en vervolgens zal een planning gemaakt worden voor de uitvoering van de pilot.
Voor informatie over kwaliteitsborging en de projectorganisatie wordt verwezen naar het plan van aanpak voor het gehele project.

Producten en activiteiten pilotontwikkeling

Hieronder volgt een overzicht van de activiteiten die worden uitgevoerd tijdens de ontwikkeling van de eerste pilot. Per activiteit worden deelactiviteiten beschreven en er wordt aangegeven wat het resultaat is van deze activiteit.

- Specificeren globaal-functionele structuur pilot
    - Bepalen functionele reikwijdte pilot;
    - Detailleren GUI schermen door middel van sequencediagram(men);
    - Ontwerpen prototype gebruikersinterface;
    - Resultaat: ontwerp van functionele structuur.
- Specificeren globaal-technische structuur pilot
    - Bepalen fysieke allocatie
    - Specificeren technisch opslagmodel
    - Resultaat: ontwerp van technische structuur.
- Opstellen pilotontwikkelplan
    - Definiëren pilotdelen
    - Definiëren bouweenheden
    - Synchroniseren bouweenheden
    - Voorbereiden beoordeling en testen pilotdelen;
    - Resultaat: ontwikkelplan voor pilot.
- Ontwerpen software-bouweenheden
    - Specificeren integratie en testen bouweenheden
    - Opstellen detailspecificaties
    - Opstellen testspecificaties
    - Resultaat: gedetailleerd ontwerp pilotdelen.
- Bouwen software-bouweenheden
    - Coderen pilotdelen
    - Voorbereiden component-test
    - Uitvoeren component-test
    - Corrigeren component
    - Resultaat: implementatie van pilot.
- Opstellen invoeringsprocedure pilot
    - Ontwerpen invoeringsprocedure
    - Beschrijven verantwoordelijkheden rond invoering
    - Specificeren tests invoeringsprocedures
    - Testen invoeringsprocedures
    - Corrigeren invoeringsprocedures

    - Resultaat: beschrijving invoeringsprocedure.
- Beoordelen en testen pilot
    - Opzetten testomgeving
    - Beoordelen en testen pilot
    - Analyseren uitkomsten
    - Corrigeren fouten
    - Rapporteren over beoordelen en testen

Naast deze stappen zijn een aantal volgens IAD voorgeschreven ontwikkelstappen bewust overgeslagen. Deze stappen zullen geen toegevoegde waarde leveren op het ontwerp of de ontwikkeling van de pilot.

De overige activiteiten die IAD voorschrijft worden niet voor dit product uitgevoerd.

- Voorbereiden pilotontwerp-workshop
  Een workshop voor de pilot is reeds gehouden in de definitiefase.
- Specificeren globaal-organisatorische inrichting
  De invloed op de organisatorische aspecten zijn beschreven in de definitiefase. Deze pilot levert geen extra veranderingen met zich mee..
- Aanpassen externe componenten
  Er hoeven geen externe componenten te worden aangepast.
- Wijzigen andere informatiesystemen
  Er is geen sprake van een ander informatiesysteem.
- Bouwen conversie-tools
  Deze pilot eist geen conversie-tools
- Ontwerpen handmatige procedures
  De handmatige procedures zijn hier niet van toepassing.
- Integreren bouweenheden
  De bouweenheden worden bij implementatie direct geïntegreerd, doordat deze onderdeel zijn van de applicatie.
- Samenstellen opleidingsmateriaal pilot
  Er is geen behoefte aan opleidingsmateriaal voor enkel deze pilot.
- Samenstellen handleiding pilot
  Deze activiteit is opgenomen tijdens de laatste pilot

<u>Planning</u>
Hieronder is de planning voor de pilot opgenomen.

Product / Proces	Tijdsduur in uren
Specificeren globaal-functionele structuur pilot	15
Specificeren globaal-technische structuur pilot	15
Opstellen pilotontwikkelplan	8
Ontwerpen software-bouweenheden	15
Bouwen software-bouweenheden	25
Opstellen invoeringsprocedure pilot	5
Beoordelen en testen pilot	2
Totaal aantal uren	95

Bijlage 4
Pilotontwikkelrapport
Pilot 2: Database

Voorwoord

Voor u ligt het ontwikkelrapport dat ter ondersteuning dient bij de implementatie van de tweede pilot uit het project *W!se Delivery*. Binnen deze pilot wordt de database en de bijbehorende structuur ontworpen.

In het systeemontwikkelingtraject van IAD volgt na de definitiefase, de fase van de pilotontwikkeling. Tijdens de pilotontwikkeling worden de pilots die in de definitiefase zijn gedefinieerd, ontwikkeld en geïmplementeerd.

Dit document is tot stand gekomen in overleg met Mevr. S. van Iterson, opdrachtgever bij de verloskundepraktijk de Maatschap.

Leiden,

Cees Kruijf
Verloskundepraktijk de Maatschap.

Inhoudsopgave

1. Inleiding

Tijdens de definitiestudie is het project verdeeld in vijf pilots. Deze pilots worden in het pilotontwikkelingstraject sequentieel ontwikkeld. De eerste pilot bestond uit het ontwerpen en ontwikkelen van de Grafische User Interface. Na het ontwikkelen van de eerste pilot, was de volgende pilot in het ontwikkeltraject het ontwikkelen van de database. Dit document dient ter ondersteuning van de ontwikkeling van tweede pilot.

Het doel van dit rapport is om de databasestructuur te ontwerpen, aan de hand van het opgestelde klassendiagram. Op basis van dit ontwerp, kan de implementatie van de database beginnen. Naast de implementatie zijn een aantal testsets opgesteld, die na uitvoering moeten resulteren in een correcte implementatie van de database. Mede op basis van de resultaten van deze tests wordt een beslissing genomen over de verdere ontwikkeling.

De indeling van dit document is als volgt. In hoofdstuk 2 wordt het pilotontwikkelplan beschreven. In het daaropvolgende hoofdstuk worden de bouweenheden ontworpen. De invoeringsprocedure van de pilot wordt in hoofdstuk 4 toegelicht. Tot slot wordt de pilot beoordeeld op inhoud en gewenst resultaat in hoofdstuk 5.

2. Pilotontwikkelplan

Bij de ontwikkeling van deze pilot is direct gestart met het opstellen van het pilotplan. De implementatie van deze pilot mag niet teveel tijd in beslag nemen en de functionele en technische structuur van de pilot voegen relevante informatie toe.

2.1. Pilotdelen en bouweenheden

Om de databasepilot gestructureerd te kunnen ontwikkelen is deze onderverdeeld in verschillende pilotdelen. De pilotdelen zijn vooral bedoeld om de pilot logisch te ontwikkelen en in te voeren. De volgende pilotdelen zijn opgezet.

1. Opzetten databaseomgeving
2. Ontwerp database
3. Ontwikkelen klassen naar modellen

De gedefinieerde pilotdelen konden worden beschouwd als opzichzelfstaande bouweenheden. De pilotdelen zijn daarom niet verder onderverdeeld en hieronder beschreven als bouweenheden

2.1.1. Opzetten databaseomgeving

Het opzetten van een databaseomgeving bestaat uit een database toevoegen aan het DBMS. Het DBMS is in het tijdsbestek van de vorige pilot al geïnstalleerd. Naast het aanmaken van de database, zal er een nieuwe gebruiker met bijbehorend wachtwoord aangemaakt.

2.1.2. Ontwerpen database

Voordat het klassendiagram kan worden omgezet naar een implementatiemodel, moeten allereerst de datatypen worden vastgelegd. Deze bouweenheid ontwikkeld en implementeert de datatypen die voor het implementeren van de database nodig zijn.

2.1.3. Vertalen van klassen naar tabellen

Het klassendiagram ka na het vastleggen van de datatypen worden vertaald naar een implementatiemodel. In deze bouweenheid zullen naast de tabellen ook de constraints en vreemde sleutels zijn opgenomen.

2.2. Ontwikkelaanpak

De ontwikkeling van de pilot zal sequentieel worden aangepakt. Door de geringe omvang van de bouweenheden zullen deze niet na de implementatie worden getest. Na invoering van alle bouweenheden, zullen enkele tests worden uitgevoerd om te controleren of de database het gewenste resultaat heeft geleverd.

3. Ontwerp software-bouweenheden

De bouweenheden gedefinieerd in het vorige hoofdstuk zullen hier op inhoud worden uitgewerkt. De eerste paragraaf zal ingaan op de inhoud per bouweenheid en paragraaf 5.2 zal de testspecificaties in kaart brengen.

3.1. Bouweenheden

In de aankomende subparagrafen worden alle bouweenheden op inhoud gedefinieerd.

3.1.1. Opzetten databaseomgeving

Allereerst moet de database aangemaakt worden, die de toekomstige applicatie zal gaan gebruiken. Voor de naam van de database is gekozen voor dezelfde naam als die van de applicatie, namelijk 'wise_delivery'.
Het volgende SQL-statement zal er voor zorgen dat deze database wordt aangemaakt:

SQL-Statement voor het aanmaken van de database
CREATE DATABASE `wise_delivery'

Naast het aanmaken van deze nieuwe database, zal er ook een aparte gebruiker worden aangemaakt die deze database zal kunnen benaderen. In MySQL kun je bij het aanmaken van een gebruiker onder andere de velden aangeven die in figuur 1 te zien zijn.

←T→	Host	User	Password	Select_priv	Insert_priv	Update_priv	Delete_priv	Create_priv	Drop_priv	Reload_priv	Shutdown_priv
☐ ✎ 🗑	localhost	root		Y	Y	Y	Y	Y	Y	Y	Y

Figuur 1

In figuur 1 is te zien dat de *Host* van de gebruiker *root* 'localhost' is, oftewel 127.0.0.1. De gebruiker *root* mag alleen verbinding maken met de database vanaf het werkstation waarop deze geïnstalleerd is.

Om de rechten van bepaalde gebruikers te scheiden, zullen er twee gebruikers aangemaakt worden. De zogenaamde *root* gebruiker; deze gebruiker krijgt alle rechten en zal niet gebruikt worden in de applicatie. De *Info* gebruiker daarentegen zal de SQL statements gaan uitvoeren binnen de applicatie. Voor veiligheidsredenen zal deze gebruiker minimale rechten toegewezen krijgen. Deze rechten zullen zijn:

- Select privilege
- Insert privilege
- Update privilege
- Delete privilege

Deze vier privileges zullen voldoende zijn om de communicatie tussen de database en de applicatie goed te kunnen laten verlopen. De twee databasegebruikers zullen ieder een wachtwoord toegewezen krijgen. In de onderstaande tabel zijn deze als platte tekst opgenomen, dit zal in de database met md5 worden beveiligd.

Host	User	Password	Privileges
Localhost	Root	rT43qaCv	All
Localhost	Info	Tt5jal21a	Select, Insert, Update, Delete

De bovenstaande gegevens zullen met behulp van de onderstaande SQL-statements worden gerealiseerd in de user-tabel van de mysql-database.

User	SQL_statement
Root	`INSERT INTO `user` ( `Host` , `User` , `Password` , `Select_priv` , `Insert_priv` , `Update_priv` , `Delete_priv` , `Create_priv` , `Drop_priv` , `Reload_priv` , `Shutdown_priv` , `Process_priv` , `File_priv` , `Grant_priv` , `References_priv` , `Index_priv` , `Alter_priv` , `Show_db_priv` , `Super_priv` , `Create_tmp_table_priv` , `Lock_tables_priv` , `Execute_priv` , `Repl_slave_priv` , `Repl_client_priv` , `ssl_type` , `ssl_cipher` , `x509_issuer` , `x509_subject` , `max_questions` , `max_updates` , `max_connections` ) VALUES ( 'localhost', 'root', PASSWORD( 'rT43qaCv' ) , 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', '', '', '', '', '0', '0', '0');`
Info	`INSERT INTO `user` ( `Host` , `User` , `Password` , `Select_priv` , `Insert_priv` , `Update_priv` , `Delete_priv` , `Create_priv` , `Drop_priv` , `Reload_priv` , `Shutdown_priv` , `Process_priv` , `File_priv` , `Grant_priv` , `References_priv` , `Index_priv` , `Alter_priv` , `Show_db_priv` , `Super_priv` , `Create_tmp_table_priv` , `Lock_tables_priv` , `Execute_priv` , `Repl_slave_priv` , `Repl_client_priv` , `ssl_type` , `ssl_cipher` , `x509_issuer` , `x509_subject` , `max_questions` , `max_updates` , `max_connections` ) VALUES ( 'localhost', 'Info', PASSWORD( 'Tt5jal21a' ) , 'Y', 'Y', 'Y', 'Y', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', '', '', '', '', '0', '0', '0');`

3.1.2. Ontwerp database

Voordat de database kan worden ontworpen, zullen allereerst enkele afspraken moeten worden vastgelegd, zodat er bij het programmeren geen problemen kunnen ontstaan omtrent de communicatie met de database.

Alle velden van de tabellen zullen een bepaald type vereisen. Met deze typen wordt bijvoorbeeld ‘varchar’ of ‘integer’ bedoeld. Deze standaarden zijn in de onderstaande tabel beschreven.

Veldtype	Gewenst type	Eventuele grootte	MySQL-type
Primaire sleutels	Integer	-	INT, not null
Datum	Char	10	Char(10)
Tijd	Char	5	Char(5)
Telefoon- of faxnummers	Char	11	Char(11)
Huisnummer	Variabele char	50	Varchar(50)
Commentaar	Text		TEXT
Namen en gerelateerd	Variabele char	150	Varchar(150)

MySQL ondersteund voor datum en tijd een eigen type, echter is MySQL zo ingesteld dat deze waarde alleen een Amerikaanse notatie mag bevatten. Om de database zo onderhoudbaar mogelijk te houden, is besloten om alle data in Europese notatie in de database te bewaren. Dit houdt in dat hiervoor het type ‘Char’ is gekozen.

Voor de meeste velden in de tabel is een standaard type gekozen, namelijk ‘Varchar(150)’. Het type ‘varchar’ heeft als voordeel dat de niet opgevulde ruimte in de velden ook daadwerkelijk geen ruimte op het systeem in gebruik zal nemen, dit in tegenstelling tot het type ‘char’. Aangezien het in deze medische tak verplicht is om alle gegevens minimaal tien jaar te bewaren, zal de database veel data gaan bevatten.

Naast het vastleggen van een aantal veldtypen, is er besloten om iedere tabel een uniek nummer toe te kennen van het type *int(11)*.

Voor het databasetype is gekozen voor InnoDB, dit type heeft als eigenschap dat het gebruik van vreemde sleutels toestaat. De overige typen in MySQL bezitten deze eigenschap niet.

3.1.3. Vertalen van klassen naar tabellen

Iedere klasse in het klassendiagram zullen in dit pilotdeel worden vertaald naar een tabel, die in het relationeel model zal worden opgenomen. De uiteindelijke tabellen zijn hieronder in SQL-notatie opgenomen

SQL-Notatie

```
CREATE TABLE `ziekenhuis` (
  `nummer` int(11) NOT NULL auto_increment,
  `naam` varchar(150) default NULL,
  `straatnaam` varchar(150) default NULL,
  `huisnummer` varchar(50) default NULL,
  `postcode` varchar(6) default NULL,
  `woonplaats` varchar(150) default NULL,
  `telefoonnummer` varchar(11) default NULL,
  `faxnummer` varchar(11) default NULL,
  PRIMARY KEY  (`nummer`)
) TYPE=InnoDB AUTO_INCREMENT=1 ;
```

```
CREATE TABLE `medici` (
  `nummer` int(11) NOT NULL auto_increment,
  `voornaam` varchar(150) default NULL,
  `tussenvoegsel` varchar(50) default NULL,
  `achternaam` varchar(150) default NULL,
  `titel` varchar(150) default NULL,
  `specialisme` varchar(150) default NULL,
  `straatnaam` varchar(150) default NULL,
  `huisnummer` varchar(50) default NULL,
  `postcode` varchar(6) default NULL,
  `woonplaats` varchar(150) default NULL,
  `telefoonnummer` varchar(11) default NULL,
  `actief` char(1) NOT NULL default 'Y',
  PRIMARY KEY  (`nummer`)
) TYPE=InnoDB AUTO_INCREMENT=1 ;
```

```
CREATE TABLE `client` (
  `nummer` int(11) NOT NULL auto_increment,
  `voornaam` varchar(150) default NULL,
  `tussenvoegsel` varchar(150) default NULL,
  `meisjesnaam` varchar(150) default NULL,
  `geboortedatum` varchar(10) default NULL,
  `burgerlijke_staat` varchar(150) default NULL,
  `straatnaam` varchar(150) default NULL,
  `huisnummer` varchar(50) default NULL,
  `postcode` varchar(6) default NULL,
  `woonplaats` varchar(150) default NULL,
  `telefoonnummer` varchar(11) default NULL,
  `huisarts_nummer` int(11) NOT NULL default '0',
  `ziekenhuis_nummer` int(11) NOT NULL default '0',
  `opleiding` varchar(150) default NULL,
  `beroep` varchar(150) default NULL,
  `werkverband` varchar(150) default NULL,
  PRIMARY KEY  (`nummer`),
  INDEX (huisarts_nummer),
  FOREIGN KEY (huisarts_nummer) REFERENCES medici(nummer) ON UPDATE CASCADE on DELETE
RESTRICT,
  INDEX (ziekenhuis_nummer),
  FOREIGN KEY (ziekenhuis_nummer) REFERENCES ziekenhuis(nummer) ON UPDATE CASCADE on
DELETE RESTRICT

) TYPE=InnoDB AUTO_INCREMENT=1 ;
```

```
CREATE TABLE `praktijk` (
  `nummer` int(11) NOT NULL auto_increment,
  `naam` varchar(150) default NULL,
  `straatnaam` varchar(150) default NULL,
  `huisnummer` varchar(50) default NULL,
  `postcode` varchar(6) default NULL,
  `woonplaats` varchar(150) default NULL,
  `telefoonnummer` varchar(11) default NULL,
  `faxnummer` varchar(11) default NULL,
  PRIMARY KEY  (`nummer`)
) TYPE=InnoDB AUTO_INCREMENT=1 ;
```

```
CREATE TABLE `werknemer` (
  `nummer` int(11) NOT NULL default '0',
  `voornaam` varchar(150) default NULL,
  `tussenvoegsel` varchar(150) default NULL,
  `achternaam` varchar(150) default NULL,
  `straatnaam` varchar(150) default NULL,
  `huisnummer` varchar(50) default NULL,
  `postcode` varchar(6) default NULL,
  `woonplaats` varchar(150) default NULL,
  `telefoonnummer` varchar(11) default NULL,
  `geboortedatum` varchar(10) default NULL,
  `wachtwoord` varchar(50) default NULL,
  `praktijk_nummer` int(11) NOT NULL,
  PRIMARY KEY  (`nummer`),
  INDEX (praktijk_nummer),
  FOREIGN KEY(praktijk_nummer) REFERENCES praktijk(nummer) ON UPDATE CASCADE ON DELETE
RESTRICT
) TYPE=InnoDB;
```

```
CREATE TABLE `algemeen` (
  `nummer` int(11) NOT NULL auto_increment,
  `cystitis` varchar(150) default NULL,
  `herpes_labialis` varchar(150) default NULL,
  `foliumzuur` varchar(150) default NULL,
  `operatie` varchar(150) default NULL,
  `bloedtransfusie` varchar(150) default NULL,
  `thrombose` varchar(150) default NULL,
  `psychiatrische_aandoening` varchar(150) default NULL,
  `SOA` varchar(150) default NULL,
  `vaginisme` varchar(150) default NULL,
  `candida` varchar(150) default NULL,
  `specialist` varchar(150) default NULL,
  `alcohol` varchar(150) default NULL,
  `drugs` varchar(150) default NULL,
  `roken` varchar(150) default NULL,
  `allergie` varchar(150) default NULL,
  `medicatie` varchar(150) default NULL,
  `bijzonderheden` text,
  `client_nummer` int(11) NOT NULL default '0',
  PRIMARY KEY  (`nummer`),
  INDEX (client_nummer),
  FOREIGN KEY (client_nummer) REFERENCES client(nummer) ON UPDATE CASCADE on DELETE
RESTRICT
) TYPE=InnoDB AUTO_INCREMENT=1 ;
```

```
CREATE TABLE `contactmoment` (
  `nummer` int(11) NOT NULL auto_increment,
  `datum` varchar(10) default NULL,
  `herkomst` varchar(150) default NULL,
  `stadium` varchar(150) default NULL,
  `zorgvraag` varchar(150) default NULL,
  `overname_datum` varchar(10) default NULL,
  `verwijzingsreden` varchar(150) default NULL,
  `overnamereden` varchar(150) default NULL,
  `werknemer_nummer` int(11) NOT NULL default '0',
  `client_nummer` int(11) NOT NULL default '0',
  PRIMARY KEY  (`nummer`),
  INDEX (client_nummer),
  FOREIGN KEY (client_nummer) REFERENCES client(nummer) ON UPDATE CASCADE on DELETE
RESTRICT,
  INDEX (werknemer_nummer),
  FOREIGN KEY (werknemer_nummer) REFERENCES werknemer(nummer) ON UPDATE CASCADE on
DELETE RESTRICT
) TYPE=InnoDB AUTO_INCREMENT=1 ;
```

```
CREATE TABLE `zwangerschap` (
  `nummer` int(11) NOT NULL auto_increment,
  `code` varchar(6) NOT NULL default '',
  `datum_begin` varchar(10) default NULL,
  `plaats_bevalling` varchar(150) default NULL,
  `kraamzorg` varchar(150) default NULL,
  `voeding` varchar(150) default NULL,
  `voorbereiding` varchar(150) default NULL,
  `aanvullend_commentaar` text,
  `client_nummer` int(11) NOT NULL default '0',
  PRIMARY KEY  (`nummer`),
  KEY `code` (`code`),
  INDEX (client_nummer),
  FOREIGN KEY(client_nummer) REFERENCES client(nummer) ON UPDATE CASCADE ON DELETE
RESTRICT
) TYPE=InnoDB AUTO_INCREMENT=1 ;
```

```
CREATE TABLE `echo` (
  `nummer` int(11) NOT NULL auto_increment,
  `datum` varchar(10) default NULL,
  `plaats` varchar(150) default NULL,
  `extra_informatie` text,
  `zwangerschaps_nummer` int(11) NOT NULL default '0',
  PRIMARY KEY  (`nummer`),
  INDEX (zwangerschaps_nummer),
  FOREIGN KEY (zwangerschaps_nummer) REFERENCES zwangerschap(nummer) ON UPDATE CASCADE
ON DELETE CASCADE
) TYPE=InnoDB AUTO_INCREMENT=1 ;
```

```
CREATE TABLE `familie` (
  `nummer` int(11) NOT NULL auto_increment,
  `diabetes` varchar(150) default NULL,
  `hypertensie` varchar(150) default NULL,
  `congenitale_afwijkingen` varchar(150) default NULL,
  `heupdysplasie` varchar(150) default NULL,
  `bijzonderheden` text,
  `client_nummer` int(11) NOT NULL default '0',
  PRIMARY KEY  (`nummer`),
  INDEX (client_nummer),
  FOREIGN KEY (client_nummer) REFERENCES client(nummer) ON UPDATE CASCADE ON DELETE
CASCADE
) TYPE=InnoDB AUTO_INCREMENT=1 ;
```

```
CREATE TABLE `partner` (
  `nummer` int(11) NOT NULL auto_increment,
  `voornaam` varchar(150) default NULL,
  `tussenvoegsel` varchar(50) default NULL,
  `achternaam` varchar(150) default NULL,
  `opleiding` varchar(150) default NULL,
  `beroep` varchar(150) default NULL,
  `werkverband` varchar(150) default NULL,
  `client_nummer` int(11) NOT NULL default '0',
  PRIMARY KEY  (`nummer`),
  INDEX (client_nummer),
  FOREIGN KEY (client_nummer) REFERENCES client(nummer) ON UPDATE CASCADE ON DELETE
RESTRICT
) TYPE=InnoDB AUTO_INCREMENT=1 ;
```

```
CREATE TABLE `familie_partner` (
  `nummer` int(11) NOT NULL auto_increment,
  `congenitale_afwijkingen` varchar(150) default NULL,
  `heupdysplasie` varchar(150) default NULL,
  `bijzonderheden` text,
  `partner_nummer` int(11) NOT NULL default '0',
  PRIMARY KEY  (`nummer`),
  INDEX (partner_nummer),
  FOREIGN KEY (partner_nummer) REFERENCES partner(nummer) ON UPDATE CASCADE ON DELETE
CASCADE
) TYPE=InnoDB AUTO_INCREMENT=1 ;
```

```
CREATE TABLE `verzekeraar` (
  `nummer` int(11) NOT NULL auto_increment,
  `naam` varchar(150) default NULL,
  `straatnaam` varchar(150) default NULL,
  `huisnummer` varchar(50) default NULL,
  `postcode` varchar(6) default NULL,
  `plaats` varchar(150) default NULL,
  `telefoonnummer` varchar(11) default NULL,
  `faxnummer` varchar(11) default NULL,
  PRIMARY KEY  (`nummer`)
) TYPE=InnoDB AUTO_INCREMENT=1 ;
```

```
CREATE TABLE `is_verzekerd` (
  `client_nummer` int(11) NOT NULL default '0',
  `verzekeraar_nummer` int(11) NOT NULL default '0',
  `datum_aanvullend` varchar(11) default NULL,
  PRIMARY KEY  (`client_nummer`,`verzekeraar_nummer`),
  INDEX ( verzekeraar_nummer ),
  FOREIGN KEY (verzekeraar_nummer) REFERENCES verzekeraar(nummer) ON UPDATE CASCADE ON
DELETE RESTRICT,
  INDEX ( client_nummer ),
  FOREIGN KEY (client_nummer) REFERENCES client(nummer) ON UPDATE CASCADE ON DELETE
CASCADE
) TYPE=InnoDB;
```

```
CREATE TABLE `partus` (
  `nummer` int(11) NOT NULL auto_increment,
  `code` varchar(50) NOT NULL default '',
  `leiding` int(11) NOT NULL default '0',
  `plaats` varchar(150) default NULL,
  `datum` varchar(11) default NULL,
  `aantal_weken` int(11) default NULL,
  `begin_stadium_a` varchar(150) default NULL,
  `begin_stadium_b` varchar(150) default NULL,
  `ontsluiting` varchar(150) default NULL,
  `pijnbestrijding_baring` varchar(150) default NULL,
  `stimulans` varchar(150) default NULL,
  `vliesbreuk` varchar(150) default NULL,
  `kleur` varchar(150) default NULL,
  `problematiek_ontsluiting` text,
  `ligging` varchar(150) default NULL,
  `hulp` varchar(150) default NULL,
  `indicatie_uitdrijving` varchar(150) default NULL,
  `schouderdystocie` varchar(150) default NULL,
  `entourage` varchar(150) default NULL,
  `pijnbestrijding_uitdrijving` varchar(150) default NULL,
  `problematiek_uitdrijving` varchar(150) default NULL,
  `leidingswijze` varchar(150) default NULL,
  `geboortewijze` varchar(150) default NULL,
  `bloedverlies` varchar(150) default NULL,
  `medicatie` varchar(150) default NULL,
  `ziektebeeld_nageboorte` varchar(150) default NULL,
  `compleet` varchar(150) default NULL,
  `eindbestemming` varchar(150) default NULL,
  `aantal_vaten` float default NULL,
  `problematiek_nageboorte` text,
  `datum_vliezen` varchar(11) default NULL,
  `tijd_vliezen` varchar(5) default NULL,
  `tijd_meepersen` varchar(5) default NULL,
  `datum_geboorte` varchar(11) default NULL,
  `tijd_geboorte` varchar(5) default NULL,
  `tijd_placenta` varchar(5) default NULL,
  `ruptuur` varchar(150) default NULL,
  `episiotomie` varchar(150) default NULL,
  `type` varchar(150) default NULL,
  `indicatie_trauma` varchar(150) default NULL,
  `ziektebeeld_trauma` varchar(150) default NULL,
  `overig_trauma` varchar(150) default NULL,
  `gehecht_door` varchar(150) default NULL,
  `problematiek_trauma` text,
  `aanvullend_commentaar` text,
  PRIMARY KEY  (`nummer`),
  KEY `code` (`code`),
  INDEX (leiding),
  FOREIGN KEY(leiding) REFERENCES werknemer(nummer) ON UPDATE CASCADE ON DELETE
RESTRICT
) TYPE=InnoDB AUTO_INCREMENT=1 ;
```

```
CREATE TABLE `kind` (
  `nummer` int(11) NOT NULL auto_increment,
  `voornaam` varchar(150) default NULL,
  `tussenvoegsel` varchar(50) default NULL,
  `achternaam` varchar(150) default NULL,
  `geslacht` enum('M','V','O') default NULL,
  `gewicht` float default NULL,
  `gewogen_apparaat` varchar(150) default NULL,
  `aspect` varchar(150) default NULL,
  `proporties` varchar(150) default NULL,
  `opname` varchar(150) default NULL,
  `apgar_1` float default NULL,
  `apgar_5` float default NULL,
  `apgar_10` float default NULL,
  `conditie` varchar(150) default NULL,
  `reanimatie` varchar(150) default NULL,
  `testis_rechts` varchar(150) default NULL,
  `testis_links` varchar(150) default NULL,
  `hoort_bij` enum('P','K') NOT NULL default 'P',
  `partus_nummer` int(11) NOT NULL default '0',
  PRIMARY KEY  (`nummer`),
  INDEX (partus_nummer),
  FOREIGN KEY(partus_nummer) REFERENCES partus(nummer) ON UPDATE CASCADE ON DELETE
RESTRICT
) TYPE=InnoDB AUTO_INCREMENT=1 ;
```

```
CREATE TABLE `kraambed` (
  `nummer` int(11) NOT NULL auto_increment,
  `voeding` varchar(150) default NULL,
  `opname` varchar(150) default NULL,
  `conditie` varchar(150) default NULL,
  `ontslag` varchar(150) default NULL,
  `lactatieremming` varchar(150) default NULL,
  `kraamzorg` varchar(150) default NULL,
  `anti_d` enum('Ja','Nee','?') NOT NULL default 'Nee',
  `bijzonderheden` text,
  `aanvullend_commentaar` text,
  `kind_nummer` int(11) NOT NULL default '0',
  `partus_nummer` int(11) NOT NULL default '0',
  PRIMARY KEY  (`nummer`),
  INDEX (partus_nummer),
  FOREIGN KEY(partus_nummer) REFERENCES partus(nummer) ON UPDATE CASCADE ON DELETE
RESTRICT,
  INDEX (kind_nummer),
  FOREIGN KEY(kind_nummer) REFERENCES kind(nummer) ON UPDATE CASCADE ON DELETE
RESTRICT
) TYPE=InnoDB AUTO_INCREMENT=1 ;
```

```
CREATE TABLE `labuitslag` (
  `nummer` int(11) NOT NULL default '0',
  `glucose_1` varchar(150) default NULL,
  `glucose_2` varchar(150) default NULL,
  `glucose_3` varchar(150) default NULL,
  `bloedgroep` varchar(150) default NULL,
  `Hemoglobine` varchar(150) default NULL,
  `MCV` varchar(150) default NULL,
  `antistoffen` varchar(150) default NULL,
  `Hepatitis_b` varchar(150) default NULL,
  `Rubella` varchar(150) default NULL,
  `HIV` varchar(150) default NULL,
  `Rhesusfactor` varchar(150) default NULL,
  `Rhesusfactor_antid` varchar(150) default NULL,
  `MCV_28` varchar(150) default NULL,
  `Hemoglobine_28` varchar(150) default NULL,
  `zwangerschaps_nummer` int(11) NOT NULL,
  PRIMARY KEY (`nummer`),
  INDEX (zwangerschaps_nummer),
  FOREIGN KEY (zwangerschaps_nummer) REFERENCES zwangerschap(nummer) ON UPDATE CASCADE
ON DELETE CASCADE
) TYPE=InnoDB;
```

```
CREATE TABLE `los_kraambed` (
  `nummer` int(11) NOT NULL auto_increment,
  `a_terme_datum` varchar(10) default NULL,
  `zekerheid` varchar(150) default NULL,
  `gravida` int(11) default NULL,
  `para` int(11) default NULL,
  `spontane_abortus` varchar(150) default NULL,
  `abortus_provocatus` varchar(150) default NULL,
  `EUG` varchar(150) default NULL,
  `foetal_loss` varchar(150) default NULL,
  `in_leven` varchar(150) default NULL,
  `bloedgroep` varchar(150) default NULL,
  `rhesusfactor` varchar(150) default NULL,
  `partus_datum` varchar(10) default NULL,
  `partus_tijd` varchar(5) default NULL,
  `leiding` varchar(150) default NULL,
  `plaats` varchar(150) default NULL,
  `voorgenomen_plaats` varchar(150) default NULL,
  `client_nummer` int(11) NOT NULL default '0',
  PRIMARY KEY  (`nummer`),
  INDEX (client_nummer),
  FOREIGN KEY (client_nummer) REFERENCES client(nummer) ON UPDATE CASCADE ON DELETE
RESTRICT
) TYPE=InnoDB AUTO_INCREMENT=1 ;
```

```
CREATE TABLE `obstetrisch` (
  `nummer` int(11) NOT NULL default '0',
  `datum` varchar(10) default NULL,
  `uitkomst` varchar(150) default NULL,
  `tijd_weken` int(11) default NULL,
  `tijd_dagen` int(11) default NULL,
  `ziekenhuis_nummer` int(11) NOT NULL default '0',
  `anti_d` int(11) default NULL,
  `toelichting` text,
  `client_nummer` int(11) NOT NULL default '0',
  PRIMARY KEY (`nummer`),
  INDEX (client_nummer),
  FOREIGN KEY (client_nummer) REFERENCES client(nummer) ON UPDATE CASCADE ON DELETE
RESTRICT,
  INDEX (ziekenhuis_nummer),
  FOREIGN KEY (ziekenhuis_nummer) REFERENCES ziekenhuis(nummer) ON UPDATE CASCADE ON
DELETE RESTRICT
) TYPE=InnoDB;
```

```
CREATE TABLE `obs_abortus` (
  `nummer` int(11) NOT NULL auto_increment,
  `ingreep` varchar(150) default NULL,
  `bijzonderheden` text,
  `obstetrisch_nummer` int(11) NOT NULL default '0',
  PRIMARY KEY  (`nummer`),
  INDEX (obstetrisch_nummer),
  FOREIGN KEY (obstetrisch_nummer) REFERENCES obstetrisch(nummer) ON UPDATE CASCADE ON
DELETE CASCADE
) TYPE=InnoDB AUTO_INCREMENT=1 ;
```

```
CREATE TABLE `obs_zwangerschap` (
  `nummer` int(11) NOT NULL auto_increment,
  `geslacht` enum('M','V') NOT NULL default 'M',
  `ligging` varchar(150) default NULL,
  `hulp` varchar(150) default NULL,
  `perineum` varchar(150) default NULL,
  `bloedverlies` varchar(150) default NULL,
  `conditie` varchar(150) default NULL,
  `apgar_1` int(11) default NULL,
  `apgar_5` int(11) default NULL,
  `kind_voornaam` varchar(150) default NULL,
  `kind_tussenvoegsel` varchar(50) default NULL,
  `kind_achternaam` varchar(150) default NULL,
  `geleid_door` varchar(150) default NULL,
  `praktijk_nummer` int(11) NOT NULL default '0',
  `verantwoordelijke` varchar(150) default NULL,
  `bijzonderheden` text,
  `toelichting` text,
  `obstetrisch_nummer` int(11) NOT NULL default '0',
  PRIMARY KEY  (`nummer`),
  INDEX (obstetrisch_nummer),
  FOREIGN KEY (obstetrisch_nummer) REFERENCES obstetrisch(nummer) ON UPDATE CASCADE ON
DELETE CASCADE,
  INDEX (praktijk_nummer),
  FOREIGN KEY (praktijk_nummer) REFERENCES praktijk(nummer) ON UPDATE CASCADE ON
DELETE RESTRICT
) TYPE=InnoDB AUTO_INCREMENT=1 ;
```

```
CREATE TABLE `onderzoek` (
  `nummer` int(11) NOT NULL auto_increment,
  `lichaamslengte` float default NULL,
  `gewicht_voor_zwangerschap` float default NULL,
  `gewicht_bij_onderzoek` float default NULL,
  `prenatala_diagnostiek` int(11) NOT NULL default '0',
  `bijzonderheden` text,
  `client_nummer` int(11) NOT NULL default '0',
  PRIMARY KEY  (`nummer`),
  INDEX (client_nummer),
  FOREIGN KEY (client_nummer) REFERENCES client(nummer) ON UPDATE CASCADE ON DELETE
RESTRICT
) TYPE=InnoDB AUTO_INCREMENT=1 ;
```

```
CREATE TABLE `termijn` (
  `nummer` int(11) NOT NULL default '0',
  `datum_laatste_menstruatie` varchar(11) default NULL,
  `menstruatie_is_onbekend` enum('0','1') NOT NULL default '0',
  `cyclus_in_dagen` int(11) default NULL,
  `zekerheid` varchar(150) default NULL,
  `fertifiteit` varchar(150) default NULL,
  `anticonceptie` varchar(150) default NULL,
  `a_terme_datum` varchar(11) default NULL,
  `a_terme_termijn` varchar(150) NOT NULL default '',
  `a_terme_zekerheid` varchar(150) default NULL,
  `zwangerschaps_nummer` int(11) NOT NULL default '0',
  PRIMARY KEY(`nummer`),
  INDEX(`zwangerschaps_nummer`),
  FOREIGN KEY (zwangerschaps_nummer) REFERENCES zwangerschap(nummer) ON UPDATE CASCADE
ON DELETE RESTRICT
) TYPE=InnoDB;
```

3.2. Testspecificatie

Na het ontwerp van de database, bestaat er de mogelijkheid dat er eventueel enkele fouten bij ingeslopen zijn, of dat de diagrammen niet meer consistent zijn. Om deze consistentie te waarborgen en de correctheid van de pilot te verzekeren, zullen er enkele tests uitgevoerd moeten worden.
Om de pilot volledig te testen, zijn de tests onderverdeeld in twee groepen, opdat ook de producten uit de voorgaande stukken getest kunnen worden:

- Voorgaande diagrammen
- Database

Testspecificatie Voorgaande diagrammen	
Case #	**Test**
1.0	Klassen in klassendiagrammen zijn gelijk aan de tabellen in de database
1.1	Klassenrelaties komen overeen met de database
1.2	Kardinalen in klassenrelaties zijn consistent met de plaatsing van de vreemde sleutels in de tabellen in de database

Testspecificatie Database	
Case #	**Test**
2.0	De vreemde sleutels in de tabellen hebben allemaal een index
2.1	De vreemde sleutels in de tabellen zijn allemaal gedefinieerd als ‘FOREIGN KEY’ met de juiste constraints.
2.2	Iedere tabel heeft hetzelfde type; ‘InnoDB’.
2.3	De constraints (ON Update en On Delete) geven het gewenste resultaat bij iedere tabel.
2.4	Er komen geen dubbele referenties voor in database, zoals werknemer -> contactmoment -> werknemer.
2.5	De gebruiker(s) voor MySQL kan inloggen met het opgegeven wachtwoord.
2.6	De gebruiker(s) voor MySQL heeft alleen beschikking over de toegewezen rechten.

4. Invoeringsprocedure pilot

Wanneer de pilot functioneert volgens de opgestelde eisen en alle testspecificaties zijn doorlopen, kan worden begonnen met het invoeren van de pilot. Dit hoofdstuk beschrijft de procedure(s) die gevolgd moeten worden om de pilot succesvol te kunnen invoeren.
In de eerstvolgende subparagraaf zal het invoeringsplan worden toegelicht. In de daaropvolgende paragraaf zullen de testspecificaties worden beschreven die betrekking hebben op het invoeringsplan.

4.1. Invoeringsplan

Het invoeren van de database zal stapsgewijs moeten gebeuren om het overzicht te kunnen bewaren. Alle tabellen in de database zijn van het type InnoDB en maken het gebruik van vreemde sleutels mogelijk. Dit heeft dus invloed op de invoering van de tabellen. Allereerst zal gekeken worden in welke volgorde de tabellen moeten worden. Dit is van belang aangezien de constraints die een tabel heeft met vreemde sleutels zou de invoering van een bepaalde tabel kunnen belemmeren.

4.1.1. De tabellen

Een voorbeeld hiervan, de tabel ‘client’ heeft een relatie met de tabel ‘medici’ en ‘ziekenhuis’, deze laatste twee tabellen zullen daarom ingevoerd moeten zijn, voordat de tabel client ingevoerd kan worden. Hieronder is een overzicht opgenomen met aan de linkerkant de tabellen die ingevoerd moeten worden en aan de rechterkant de tabellen die ingevoerd moeten zijn voordat deze kunnen worden ingevoerd.

Tabelnaam	Tabel(len) vereist
algemeen	client
client	medici ziekenhuis
contactmoment	client werknemer
echo	zwangerschap
familie	client
familie_partner	partner
is_verzekerd	client verzekeraar
kind	partus
kraambed	kind partus
labuitslag	zwangerschap
los_kraambed	client
medici	-
obs_abortus	obstetrisch
obs_zwangerschap	obstetrisch
obstetrisch	client ziekenhuis
onderzoek	client
partner	client
partus	werknemer

praktijk	-
termijn	zwangerschap
verzekeraar	-
werknemer	praktijk
ziekenhuis	-
zwangerschap	client

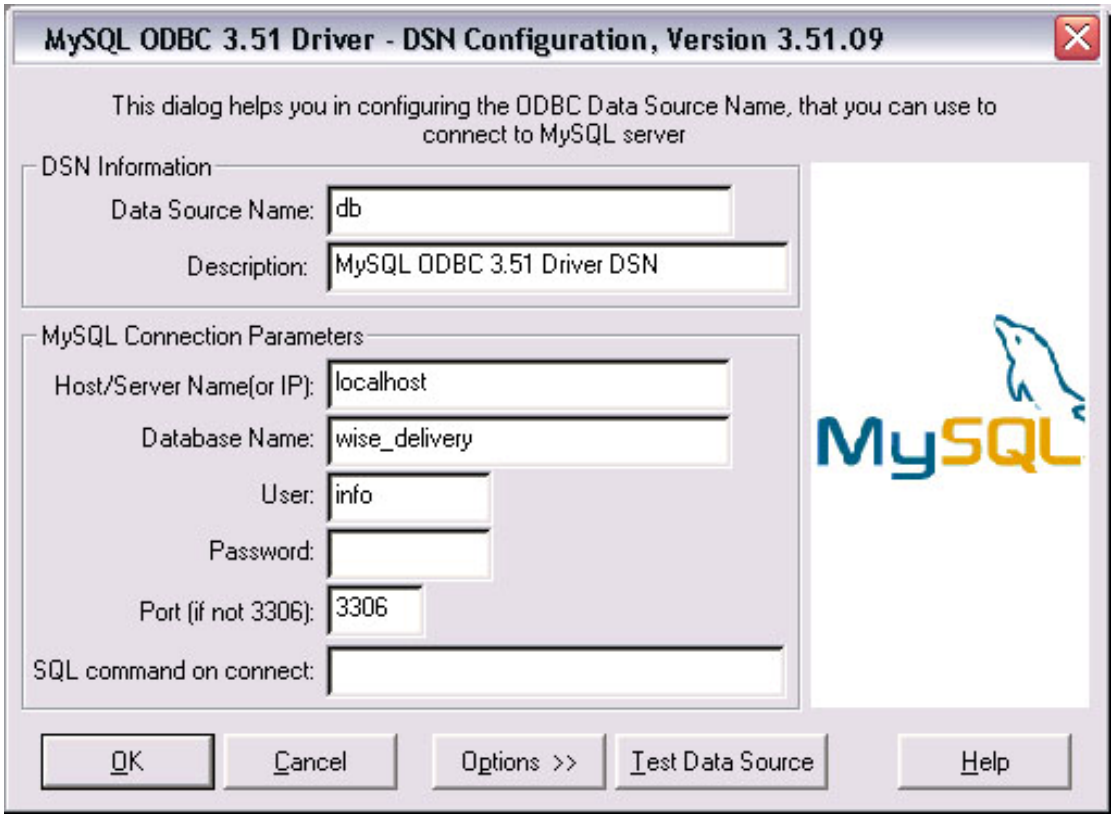
Uit bovenstaande lijst zal door middel van prioriteiten worden bepaald welke tabellen als eerste ingevoerd zullen moeten worden. Zoals te zien is in de bovenstaande lijst blijkt wel dat de tabellen medici, praktijk, verzekeraar en ziekenhuis geen referenties hebben, dus zullen die als eerste ingevoerd worden. Hieronder volgt de gehele lijst, met bijbehorende prioriteiten.
De toegekende prioriteiten zullen lopen van A – C, waarin A de tabellen zijn die geen afhankelijkheden hebben. Tabellen met prioriteit B zijn de tabellen die een referentie hebben met een andere tabel. De laatste prioriteit (C) zijn de overgebleven tabellen

Tabelnaam	Prioriteit
medici	A
praktijk	A
verzekeraar	A
ziekenhuis	A
client	B
werknemer	B
zwangerschap	B
obstetrisch	B
partus	B
kind	B
Partner	B
contact_moment	C
algemeen	C
Echo	C
Familie	C
Familie_partner	C
Is_verzekerd	C
Kraambed	C
Labuitslag	C
Los_kraambed	C
Obs_abortus	C
Obs_zwangerschap	C
Onderzoek	C
Termijn	C

4.1.2. MyODBC

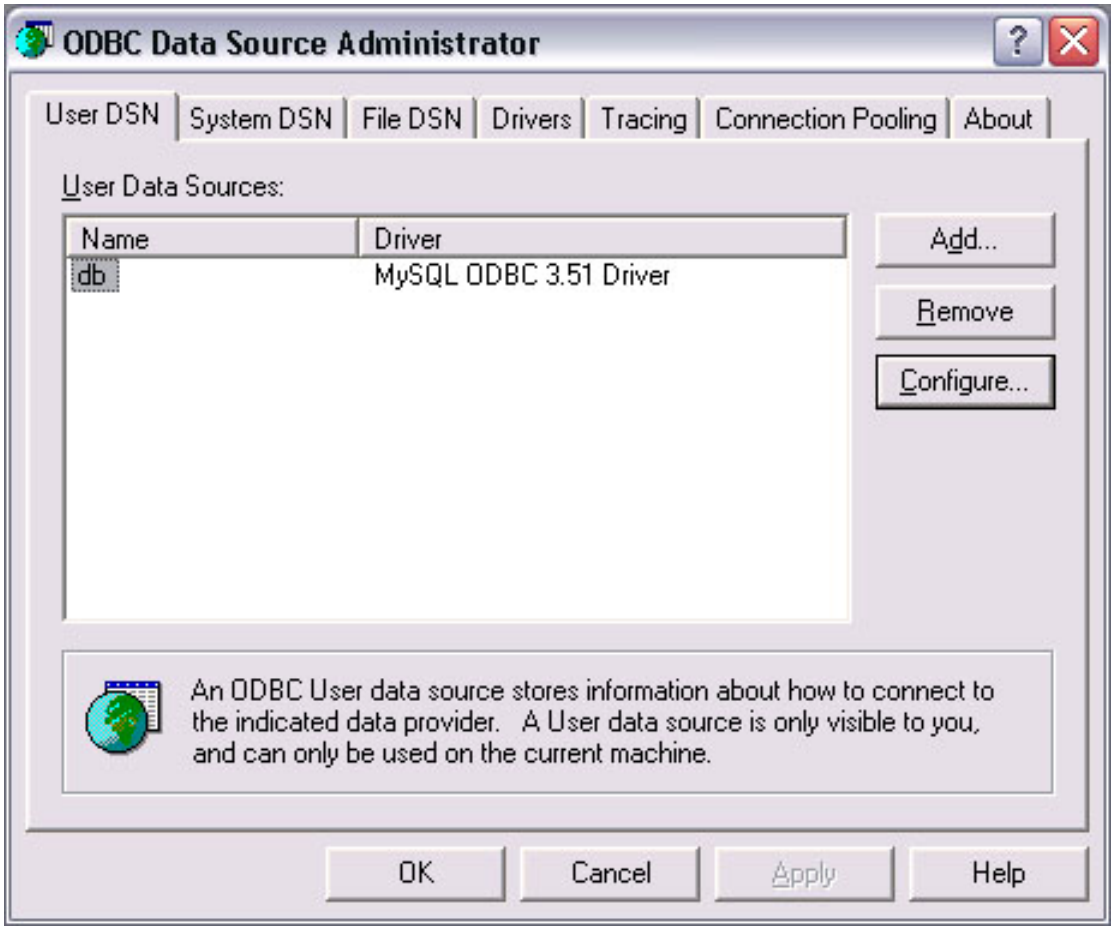
Naast het invoeren van de tabellen, is een communicatiemiddel tussen de applicatie en het DBMS nodig. Een stuk software dat een deel van de communicatie afvangt is **O**pen **D**ata**B**ase **C**onnectivity (ODBC). Dit stuk software is ook speciaal voor het DBMS MySQL ontwikkeld en heet MyODBC.

Na het installeren van MyODBC, zal deze de juiste gegevens moeten bevatten om de database ‘wise_delivery’ te kunnen benaderen. In het onderstaande figuur is te zien hoe deze instellingen er uit zullen moeten gaan zien, dit is alleen geldig voor het besturingssysteem XP.



Figuur 2

In Figuur 2 is te zien dat de in de applicatie ‘db’ aangeroepen zal moeten worden als MyODBC Resource.
De description is een standaardbeschrijving en heeft voor de applicatie geen doel.
Het DBMS zal in de toekomstige situatie gaan draaien op hetzelfde werkstation als waar de applicatie draait en kan dus worden aangeroepen met ‘localhost’ (127.0.0.1).
De naam van de database is wise_delivery en de gebruiker heet ‘Info’ en het bijbehorende wachtwoord is hier niet opgenomen, maar is in dit hoofdstuk al besproken.
De poort waarop de MySQL draait is 3306 en zal hier niet anders zijn.



Figuur 3

Figuur 3 laat zien dat het aanmaken van de connectie middels ODBC met het DBMS gelukt is.

4.2. Testspecificaties

Om te testen of de invoer van de database correct gebeurt is, zullen er een aantal tests uitgevoerd moeten worden. Deze tests zullen niet gericht zijn op de inhoud van de pilot, maar op de invoering ervan.

Testspecificatie Invoeringsplan	
Case #	**Test**
3.0	Er is een versie van MySQL aanwezig die het tabeltype ‘InnoDB’ ondersteund; 4.0.22.
3.1	De tabel ‘wise_delivery’ is aangemaakt en benaderbaar.
3.2	De laatste versie van MyODBC is geïnstalleerd.
3.3	De connectie met het DBMS lukt middels MyODBC.

5. Testen en beoordelen Pilot

Na de implementatie van iedere bouweenheid zijn deze getest aan de hand van een aantal testspecificaties uit hoofdstuk 4. Dit hoofdstuk beschrijft deze tests en diens resultaten en licht waar nodig de mislukte cases toe.

5.1. Testspecificatie voorgaande diagrammen

Case #	Test	Resultaat
1.0	Klassen in klassendiagrammen zijn gelijk aan de tabellen in de database	Mislukt
1.1	Klassenrelaties komen overeen met de database	Mislukt
1.2	Kardinalen in klassenrelaties zijn consistent met de plaatsing van de vreemde sleutels in de tabellen in de database	OK

Mislukte case #1.0
De klassen die in het klassendiagram waren aangebracht, kwamen niet overeen met de tabellen in de database. De reden hiervoor was, dat klassendiagram tijdens het ontwerp van de tweede pilot gewijzigd was. Dit was echter niet doorgevoerd in de ontwikkelingen van deze pilot.
De oplossing hiervoor was om te vergelijken waar de verschillen zaten en deze in de database aan te brengen.

Mislukte case #1.1
Deze mislukte testcase was een direct gevolg op de eerste. Doordat de klassen niet overeenkwamen met de tabellen, waren de relaties tussen de klassen niet overeenstemming met de database.
Na het aanpassen van het klassendiagram zijn de testcases #1.0 en #1.1 nogmaals uitgevoerd en ditmaal met succes.

5.2. Testspecificatie Database

Case #	Test	Resultaat
2.0	De vreemde sleutels in de tabellen hebben allemaal een index	OK
2.1	De vreemde sleutels in de tabellen zijn allemaal gedefinieerd als ‘FOREIGN KEY’ met de juiste constraints.	OK
2.2	Iedere tabel heeft hetzelfde type; ‘InnoDB’.	Ok
2.3	De constraints (ON Update en On Delete) geven het gewenste resultaat bij iedere tabel.	OK
2.4	Er komen geen dubbele referenties voor in database, zoals werknemer -> contactmoment -> werknemer.	OK
2.5	De gebruiker(s) voor MySQL kan inloggen met het opgegeven wachtwoord.	OK
2.6	De gebruiker(s) voor MySQL heeft alleen beschikking over de toegewezen rechten.	OK

5.3. Testspecificaties invoeringsplan

Case #	Test	Resultaat
3.0	Er is een versie van MySQL aanwezig die het tabeltype 'InnoDB' ondersteund; 4.0.22.	Ok
3.1	De tabel 'wise_delivery' is aangemaakt en benaderbaar.	Ok
3.2	De laatste versie van MyODBC is geïnstalleerd.	Ok
3.3	De connectie met het DBMS lukt middels MyODBC.	Ok

5.4. Beoordeling pilot

De beoordeling van de pilot kon pas worden uitgevoerd, nadat de verbeteringen waren aangebracht in de bouweenheden. Naast het mislukken van de eerste testcases, kon de pilot als succesvol worden beschouwd. Er was voldaan aan het gewenste resultaat dat in de definitiestudie was opgesteld voor deze pilot.
De database was opgesteld en ingevoerd aan de hand van het klassendiagram.

Conclusie

In dit document is een ontwerp gemaakt dat heeft gediend ter ondersteuning van de implementatie van de database. Na de bouw van iedere bouweenheid zijn de testcases uitgevoerd . De resultaten van de uitgevoerde testcases hebben uitgewezen dat in eerste instantie de pilot niet voldeed aan de eisen. De tabellen en de bijbehorende relaties in de database waren niet correct.
Door het uitvoeren van de verbeteringen, voldeed de pilot aan de gestelde eisen.

Bijlage A: Plan van aanpak pilot 2 'Database'

Inleiding

In dit document worden de onderdelen van de tweede pilot toegelicht en in een planning opgenomen. Dit plan van aanpak is een detailplanning voor de tweede pilot en toevoeging op het plan van aanpak voor het gehele project.
Allereerst zullen de resultaten die de pilot oplevert worden beschreven en vervolgens zal een planning gemaakt worden voor de uitvoering van de pilot.
Voor informatie over kwaliteitsborging en de projectorganisatie wordt verwezen naar het plan van aanpak voor het gehele project.

Activiteiten pilotontwikkeling

Hieronder volgt een overzicht van de activiteiten die worden uitgevoerd tijdens de ontwikkeling van de eerste pilot. Per activiteit worden deelactiviteiten beschreven en er wordt aangegeven wat het resultaat is van deze activiteit.

- Opstellen pilotontwikkelplan
    - Definiëren pilotdelen
    - Definiëren bouweenheden
    - Voorbereiden beoordeling en testen pilotdelen;
    - Resultaat: ontwikkelplan voor pilot.
- Ontwerpen software-bouweenheden
    - Specificeren integratie en testen bouweenheden
    - Opstellen detailspecificaties
    - Opstellen testspecificaties
    - Resultaat: gedetailleerd ontwerp pilotdelen.
- Bouwen software-bouweenheden
    - Coderen pilotdelen
    - Voorbereiden component-test
    - Uitvoeren component-test
    - Corrigeren component
    - Resultaat: implementatie van pilot.
- Opstellen invoeringsprocedure pilot
    - Ontwerpen invoeringsprocedure
    - Specificeren tests invoeringsprocedures
    - Testen invoeringsprocedures
    - Corrigeren invoeringsprocedures
    - Resultaat: beschrijving invoeringsprocedure.
- Beoordelen en testen pilot
    - Opzetten testomgeving
    - Beoordelen en testen pilot
    - Analyseren uitkomsten
    - Corrigeren fouten
    - Rapporteren over beoordelen en testen

Naast deze stappen zijn een aantal volgens IAD voorgeschreven ontwikkelstappen bewust overgeslagen. Deze stappen zullen geen toegevoegde waarde leveren op het ontwerp of de ontwikkeling van de pilot.

De overige activiteiten die IAD voorschrijft worden niet voor dit product uitgevoerd.

- Voorbereiden pilotontwerp-workshop
  Een workshop voor het gehele project is reeds gehouden in de definitiefase.
- Specificeren globaal-functionele structuur pilot
  Er hoeft geen functionele structuur van de database worden ontworpen.
- Specificeren globaal-technische structuur pilot
  De technische structuur van de pilot is onveranderd sinds de laatste pilot.
- Specificeren globaal-organisatorische inrichting
  De invloed op de organisatorische aspecten zijn beschreven in de definitiefase. Deze pilot levert geen extra veranderingen met zich mee..
- Aanpassen externe componenten
  Er hoeven geen externe componenten te worden aangepast.
- Wijzigen andere informatiesystemen
  Er is geen sprake van een ander informatiesysteem.
- Bouwen conversie-tools
  Deze pilot vereist geen conversie-tools
- Ontwerpen handmatige procedures
  De handmatige procedures zijn hier niet van toepassing.
- Integreren bouweenheden
  De bouweenheden worden bij implementatie direct geïntegreerd, doordat deze onderdeel zijn van de applicatie.
- Samenstellen opleidingsmateriaal pilot
  Er is geen behoefte aan opleidingsmateriaal voor enkel deze pilot.
- Samenstellen handleiding pilot
  Er behoeft geen handleiding te worden geschreven voor deze pilot

<u>Planning</u>
Hieronder volgt een planning van de uit te voeren activiteiten.

Product / Proces	Tijdsduur in uren
Opstellen pilotontwikkelplan	8
Ontwerpen software-bouweenheden	25
Bouwen software-bouweenheden	25
Opstellen invoeringsprocedure pilot	5
Beoordelen en testen pilot	2
Totaal aantal uren	65

Bijlage 5
Pilotontwikkelrapport
Pilot 3: GUI-Database Interface

Voorwoord

Voor u ligt het pilotontwikkelrapport dat ter ondersteuning dient bij de implementatie van de derde pilot uit het project W!se delivery. Tijdens deze pilot worden de functionaliteiten gedefinieerd, die in definitiefase als de derde pilot is benoemd.

In het systeemontwikkelingtraject van IAD volgt na de definitiefase, de fase van de pilotontwikkeling. Tijdens de pilotontwikkeling worden de pilots die in de definitiefase zijn gedefinieerd, ontwikkeld en geïmplementeerd.

Dit document is tot stand gekomen in overleg met Mevr. S. van Iterson, opdrachtgever bij de verloskundepraktijk de Maatschap.

Leiden,

Cees Kruijf
Verloskundepraktijk de Maatschap.

Inhoudsopgave

1. Inleiding

Tijdens de definitiestudie is het project verdeeld in vijf pilots. Deze pilots worden in het pilotontwikkelingstraject sequentieel ontwikkeld. De eerste pilot bestond uit het ontwerpen en ontwikkelen van de Grafische User Interface. De tweede pilot bestond uit het ontwikkelen van de databasestructuur. Dit document dient ter ondersteuning van de ontwikkeling van derde pilot “GUI-database Interface”.

Het doel van dit rapport is om de functionaliteiten van de GUI-Database interface zodanig te ontwerpen, dat op basis hiervan met de implementatie kan worden begonnen. Daarnaast worden voor iedere bouweenheid in dit document testen gespecificeerd. De uitvoering van de test maakt het mogelijk om de ontworpen bouweenheden te valideren en verifiëren op de inhoud. Mede op basis van de resultaten van deze tests wordt een beslissing genomen over de verdere ontwikkeling.

De opbouw van dit document is als volgt, in het volgende hoofdstuk wordt de functionele structuur ontworpen. In hoofdstuk 3 is de technische structuur toegelicht en wordt in het daaropvolgende hoofdstuk een pilotontwikkelplan opgesteld. De gedefinieerde bouweenheden uit hoofdstuk 4, worden ontworpen in hoofdstuk 5. Na het ontwerp wordt de invoeringsprocedure van de pilot besproken in hoofdstuk 5. Vervolgens wordt de invoeringsprocedure van de pilot beschreven in hoofdstuk 6. In het laatste hoofdstuk worden de resultaten in overweging genomen en op basis daarvan wordt een oordeel over de pilot gegeven.

2. Functionele structuur

Het doel van dit hoofdstuk is om de functionele inhoud dusdanig te ontwerpen dat deze als basis kan dienen voor de bouw van de pilot. Deze pilot betreft het ontwerp van de communicatie tussen het DBMS en de applicatie.
Zoals ook al in de vorige pilot aan het licht is gekomen, is voor de communicatie tussen de database en de applicatie een stuk software nodig. Om dit stuk software aan te sturen zullen een aantal functionaliteiten moeten worden ontworpen. In paragraaf 2.1 worden de mogelijkheden waarop deze functionaliteiten worden ontworpen in kaart gebracht en wordt hieruit een keuze gemaakt. In paragraaf 2.2. wordt een visueel voorbeeld gegeven in de vorm van een sequencediagram. Hoofdstuk 2 wordt afgesloten met paragraaf 2.3 waarin de inhoud van de databaseklasse wordt toegelicht.

2.1. Communicatiemogelijkheden

Voordat de pilotdelen en bouweenheden vastgesteld kunnen worden, is het noodzakelijk om te onderzoeken hoe de wijze van communicatie zal gaan verlopen. Na dit onderzoek zal tevens een keuze gemaakt moeten worden, zodat op basis van deze keuze de overige functionaliteiten ontworpen kunnen worden:

2.1.1. Mogelijkheden

De manier van communiceren met de database zal als volgt gaan:

Stap	
1	De applicatie roept een functie aan
2	Deze functie vertaalt de aanroep naar een SQL-query
3	De SQL-query wordt uitgevoerd en het resultaat wordt teruggegeven aan de functie
4	De functie retourneert het resultaat.

De bovengenoemde stappen zijn vrij algemeen en zijn daarom op diversen manieren te implementeren. De eerste en vierde stap zullen echter maar op één manier kunnen worden geïmplementeerd. De stappen twee en drie zijn daarentegen op verschillende manieren te benaderen:

Stap	Optie	Benadering
2	a	De functie die de aanroep vertaald naar een SQL-query is eigendom van een klasse
2	b	De functie die de aanroep vertaald naar een SQL-query is eigendom van een aparte databaseklasse
2	c	De functie is vertaald naar code en direct in de applicatie opgenomen.
3	a	De eigenaar van de functie (klasse) voert de SQL-query uit en voert de nodige bewerkingen uit op het resultaat en retourneert het bewerkte

		resultaat aan de applicatie
3	b	De klasse roept een functie van de databaseklasse aan en deze voert de query uit en geeft het resultaat terug aan de eigenaar. De klasse zal op diens beurt de benodigde operaties uitvoeren en het bewerkte resultaat teruggeven aan de applicatie.
3	c	De databaseklasse voert de query uit en geeft het resultaat direct terug aan de applicatie, die vervolgens de bewerkingen hierop uitvoert.
3	d	De databaseklasse voert de query uit, bewerkt het resultaat en geeft deze terug aan de applicatie.
3	e	De applicatie voert de benodigde operaties uit om tot het gewenste resultaat komen.

2.1.2. Keuze

In de vorige paragraaf zijn een aantal mogelijkheden naar voren gekomen hoe de stappen twee en drie geïmplementeerd zouden kunnen gaan worden. Om uit deze mogelijkheden een keuze te kunnen maken, zijn deze onderworpen aan een aantal criteriapunten:

I. Is deze wijze van implementatie OO (Object Oriented).
II. Zorgt deze wijze van implementatie voor een overdraagbare applicatie.
III. In hoeverre is de applicatie – gezien de wijze van implementatie - uitbreidbaar.
IV. Is deze wijze van implementatie correct. (correct; gezien het gebruik van UML en Delphi).

Deze punten zijn opgesteld aan de hand van het software-engineering boek van Ghezzi en met behulp van ervaring met programmeren. De keuze is gemaakt door ieder punt te onderwerpen aan de criteria. Door een kruistabel op te stellen is in één oogopslag te zien welke opties het meest geschikt zijn als oplossing. De mate van toepassing (het gewicht) varieert van – (niet van toepassing / heel slecht) tot ++ (heel goed).

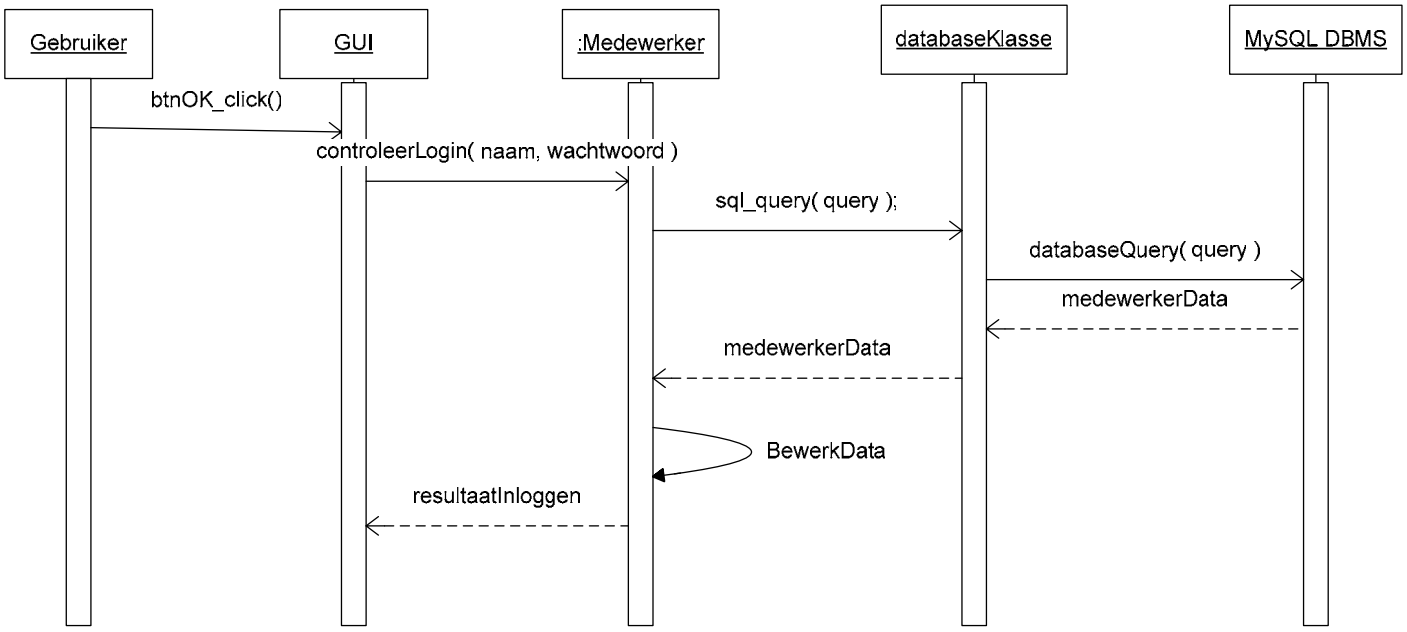
Stap+Optie	**Keuzemogelijkheden (I – IV)**			
	I	**II**	**III**	**IV**
2a	++	+	++	++
2b	+/-	+	+	+
2c	--	+/-	+	--
3a	++	+/-	+	+
3b	++	+	+	+
3c	-	+/-	+/-	-
3d	+	+	+	-
3e	-	-	+/-	--

Uit het bovenstaande diagram blijkt dat voor stap twee, optie ‘a’ de beste is. Voor stap drie blijkt dat optie ‘b’ de beste oplossing is. Dit leidt tot de conclusie dat de volgende stappen uitgevoerd gaan worden met betrekking tot de communicatie:

Stap	
1	De applicatie roept een functie aan
2	De functie die de aanroep vertaald naar een SQL-query is eigendom van een klasse
3	De klasse roept een functie van de databaseklasse aan en deze voert de query uit en geeft het resultaat terug aan de eigenaar. De klasse zal op diens beurt de benodigde operaties uitvoeren en het bewerkte resultaat teruggeven aan de applicatie.
4	De functie retourneert het resultaat.

2.2. Sequencediagram

In de vorige paragraaf is de wijze van communicatie vastgelegd. In deze paragraaf zal middels een sequencediagram een toelichting worden gegeven. Deze toelichting zal bestaan uit de use-case 'Inloggen', die reeds in de fase definitiestudie is ontworpen.



In bovenstaand diagram is te zien, dat de communicatie vanaf de gebruiker tot het DBMS drie klassen zal doorlopen:

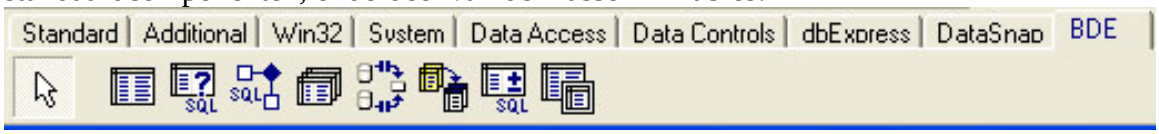
- De GUI; de Grafische User-interface waar de gebruiker op een 'button' klikt en hierbij de functie controleerLogin van de klasse Medewerker activeert.
- De Klasse Medewerker (volgens het klassendiagram); vertaalt de variabelen naar een geldige SQL-vraag en geeft deze door aan de functie sql_query van de databaseklasse.
- De databaseklasse verzorgt de connectie met de database, de foutafhandelingen en communiceert met 'MyODBC'.

2.3. Delphi en MySQL

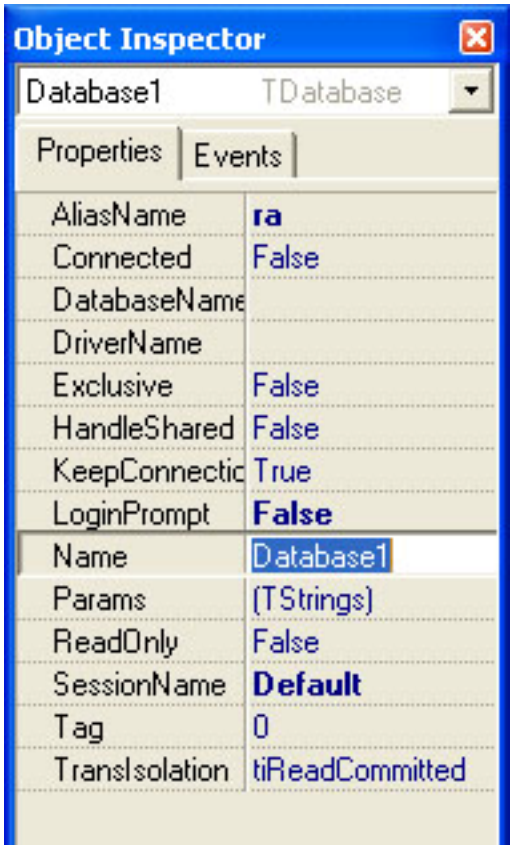
De database klasse is een klasse die in Delphi de queries zal gaan uitvoeren op het DBMS en het resultaat hiervan zal verwerken en teruggeven aan de ‘parent’ (een parent is de klasse die de uit te voeren functie heeft aangeroepen).
De wijze waarop Delphi omgaat met het verwerken van resultaten in queries en databases dient allereerst te worden onderzocht, zodat op basis daarvan de functionaliteiten van de klasse in kaart gebracht kunnen worden.

2.3.1. MySQL benadering

Delphi is voor de benadering van een DBMS in bezit van een aantal standaardcomponenten, onderdeel van de klasse DBTables:



Om een connectie met een database op te zetten wordt het component TDatabase gebruikt. In het onderstaande figuur is te zien hoe een instantie van TDatabasecomponent eruitziet.



Om met de instantie ‘Database1’ een connectie op te zetten met een MySQL-database zijn de volgende variabelen van belang.

- Connected
- AliasName / Databasename
- LoginPrompt
- Params

Zodra ‘*Connected*’ op True staat wordt verbinding gezocht met de MySQL-database met behulp van de variabelen *AliasName* of *DatabaseName*.

AliasName is hier de naam die toegekend is aan de ODBC-connectie, in dit voorbeeld ‘ra’.

DatabaseName is de werkelijke naam van de database, die alleen ingevuld hoeft te zijn, indien er geen sprake is van de *AliasName*.

Loginprompt zorgt ervoor dat er een loginscherm verschijnt zodra een verbinding met de database wordt gezocht.

Indien de *loginPrompt* op ‘false’ staat en er geen *AliasName* is opgegeven is een vereiste dat in *Params* (Parameters) de velden ‘User name’ en ‘Password’ zijn opgenomen met geldige waarden.

In dit geval zijn alleen de variabelen *AliasName* en Connected van belang.

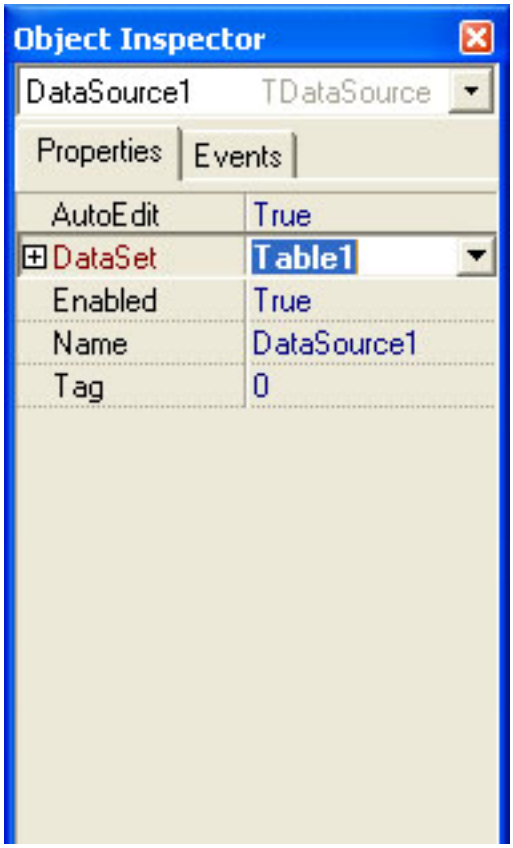
2.3.2. TDatasource, TQuery en TTable

Zodra er een connectie met de database gemaakt is, zijn er twee/drie extra componenten nodig die de communicatie met de database afhandelen:

- TDatasource
- TQuery
- TTable

De functie van drie componenten zullen hier stuk voor stuk behandeld worden.

TDatasource

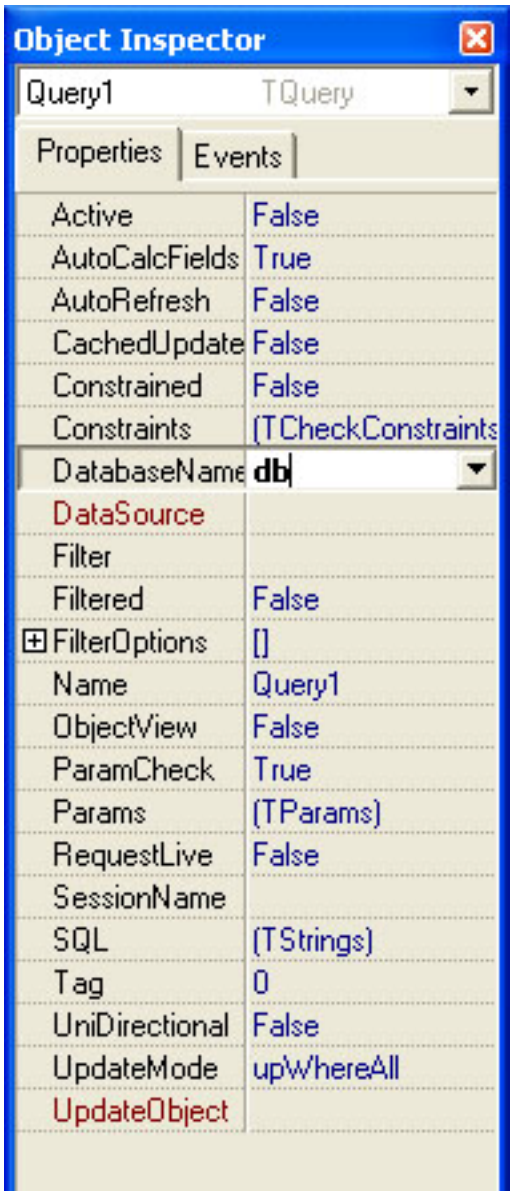


Dit component bezit de verzameling van data uit een bepaalde dataset. Zoals in het voorbeeld te zien is, is dat in dit geval de data uit een instantie van het component TTable, Table1.

De dataset kan naast de data uit een tabel ook de resultaten van een instantie van het component TQuery bevatten.

Het nut van dit component is, dat er bewerkingen op kunnen worden uitgevoerd, hetgeen niet mogelijk is op bijvoorbeeld een query of een table.

TQuery



Het component TQuery is noodzakelijk voor het uitvoeren van iedere query. Om een query te kunnen opzetten zal eerst een instantie van dit component worden aangemaakt, hetgeen in het voorbeeld is gedaan: ‘Query1’.

De volgende velden zijn bij het uitvoeren van *Query1* van belang:

- Active
- DatabaseName
- DataSource
- SQL

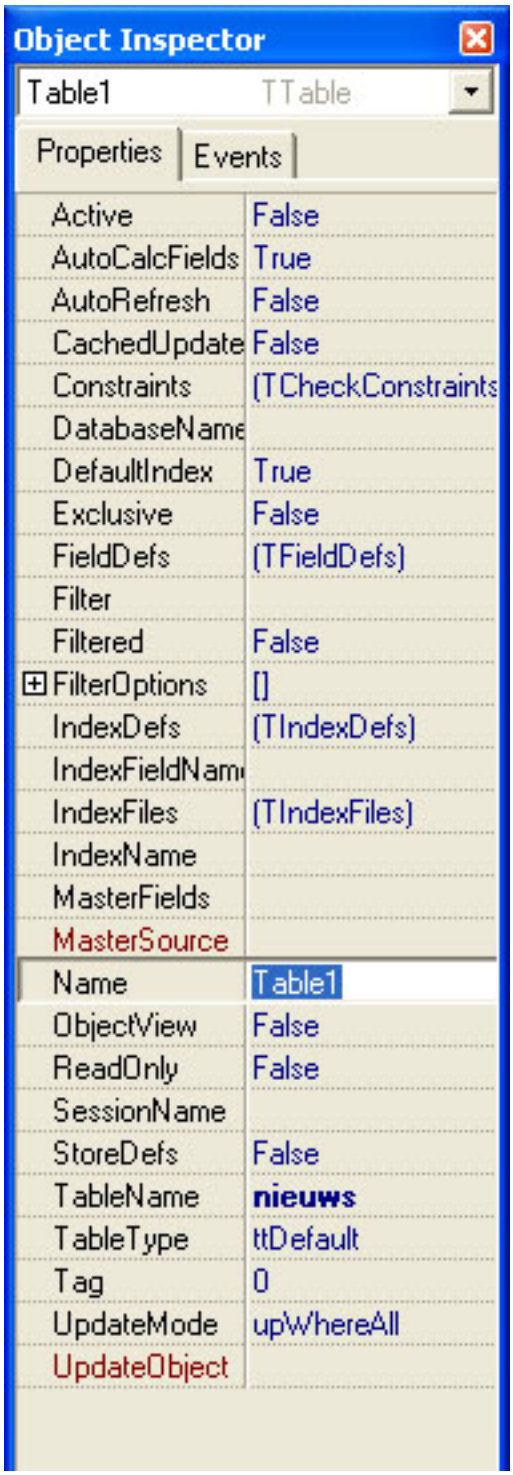
Zodra het veld *Active* op True wordt gezet, wordt Query1 uitgevoerd met de opgegeven query in SQL.

DatabaseName is de naam van een instantie van het component TDatabase, die de verbinding met de database verzorgt.

DataSource is een instantie van het component TDataSource. Deze instantie zal de resultaten van Query1 bevatten, zodra deze is uitgevoerd.

SQL dit is instantie van TStrings, waarin de query wordt geplaatst. TStrings is een verzameling van strings.

TTable



Tot slot het component TTable. Een instantie van het component TTable, 'Table1', is een doorgeefluik voor een willekeurige tabel in database. De gehele tabel zou in dit voorbeeld in een 'Database Grid' kunnen worden geplaatst, zodat de gehele tabel en de inhoud ervan zichtbaar is. Dit zou bijvoorbeeld handig kunnen zijn om een overzicht te krijgen van cliënten.

De variabelen die hier van belang zijn, zijn de volgende:

- Active
- DatabaseName
- TableName

De variabele *Active* activeert de instantie en haalt de gegevens uit de tabel in *TableName* op.

DatabaseName is, net als in de voorgaande componenten, de naam van een instantie van het component TDatabase.

TableName, dit is de naam van de tabel die zich in de database bevindt.

Met het gebruik van deze vier componenten is het opzetten van een verbinding met de MySQL database vanuit Delphi mogelijk.

2.3.3. Pre- en postcondities

De databaseklasse zal toegang moeten hebben tot de componenten die in de voorgaande paragraaf beschreven zijn. Om deze toegang en de controle hierop te kunnen realiseren zullen de componenten onderdeel moeten zijn van de databaseklasse.
Naast de vier componenten zullen er functies / procedures geschreven moeten worden die de connectie met de database afhandelen en de queries uitvoeren:

Functienaam	checkConnection
Toegang	Private
Procedure / Functie	Functie : Boolean
Preconditie	T
Postconditie	De status van de connectie met de database is gecontroleerd.
Beschrijving	Deze functie controleert of de verbinding met de database is gemaakt
Uitzondering(en)	-

Functienaam	connect
Toegang	Private
Procedure / Functie	Procedure
Preconditie	Er is nog geen connectie met de database
Postconditie	Connectie met de database tot stand gebracht
Beschrijving	Er wordt connectie met de database gemaakt
Uitzondering(en)	- Database is onbereikbaar - Gebruikersnaam en / of wachtwoord is onjuist voor MySQL benadering - MyODBC niet bereikbaar

Functienaam	sql_query
Toegang	Public
Procedure / Functie	Functie: TDatasource
Preconditie	Er is een query meegegeven aan de functie
Postconditie	De query is uitgevoerd en het resultaat is geretourneerd.
Beschrijving	Er wordt gecontroleerd of de verbinding met de database bestaat. Indien deze verbinding niet bestaat wordt deze alsnog gemaakt. Vervolgens wordt de query aangemaakt en uitgevoerd. Het resultaat van de query wordt geretourneerd en vastgelegd in de klasse.
Uitzondering(en)	- De connectie kan niet worden geactiveerd - De query kan niet worden uitgevoerd Alle uitzonderingen worden vastgelegd in een logbestand.

Functienaam	Select_table
Toegang	Public
Procedure / Functie	Functie: TDatasource
Preconditie	Er is een tabelnaam meegegeven aan de functie
Postconditie	De tabelnaam is uitgevoerd en geretourneerd.
Beschrijving	Er wordt gecontroleerd of de verbinding met de database bestaat. Indien deze verbinding niet bestaat wordt deze alsnog gemaakt. Vervolgens wordt de tabel geselecteerd en wordt de datasource hiervan geretourneerd
Uitzondering(en)	- De connectie kan niet worden geactiveerd - De tabel kan niet worden gevonden Alle uitzonderingen worden vastgelegd in een logbestand.

Functienaam	Create
Toegang	Public
Procedure / Functie	Constructor
Preconditie	T
Postconditie	De klasse is aangemaakt en er zijn instanties aangemaakt van alle componenten
Beschrijving	De oude constructor wordt aangeroepen. De overige componenten worden gecreëerd en de nodige variabelen worden vastgelegd.
Uitzondering(en)	- Instanties kunnen niet worden aangemaakt

Functienaam	Free
Toegang	Public
Procedure / Functie	Procedure /. Destructor
Preconditie	T
Postconditie	Het geheugen van de klasse en alle bijbehorende componenten is vrijgegeven.
Beschrijving	Alle instanties worden vernietigd en tot slot wordt de oude procedure Free aangeroepen om de instantie van de klasse zelf te vernietigen.
Uitzondering(en)	- Instanties kunnen niet worden vrijgegeven.

3.

Technische structuur

Dit hoofdstuk beschrijft de technische aspecten met betrekking tot het ontwikkelen van de databaseklasse. Deze aspecten zijn reeds globaal gedefinieerd tijdens de fase definitiestudie en zullen hier verder worden gedetailleerd. In paragraaf 3.1 wordt de fysieke allocatie toegelicht. In de daaropvolgende paragraaf worden de technische elementen van de databaseklasse toegelicht.

3.1. Fysieke allocatie

In het voorgaande hoofdstuk is beschreven hoe de databaseklasse er functioneel uitziet. Verschillende componenten leveren ieder een functionaliteit om de totstandkoming van de communicatie met het DBMS te verwezenlijken. Hieronder volgt de technische specificatie:

De klasse zal net als alle overige bestanden in dezelfde directory worden geplaatst en de unit zal ‘database’ gaan heten en als database.pas worden opgeslagen.

3.2. Databaseklasse

De databaseklasse zal TDb gaan heten en de bovenstaande componenten zullen worden toegevoegd aan de klasse. Dit geeft de onderstaande Delphi-klasse:

```
unit database;

interface

uses DBGrids, DB, DBTables, Grids, Dialogs, SysUtils, client,
logging;

type Tdb = class
  db: TDatabase;
  ds: TDatasource;
  q: TQuery;
  t: TTable;
  Log: TLog;
  error: string;
  public
    function sql_query( query: string ): TDatasource;
    function select_table( tableName: string ): TDatasource;
    constructor Create(); overload;
    procedure Free; overload;
  private
    procedure errorToLog( error: string );
    procedure writeToLog( logString: string);
    function checkConnection: boolean;
    procedure connect;
```

```
end;
```

Het ontstaan van de klasse wordt in hoofdstuk 5 van dit rapport toegelicht evenals het ontstaan van de procedures errorToLog en writeToLog.

4.

Pilotontwikkelplan

Deze pilot is verdeeld in verschillende pilotdelen, zoals besproken in de definitiestudie. Deze pilotdelen zijn het ontwikkelen van de functies voor het aansturen van de communicatie en het afhandelen van de gegevens. De laatste is in dit hoofdstuk opgesplitst in twee pilotdelen.

4.1. Pilotdelen

1. Realiseren van de databaseklasse
2. Procedures en functies implementeren
3. Foutafhandeling

De pilotdelen worden in de volgende paragrafen besproken en onderverdeeld in bouweenheden.

4.2. Realiseren van de databaseklasse

Voor het realiseren van de databaseklasse zijn twee bouweenheden gedefinieerd:

1. Het vaststellen in het juiste gebruik van Delphi met betrekking tot objecten
2. Implementeren van de databaseklasse

Het vaststellen in het juiste gebruik is eigenlijk een klein onderzoek naar hoe de opbouw van een nieuw object werkt en hoe deze in andere objecten aangeroepen kan worden. Aangezien dit een belangrijk object is en in de gehele applicatie beschikbaar moet zijn, maar niet meerdere malen mag worden aangemaakt. De reden hiervan wordt toegelicht in het volgende hoofdstuk.
De implementatie van de databaseklasse zal bestaan uit het bepalen van de naamgeving van de unit en de fysieke opslag in het project. Naast de naamgeving en opslag zal de eerder besproken databaseklasse in een unit van Delphi worden opgenomen en gecompileerd, zodat deze in het project kan worden gebruikt.

4.3. Procedures en functies implementeren

Dit pilotdeel zal bestaan uit twee verschillende bouweenheden die betrekking hebben op de bouw van de procedures en functies:

1. Ontwikkeling van de publieke functies.(public)
2. Ontwikkeling van de beschermde functies (private)

4.4. Foutafhandeling

Er is in dit stadium gekozen om foutafhandeling als een apart pilotdeel te beschouwen. De applicatie gaat er in de toekomst namelijk van uit dat de interface vlekkeloos werkt en de fouten binnen deze klasse worden afgehandeld door de databaseklasse.

Vanuit dit oogpunt is de keuze gemaakt om een pilotdeel te maken die zich specifiek met dit aspect van de klasse bezighoudt.
Om de aanpak structuur aan te brengen zijn de volgende bouweenheden binnen dit deel gedefinieerd:

1. Onderzoeken en aanpassen foutafhandeling functies en procedures.
2. Onderzoeken en ontwikkelen registratie foutafhandeling.

5.

Ontwerp software-bouweenheden

De bouweenheden zijn in het vorige hoofdstuk gedefinieerd en zullen hier op het ontwerp worden toegelicht. De bouweenheden – voorzover deze per pilotdeel zijn gedefinieerd – worden hier sequentieel behandeld. Na de behandeling in de eerste paragraaf worden testspecificaties per bouweenheid toegelicht in paragraaf 5.2.

5.1. Bouweenheden

De bouweenheden worden in de aankomende subparagrafen toegelicht op inhoud. De nadruk ligt vooral op het leggen van de basis van de bouw van deze eenheden.

5.1.1. Het vaststellen in het juiste gebruik ...

Zoals al eerder aangegeven bestaat het pilotdeel ‘realiseren van de databaseklasse’ uit twee bouweenheden. In deze subparagraaf wordt de eerste van de twee uitgewerkt.

Het meest belangrijke aspect van deze bouweenheid is dat de aanroep van dit object eenmalig gebeurd. Na deze aanroep (instantie) van het object moet het in staat worden gesteld dat deze in de gehele applicatie beschikbaar is. Mocht het zo zijn dat het object meerdere instanties heeft dan kan het zijn dat eventuele data uit de database verloren kan gaan in één van de instanties. Om dit te voorkomen is bekeken welke verschillende manieren een object kan worden opgebouwd in Delphi en welke van deze de juiste is met betrekking tot deze bouweenheid.

In Delphi 7 is het mogelijk om een los staan stuk software te ontwikkelen op drie manieren:

- Een DLL-bestand (bibliotheekbestand aanmaken).
- Middels een klasse binnen de bestaande applicatie
- Middels een nieuw bestand alleen voor de klasse

Een DLL-bestand

De eerste optie is het maken van een nieuw bestand met de extensie ‘DLL’ (Dynamic Linked Library). Het doel van de DLL is dat het meerdere malen gebruikt kan worden en een aantal cruciale functies en procedures bevat die ook door andere applicaties gebruikt kunnen worden.

Een klasse binnen de applicatie

Een klasse binnen een applicatie, wil zeggen dat de klasse wordt ondergebracht in een bestaande unit. Dit bestaande unit, mag dan nooit vrijgegeven of vernietigd worden en moet een globale variabele opnemen, zodat de databaseklasse bereikbaar blijft.

Nieuw bestand voor de klasse

Een bestand geheel toegewijd aan de databaseklasse. Deze kan gecompileerd worden, losstaand van de overige units.

Voordelen en nadelen.

Om een weloverwogen beslissing te maken voor het onderbrengen van de klasse is een lijst opgesteld met voor- en nadelen voor het gebruik van de drie alternatieven.

Alternatief	Voordelen	Nadelen
DLL-bestand	De functies van de klasse zijn herbruikbaar De applicatie hoeft het bestand maar één keer te initialiseren	Het bestand neemt relatief veel geheugen in beslag. Het dll-bestand kan worden afgesloten zonder dat de applicatie dit kan controleren
Klasse binnen de applicatie	De functies verbruiken weinig extra geheugen Ze zijn onderdeel van de applicatie	Een globale variabele moet worden gedefinieerd binnen een bestaande unit De bestaande instantie mag nooit worden vrijgegeven of beëindigd. (eventueel verlies van data).
Nieuw bestand	De functies verbruiken weinig extra geheugen De unit staat een 'overload' van de functies 'create' en 'free' toe	Er zal een globale variabele moeten worden aangemaakt om één instantie van het object te garanderen.

Het tweede alternatief valt al snel af door het tweede nadeel. Het is bovendien niet 'netjes' (onderhoudbaarheid) om een aparte klasse onder te brengen in een bestaande klasse.
Het eerste alternatief is een goed alternatief, ware het niet dat een dll-bestand door de gebruiker beëindigd kan worden zonder toestemming van de applicatie die deze gebruikt. De gegevens die op dat moment in het geheugen van het dll-bestand aanwezig zijn kunnen op deze manier verloren gaan.
Het laatste alternatief is daarom, afgezien van een klein nadeel, de beste oplossing.

5.1.2. Implementeren van de databaseklasse

Het implementeren van de databaseklasse bestaat uit de fysieke opslag en daarbij ook direct de naamgeving van de unit.

In Delphi is het zo dat de naam van unit gelijk moet zijn aan de fysieke naam met de extensie '.PAS'. Indien dit niet het geval is, zal de unit niet compileren. Aangezien het

hier om een databaseklasse gaat, is de unit ‘database’ genoemd. Waarbij de fysieke naam ‘database.PAS’ is.
De applicatie kan nu volstaan met het ‘includen’ van de unit database. Dit wil zeggen dat de applicatie alle functies van de unit database kan gebruiken in de applicatie. Maar het zijn hier geen functies, maar het is een klasse die beschikbaar moet zijn. De constructie van een unit in Delphi ziet er als volgt uit:

```
unit database;

uses ... ;

interface

...

implementation

...

end.
```

In het interface gedeelte worden alle constanten en typedefinities aangemaakt. In dit geval moet er een type aangemaakt worden die alle eerder beschreven functies ondersteund, public en private. Omdat de naamgeving van dit type niet gelijk mag zijn aan de naam van de unit, is gekozen voor een afkorting tot ‘Db’. Bovendien wordt iedere klasse in Delphi voorzien van ‘T’ voor de klassennaam, resulterend in de klassennaam TDb.

```
unit database;

uses DBGrids, DB, DBTables, Grids, Dialogs, SysUtils, client,
logging;

type Tdb = class
  db: TDatabase;
  ds: TDatasource;
  q: TQuery;
  t: TTable;
  Log: TLog;
  error: string;
public
    function sql_query( query: string ): TDatasource;
    function select_table( tableName: string ): TDatasource;
    constructor Create(); overload;
    procedure Free; overload;
  private
    procedure errorToLog( error: string );
    procedure writeToLog( logString: string);
    function checkConnection: boolean;
    procedure connect;
end;

implementation

...

end.
```

Het implementatie gedeelte van de klasse bevat de functiebeschrijvingen en de globale variabelen. Die worden ontwikkeld in de volgende bouweenheden.

5.1.3. Ontwikkeling van de publieke functies

De publieke functies van de klasse zijn de functies die ook buiten een instantie van het object beschikbaar zijn. Zoals al eerder aangegeven zijn dat de functies sql_query, select_table, create en free. Deze functies worden in de subparagraaf sequentieel behandeld.

Sql_query

Allereerst deze functie controleren of er een geldige connectie met de database bestaat door de beschermde functie checkConnection aan te roepen. Vervolgens wordt de query uitgevoerd en wordt het resultaat in een de variabele ‘ds’ gezet en teruggegeven aan de parent van de functie.

Select_table
Deze functie controleert net als de functie sql_query de connectie. De functie selecteert vervolgens de gegeven tabel en zet de tabel in ‘t’ en geeft het resultaat terug in aan de parent.

Create
Deze functie heeft een extra attribuut genaamd ‘overload’. Dit attribuut wil zeggen dat de bestaande functie create wordt overschreven met een eigen variant van create. Om het hele proces van een create niet te herschrijven wordt de bestaande functie hergebruikt en worden alle variabelen na het creatiestatement geinitialiseerd.

```
Constructor TDb.Create();
begin
  inherited Create();

  self.db := TDatabase.Create(nil);
  self.ds := TDatasource.Create(nil);
  self.q := TQuery.Create(nil);
  self.t := TTable.Create(nil);
  self.log := TLog.Create;

  self.Log.path := 'C:\';
  self.Log.fileName := 'database.log';
  self.db.AliasName := 'db';
  self.db.DatabaseName := 'db';
  self.db.LoginPrompt := false;

end;
```

Free
Deze functie geeft de instantie van het object vrij zodra de applicatie wordt afgesloten. Omdat de variabelen van de databaseklasse zelf ook instanties van klassen zijn en daarom geheugen in beslag nemen, moeten ook deze worden vrijgegeven. Als dat niet wordt gedaan loopt het geheugen van het werkstation langzaam ‘vol’ en is uiteindelijk een herstart nodig om het geheugen vrij te maken.
Deze procedure is omgekeerde versie van create, dat wil zeggen dat het ook de bestaande functie overerft. Het verschil is alleen dat deze de functie pas na alle operaties overerft, anders zou de instantie worden vrijgegeven voordat de operaties kunnen worden uitgevoerd.

5.1.4. Ontwikkeling van de beschermde functies

In deze klasse zijn er vier beschermde functies, waarvan er twee niet eerder zijn gedefinieerd, namelijk errorToLog en writeToLog. Allereerst zal ik de al eerder besproken functies toelichten: checkConnection en connect.

checkConnection
De functie checkConnection is een functie met als resultaat een variabele van het type Boolean. Deze controleert of de er een geldige connectie is met de database. Indien dit het geval is wordt een ‘true’ teruggegeven, in het andere geval een ‘false’.

Connect
Deze procedure maakt een connectie met de database.

De nog niet eerder besproken procedures zijn ontworpen in het kader van de registratie van de foutafhandeling en worden in laatste bouweenheid behandeld.

5.1.5. Onderzoeken en aanpassen foutafhandeling bestaande functies

Het onderzoeken en aanpassen van de foutafhandeling is het controleren voor alle functies op punten waar deze een ongewenst resultaat zou kunnen genereren. Op al deze punten wordt vervolgens een mechanisme geïnstalleerd die de fout afvangt en het ongewenste resultaat onopgemerkt afhandelt. Dit wordt door middel van het ‘try ... catch except’ principe gedaan.

5.1.6. Ontwikkelen en implementeren registratie foutafhandeling

Met de registratie van de foutafhandeling wordt bedoeld dat deze wordt opgeslagen op een veilige locatie, zodat deze buiten de applicatie om – indien nodig – kan worden opgevraagd.
Ondanks dat de gebruiker niets mag merken van een fout, wordt deze onopgemerkt afgehandeld zodat verder kan worden gewerkt met de applicatie. Om de registratie van fouten en beveiligingen is besloten een log-bestand bij te houden van de afhandeling.

Om succesvol de afhandeling te registreren zijn twee functies ontworpen:

- errorToLog
- writeToLog

De eerste registreert een fout en de tweede functie registreert een eventuele beveiligingsovertreding (zoals een foutieve gebruikersnaam of wachtwoord bij het inloggen).

Om dit te kunnen realiseren is er in de klasse een variabele log van het type TLog aangemaakt. Dit type is een zelfgemaakte klasse en een instantie van dit type ondersteund het schrijven naar en lezen van een bestand op het werkstation. De instantie hiervan

wordt aangemaakt in de beschermde functie van de databaseklasse Create. Bovendien wordt naast een log-bestand tevens de laatste opgetreden fout opgenomen in een variabele ‘error’, zodat deze eventueel teruggekoppeld kan worden aan de gebruiker. Indien dit wordt teruggekoppeld wordt, moet dit uiteraard heel concreet en duidelijk worden geformuleerd, zodat het voldoet aan de niet-functionele systeemeis: “*Foutmeldingen moeten concreet en begrijpelijk zijn*“.

De klasse TLog ziet er als volgt uit:

```
Type
  TLog = class
    fileName: string;
    path: string;
    logFile: textfile;
    public
      procedure write( value: string );
      procedure error( value: string );
end;
```

De procedure ‘write’ zal de datum van het registeren in het bestand moeten vastleggen en controleren of het bestand leeg is. Als er reeds naar het bestand is geschreven, zal het bestand moeten worden aangevuld en niet worden overschreven.
Naast de controle op de inhoud van het bestand, zal ook het bestaan hiervan moeten worden gecontroleerd. Ook de locatie zal moeten worden gecontroleerd.
De procedure ‘error’ zal de procedure ‘write’ gebruiken om de fouten vast te laten leggen.

errorToLog
Deze procedure roept de publieke procedure ‘error’ van TLog aan.

writeToLog
Deze procedure roept de publieke procedure ‘write’ van TLog aan.

5.2. Testspecificaties

De testspecificaties zijn opgesteld per bouweenheid. Deze specificaties houden in dat er gecontroleerd wordt volgens het black-box principe. Dit principe houdt in dat er getest wordt of de gegeven invoer ook de verwachte uitvoer geeft. De mislukte cases worden in het hoofdstuk ‘Testen en beoordelen pilot’ toegelicht.

5.2.1. Het vaststellen in het juiste gebruik van Delphi met betrekking tot objecten

Case #	Test	Invoer	Verwachte uitvoer
1.0	De unit database klasse kan worden gebruikt	In Delphi : Uses database	Er treden geen compilatiefouten op.

5.2.2. Implementeren van de databaseklasse

Voor het testen van deze bouweenheid zijn enkele debugberichten opgenomen in de functies. Aangezien de testcases uitgevoerd dienen te worden, nadat de bouweenheid afgerond is, bestaan de functies alleen uit naamgeving. De inhoud van de functies is in de vorm van een messageBox, waarin de gegeven variabele zal worden weergegeven.

Case #	**Test**	**Invoer**	**Verwachte uitvoer**
2.0	De databaseklasse kan worden aangemaakt	dbKlasse = TDb.Create()	dbKlasse is een instantie van de databaseklasse
2.1	De functies kunnen worden aangeroepen	dbKlasse.sql_query(‘test1’)	MessageBox met ‘test1’
		dbKlasse.Select_table(‘test2’)	MessageBox met ‘test2’
		dbKlasse.free()	dbKlasse is vrijgegeven
		dbKlasse.errorToLog(‘test3’)	MesageBox met ‘test3’
		dbKlasse.writeToLog(‘test4’);	MessageBox met ‘test4’
		dbKlasse.checkConnection	False
		dbKlasse.connect	MessageBox met ‘test’
2.2	De databaseklasse kan worden vrijgegeven	dbKlasse.Free	dbKlasse is nil

5.2.3. Ontwikkeling van de publieke functies.(public)

De testspecificaties van deze bouweenheid, zullen vooral gericht zijn op het resultaat van de functies. Om dit resultaat te kunnen controleren is een tijdelijk databaseGrid gebruikt, waarin een TDatasource kan worden weergegeven. De naam van het databaseGrid is voor het gemak afgekort tot dbGrid.
Tevens is de tabel medewerker gevuld met twee medewerkers, om het resultaat te kunnen weergeven.

Case #	**Test**	**Invoer**	**Verwachte uitvoer**
3.0	De create functie werkt correct	dbKlasse = TDb.Create()	dbKlasse is een instantie van TDb De variabelen db, ds, q, t en Log zijn ook geinitialiseerd.
3.1	Selecteren van een	dbKlasse.select_table(	dbGrid bevat de tabel

	tabel geeft een geldige tabel terug	'medewerker')	medewerker.
3.2	Het uitvoeren van de functie sql_query werkt correct	dbKlasse.sql_query('Select * from medewerker')	dbGrid bevat de twee medewerkers.
3.3	De Free functie geeft alle variabelen en de dbKlasse vrij	dbKlasse.Free()	De variabelen db, ds, q, t en Log zijn vrijgegeven. De dbKlasse is vrijgegeven.

5.2.4. Ontwikkeling van de beschermde functies (private)

Case #	**Test**	**Invoer**	**Verwachte uitvoer**
4.0	Er kan een verbinding worden opgezet met de database	dbKlasse.connect	Connectie met de database bestaat.
4.1	Controleren van de connectie geeft het juiste resultaat	dbKlasse.connect() dbKlasse.checkConnection	Resultaat is true

5.2.5. Onderzoeken en aanpassen foutafhandeling functies en procedures.

Het doel van de controle op deze bouweenheid, is het vaststellen of de fouten die kunnen optreden door de opgestelde functies op de juiste manier worden afgevangen. Voor een aantal belangrijke functies, die tot deze bouweenheid zijn ontwikkeld, is minimaal één testcase opgesteld om dit te controleren.

Case #	**Test**	**Invoer**	**Verwachte uitvoer**
5.0	foutafhandeling TDb.create	Self.db.aliasName = 'bestaat_niet' TDb.create	Fout wordt vastgelegd in error en weergegeven.
5.1	Foutafhandeling select_table	dbKlasse.select_table('bestaat_niet')	Fout wordt vastgelegd in error. Resultaat is nil
5.2	Foutafhandeling sql_query	dbKlasse.sql_query(' Ongeldige query ')	Fout wordt vastgelegd in error. Resultaat is nil
5.3	Foutafhandeling connect	MyODBC verwijderd. dbKlasse.connect	Fout wordt vastgelegd in error. Terugkoppeling aan gebruiker

5.2.6. Onderzoeken en ontwikkelen registratie foutafhandeling.

Om de klasse TLog te kunnen controleren,wordt allereerst een instantie van het object aangemaakt. Bovendien krijgen de variabelen: ‘filename’, ‘path’ en ‘logFile’, de waarden zoals die ook in de toekomstige applicatie zullen worden gebruikt. Zie ook de toelichting bij de functie Create van bouweenheid 3.

Case #	**Test**	**Invoer**	**Verwachte uitvoer**
6.0	TLog klasse kan worden geinitialiseerd	Log = TLog.create	Log is een instantie van TLog
6.1	Controleren functie write.	Log.write(‘Test-Functie’)	Het bestand ‘C:\Database.log’ bevat de tekst: ‘Test-Functie’.
6.2	Controleren functie error	Log.error(‘Foutmelding’)	Het bestand ‘C:\database.log’ bevat de tekst ‘Foutmelding’

6. Invoeringsprocedure Pilot

Wanneer de pilot functioneert volgens de opgestelde eisen en de testgevallen succesvol zijn doorlopen, kan worden overgegaan tot het operationeel maken van de pilot. Dit hoofdstuk beschrijft de procedure die dient te worden gevolgd wanneer de pilot wordt ingevoerd. In paragraaf 6.1 wordt een plan opgesteld dat als uitgangspunt dient bij de invoering van de pilot. Vervolgens worden in paragraaf 6.2 testspecificaties opgesteld. De testgevallen kunnen worden uitgevoerd nadat de pilot is ingevoerd en dienen ter controle op een correcte invoering.

6.1. Invoeringsplan

In deze subparagraaf is een invoeringsprocedure geschreven die toelichting geeft op de correcte invoering van de pilot. Deze procedure bestaat uit een klein stappenplan:

1. Controle invoering hard- en software
   Voordat met de invoering kan worden begonnen moet eerst worden gekeken of de onderstaande software nog operationeel is:
   a. MySQL
   b. MyODBC
2. Invoeren nieuwe applicatie
   Vervolgens wordt de opnieuw gecompileerde applicatie overgezet, aangezien de databaseklasse onderdeel is van de applicatie.

6.2. Testspecificaties

Hieronder is een lijst met testspecificaties opgenomen die betrekking hebben op het invoeringsplan.

Testspecificatie Invoeringsplan			
Case #	**Test**	**Invoer**	**Verwachte uitvoer**
7.0	De pakketten zijn operationeel	Testbutton MyODBC wordt uitgevoerd	Connectie met de database kan worden gemaakt
7.1	De nieuwe applicatie is operationeel	Testcases wederom uitvoeren	Testcases zijn succesvol doorlopen

7. Testen en beoordelen Pilot

Na de implementatie van iedere bouweenheid is deze getest aan de hand van de specificaties in hoofdstuk 5 van dit document. In dit hoofdstuk zijn de resultaten hiervan beschreven na de implementatie van iedere bouweenheid. Paragraaf 7.1 beschrijft de resultaten van testcases en de omgeving waarin deze zijn uitgevoerd en in paragraaf 7.2 wordt de pilot beoordeeld.

7.1. Testomgeving

Om de tests van de bouweenheden uit te voeren is gekozen voor een lokale testomgeving. De specificaties van dit systeem zijn hieronder opgenomen:

- Athlon XP 2400+
- 1280 MB RAM

De volgende softwarepakketten waren hier op geïnstalleerd:

- Borland Delphi Enterprise Edition
- Windows XP besturingssysteem
- Microsoft Office XP
- MySQL DBMS
- MyODBC

7.1.1. Testen bouweenheid 1

Case #	Test	Invoer	Verwachte uitvoer	Resultaat
1.0	De unit database klasse kan worden gebruikt	In Delphi : Uses database	Er treden geen compilatiefouten op.	OK

7.1.2. Testen bouweenheid 2

Case #	Test	Invoer	Verwachte uitvoer	Resultaat
2.0	De databaseklasse kan worden aangemaakt	dbKlasse = TDb.Create()	dbKlasse is een instantie van de databaseklasse	OK
2.1	De functies kunnen worden aangeroepen	dbKlasse.sql_query(‘test1’)	MessageBox met ‘test1’	OK
		dbKlasse.Select_table(‘test2’)	MessageBox met ‘test2’	OK
		dbKlasse.free()	dbKlasse is vrijgegeven	OK
		dbKlasse.errorToLog(‘test3’)	MesageBox met ‘test3’	OK
		dbKlasse.writeT	MessageBox met	OK

		oLog(‘test4’);	‘test4’	
		dbKlasse.checkConnection	False	OK
		dbKlasse.connect	MessageBox met ‘test’	OK
2.2	De databaseklasse kan worden vrijgegeven	dbKlasse.Free	dbKlasse is nil	OK

7.1.3. Testen bouweenheid 3

Case #	**Test**	**Invoer**	**Verwachte uitvoer**	**Resultaat**
3.0	De create functie werkt correct	dbKlasse = TDb.Create()	dbKlasse is een instantie van TDb De variabelen db, ds, q, t en Log zijn ook geinitialiseerd.	Mislukt!
3.1	Selecteren van een tabel geeft een geldige tabel terug	dbKlasse.select_table(‘medewerker’)	dbGrid bevat de tabel medewerker.	OK
3.2	Het uitvoeren van de functie sql_query werkt correct	dbKlasse.sql_query(‘Select * from medewerker’)	dbGrid bevat de twee medewerkers.	Mislukt
3.3	De Free functie geeft alle variabelen en de dbKlasse vrij	dbKlasse.Free()	De variabelen db, ds, q, t en Log zijn vrijgegeven. De dbKlasse is vrijgegeven.	OK

7.1.4. Testen bouweenheid 4

Case #	**Test**	**Invoer**	**Verwachte uitvoer**	**Resultaat**
4.0	Er kan een verbinding worden opgezet met de database	dbKlasse.connect	Connectie met de database bestaat.	OK
4.1	Controleren van	dbKlasse.connec	Resultaat is true	OK

	de connectie geeft het juiste resultaat	t() dbKlasse.check Connection		

7.1.5. Testen bouweenheid 5

Case #	**Test**	**Invoer**	**Verwachte uitvoer**	**Resultaat**
5.0	foutafhandeling TDb.create	Self.db.aliasName = ‘bestaat_niet’ TDb.create	Fout wordt vastgelegd in error en weergegeven.	OK
5.1	Foutafhandeling select_table	dbKlasse.select_table(‘bestaat_niet’)	Fout wordt vastgelegd in error. Resultaat is nil	OK
5.2	Foutafhandeling sql_query	dbKlasse.sql_query(‘ Ongeldige query ’)	Fout wordt vastgelegd in error. Resultaat is nil	OK
5.3	Foutafhandeling connect	MyODBC verwijderd. dbKlasse.connect	Fout wordt vastgelegd in error. Terugkoppeling aan gebruiker	OK

7.1.6. Testen bouweenheid 6

Case #	**Test**	**Invoer**	**Verwachte uitvoer**	**Resultaat**
6.0	TLog klasse kan worden geinitialiseerd	Log = TLog.create	Log is een instantie van TLog	OK
6.1	Controleren functie write.	Log.write(‘Test-Functie’)	Het bestand ‘C:\Database.log’ bevat de tekst: ‘Test-Functie’.	OK
6.2	Controleren functie error	Log.error(‘Foutmelding’)	Het bestand ‘C:\database.log’ bevat de tekst	OK

			‘Foutmelding’	

7.1.7. Testen invoeringsprocedure

Testspecificatie Invoeringsplan				
Case #	**Test**	**Invoer**	**Verwachte uitvoer**	**Resultaat**
7.0	De pakketten zijn operationeel	Testbutton MyODBC wordt uitgevoerd	Connectie met de database kan worden gemaakt	OK
7.1	De nieuwe applicatie is operationeel	Testcases wederom uitvoeren	Testcases zijn succesvol doorlopen	OK

7.2. Beoordeling Pilot

Naast de mislukte cases in de derde bouweenheid, is de ontwikkeling en het testen over het algemeen goed verlopen. Op dit moment is het mogelijk om via benadering van deze klasse een sql query uit te voeren op de database en het resultaat weer te geven. De verbinding met de database wordt opgezet en de klasse handelt alle fouten af en bovendien worden deze ook geregistreerd in een daarvoor bestemd bestand.

Het gewenste resultaat zoals die in de definitiestudie is gedefinieerd:

- Het gewenste resultaat van deze pilot is het afvangen van de communicatie tussen de database en de grafische user interface. De functies zullen geheel functioneel moeten zijn of in het uiterste geval een vergelijkbare testwaarde moeten teruggeven, zodat de subsystemen deze zonder problemen kunnen aanroepen

De interface heeft naast een functioneel karakter ook de registratie van de fouten afgevangen, wat resulteert in een pilot met het gewenste resultaat.

Conclusie

In dit document is een ontwerp gemaakt dat ter ondersteuning heeft gediend bij de implementatie van de derde pilot, waarin de GUI-database Interface wordt gerealiseerd. Na de implementatie van iedere bouweenheid zijn de opgestelde testspecificaties uitgevoerd. De resultaten hebben uitgewezen dat de pilot voldoet aan alle eisen die zijn vastgesteld in de definitiestudie.

Bijlage A: Plan van aanpak pilot 3 'GUI-Database Interface'

Inleiding

In dit document worden de onderdelen van de derde pilot toegelicht en in een planning opgenomen. Dit plan van aanpak is een detailplanning voor de derde pilot en toevoeging op het plan van aanpak voor het gehele project.
Allereerst zullen de resultaten die de pilot oplevert worden beschreven en vervolgens zal een planning gemaakt worden voor de uitvoering van de pilot.
Voor informatie over kwaliteitsborging en de projectorganisatie wordt verwezen naar het plan van aanpak voor het gehele project.

Activiteiten pilotontwikkeling

Hieronder volgt een overzicht van de activiteiten die worden uitgevoerd tijdens de ontwikkeling van de eerste pilot. Per activiteit worden deelactiviteiten beschreven en er wordt aangegeven wat het resultaat is van deze activiteit.

- Specificeren globaal-functionele structuur pilot
    - Bepalen functionele reikwijdte pilot;
    - Detailleren GUI schermen door middel van sequencediagram(men);
    - Ontwerpen prototype gebruikersinterface;
    - Resultaat: ontwerp van functionele structuur.
- Specificeren globaal-technische structuur pilot
    - Bepalen fysieke allocatie
    - Specificeren technisch opslagmodel
    - Resultaat: ontwerp van technische structuur.
- Opstellen pilotontwikkelplan
    - Definiëren pilotdelen
    - Definiëren bouweenheden
    - Voorbereiden beoordeling en testen pilotdelen;
    - Resultaat: ontwikkelplan voor pilot.
- Ontwerpen software-bouweenheden
    - Specificeren integratie en testen bouweenheden
    - Opstellen detailspecificaties
    - Opstellen testspecificaties
    - Resultaat: gedetailleerd ontwerp pilotdelen.
- Bouwen software-bouweenheden
    - Coderen pilotdelen
    - Voorbereiden component-test
    - Uitvoeren component-test
    - Corrigeren component
    - Resultaat: implementatie van pilot.
- Opstellen invoeringsprocedure pilot
    - Ontwerpen invoeringsprocedure
    - Specificeren tests invoeringsprocedures
    - Testen invoeringsprocedures

    - Corrigeren invoeringsprocedures
    - Resultaat: beschrijving invoeringsprocedure.
- Beoordelen en testen pilot
    - Opzetten testomgeving
    - Beoordelen en testen pilot
    - Analyseren uitkomsten
    - Corrigeren fouten
    - Rapporteren over beoordelen en testen

Naast deze stappen zijn een aantal volgens IAD voorgeschreven ontwikkelstappen bewust overgeslagen. Deze stappen zullen geen toegevoegde waarde leveren op het ontwerp of de ontwikkeling van de pilot.

De overige activiteiten die IAD voorschrijft worden niet voor dit product uitgevoerd.

- Voorbereiden pilotontwerp-workshop
  Een workshop voor het gehele project is reeds gehouden in de definitiefase.
- Specificeren globaal-organisatorische inrichting
  De invloed op de organisatorische aspecten zijn beschreven in de definitiefase. Deze pilot levert geen extra veranderingen met zich mee..
- Aanpassen externe componenten
  Er hoeven geen externe componenten te worden aangepast.
- Wijzigen andere informatiesystemen
  Er is geen sprake van een ander informatiesysteem.
- Bouwen conversie-tools
  Deze pilot vereist geen conversie-tools
- Ontwerpen handmatige procedures
  De handmatige procedures zijn hier niet van toepassing.
- Integreren bouweenheden
  De bouweenheden worden bij implementatie direct geïntegreerd, doordat deze onderdeel zijn van de applicatie.
- Samenstellen opleidingsmateriaal pilot
  Er is geen behoefte aan opleidingsmateriaal voor enkel deze pilot.
- Samenstellen handleiding pilot
  Er behoeft geen handleiding te worden geschreven voor deze pilot

Planning

Hieronder volgt een planning van de uit te voeren activiteiten.

Product / Proces	Tijdsduur in uren
Specificeren globaal-functionele structuur pilot	40
Specificeren globaal-technische structuur pilot	15
Opstellen pilotontwikkelplan	6
Ontwerpen software-bouweenheden	25
Bouwen software-bouweenheden	30
Opstellen invoeringsprocedure pilot	5
Beoordelen en testen pilot	2
Totaal aantal uren	123

Bijlage
Pilotontwikkelrapport
Pilot 4: “Subsystemen één tot en met drie”

Voorwoord

Voor u ligt het pilotontwikkelrapport dat ter ondersteuning dient bij de implementatie van de vierde pilot uit het project W!se delivery. Tijdens deze pilot worden de functionaliteiten, die in de definitiestudie als de eerste drie subsystemen zijn gedefinieerd.

In het systeemontwikkelingtraject van IAD volgt na de definitiefase, de fase van de pilotontwikkeling. Tijdens de pilotontwikkeling worden de pilots die in de definitiefase zijn gedefinieerd, ontwikkeld en geïmplementeerd.

Dit document is tot stand gekomen in overleg met Mevr. S. van Iterson, opdrachtgever bij de verloskundepraktijk de Maatschap.

Leiden, 7 januari 2005

Cees Kruijf
Verloskundepraktijk de Maatschap.

Inhoudsopgave

1. Inleiding

Tijdens de definitiestudie is het project verdeeld in vijf pilots. Deze pilots worden in het pilotontwikkelingstraject sequentieel ontwikkeld. De eerste pilot bestond uit het ontwerpen en ontwikkelen van de Grafische User Interface. De tweede pilot bestond uit het ontwikkelen van de databasestructuur. In de derde pilot is de interface tussen de GUI en de database ontwikkeld. In de vierde pilot worden de functionaliteiten van de eerste drie subsystemen ontworpen en geïmplementeerd. Dit document dient ter ondersteuning van de ontwikkeling van vierde pilot.

Het doel van dit rapport is om de functionaliteiten van de drie subsystemen zodanig te ontwerpen, dat op basis hiervan met de implementatie kan worden begonnen. Daarnaast worden voor iedere bouweenheid in dit document testen gespecificeerd. De uitvoering van de test maakt het mogelijk om de ontworpen bouweenheden te valideren en verifiëren op de inhoud. Mede op basis van de resultaten van deze tests wordt een beslissing genomen over de verdere ontwikkeling.

De opbouw van dit document is als volgt, in het volgende hoofdstuk wordt de functionele structuur ontworpen. In hoofdstuk 3 wordt een pilotontwikkelplan opgesteld, waarin tevens de bouweenheden worden gedefinieerd. Het ontwerp van deze bouweenheden wordt in het daaropvolgende hoofdstuk 4 behandeld. Na het ontwerp wordt de invoeringsprocedure van de pilot besproken in hoofdstuk 5 en in hoofdstuk 6 wordt de pilot getest en beoordeeld. In het laatste hoofdstuk worden de resultaten in overweging genomen en op basis daarvan wordt een oordeel over de pilot gegeven.

2.

Functionele structuur

Het doel van dit hoofdstuk is om de functionele inhoud van de pilot dusdanig te ontwerpen dat de basis voor de bouw van de pilot gelegd is. Het ontworpen systeemconcept zal hier in detail worden uitgewerkt, waarbij vooral de interactie tussen de objecten en die met het systeem worden uitgebreid. De reikwijdte van de functionaliteit van de pilot wordt in de eerste paragraaf uitgewerkt. Vervolgens worden in de daaropvolgende paragrafen de sequencediagrammen behandeld voor de subsystemen één tot en met drie.

2.1. Functionele reikwijdte

In het rapport definitiestudie zijn UML use-cases ontworpen, die de interactie met het systeem in kaart brengen. Deze use-cases zijn vervolgens opgedeeld in subsystemen naar aanleiding van de systeemeisen. De subsystemen in combinatie met de use-cases vormen het uitgangspunt voor de ontwikkeling van de sequence-diagrammen. Het sequence-diagram geeft inzicht in de interactie met de objecten. Dit inzicht is nodig voor de verdere ontwikkeling en bouw van de use-cases.
De functionele reikwijdte van de pilot geeft aan in hoeverre de opgestelde systeemeisen in de definitiefase worden gedekt. Hierbij wordt onderscheid gemaakt tussen de functionele en niet-functionele systeemeisen.

2.1.1. Functionele systeemeisen

In de definitiefase is vastgelegd dat de systeemeisen worden ingedeeld volgens het MoSCoW-principe. Daaruit is ook gebleken dat de pilots ontwikkeld zouden worden volgens een aantal subsystemen, waarbij de eisen met de “Must Have”, “Should Have” en “Could Have” prioriteit onderverdeeld zijn in respectievelijk subsysteem één, twee en drie.

De functionele reikwijdte van deze pilot zal dus de eisen met de prioriteit ‘M’, ‘S’ en ‘C’ moeten dekken, hieronder is hiervan een overzicht opgenomen.

Code	Systeemeis	Prioriteit
M1	Het systeem moet de gegevens op de oude zwangerschapskaart kunnen realiseren.	M
M2	De gegevens van cliënten die verwerkt worden in het systeem moeten grotendeels variabel zijn. (Doktoren, ziekenhuizen, ziektes, etc)	M
M3	Het moet mogelijk zijn om te zoeken naar de cliënten op eventueel unieke gegevens	M
M4	Het moet mogelijk zijn om ziekenhuizen bij te werken en er in te zoeken	M

M5	Ziekenhuizen, verzekeraars, gynaecologen, doktoren, plaatsnamen, achternamen en voornamen. Deze gegevens moeten allemaal in een ‘zelfgroeiende’ lijst kunnen worden geselecteerd, zodra het systeem vereist dat deze worden ingevoerd.	M
S1	Het systeem zal middels een functie de cliëntdata moeten kunnen exporteren naar een locatie buiten het systeem.	S
S2	De gebruiker en/of beheerder kan alleen toegang krijgen tot het systeem middels een gebruikersnaam en wachtwoord	S
S3	Er moet een mogelijkheid zijn om de gegevens te rapporteren	S
C1	De LVR (Landelijke Verloskunde Registratie) vereist dat er periodiek een formulier wordt toegezonden met het aantal cliënten en extra gegevens, deze moet door het systeem kunnen worden gegenereerd.	C
C2	De instellingen binnen de applicatie moeten kunnen worden veranderd door een beheerder	C
C3	Gebruikers van het systeem moeten kunnen worden toegevoegd	C

In tegenstelling tot in de definitiefase zijn aan de systeemeisen codes toegekend. Deze codes zullen bij het evalueren van de pilot gebruikt worden om te controleren of al deze systeemeisen gedekt worden.

2.1.2. Niet-functionele systeemeisen

Code	**Systeemeis**
NF1	De grafische user interface moet overzichtelijk zijn
NF2	Er mogen niet meerdere gebruikers tegelijkertijd met het systeem werken
NF3	Het systeem moet werken op een werkstation met een windows operating systeem
NF4	Het systeem zal dag en nacht moeten kunnen draaien.
NF5	Registratie van eerste gesprek met client mag maximaal een half uur duren
NF6	Foutmeldingen moeten concreet en begrijpelijk zijn

De niet-functionele systeemeisen zijn hierboven opgenomen, met een bijbehorende code. In deze paragraaf zijn alle niet-functionele systeemeisen vanuit de definitiefase overgenomen, dit wil echter niet zeggen dat deze ook daadwerkelijk van toepassing zijn. Dit zal blijken uit de evaluatie van de pilot moeten blijken.

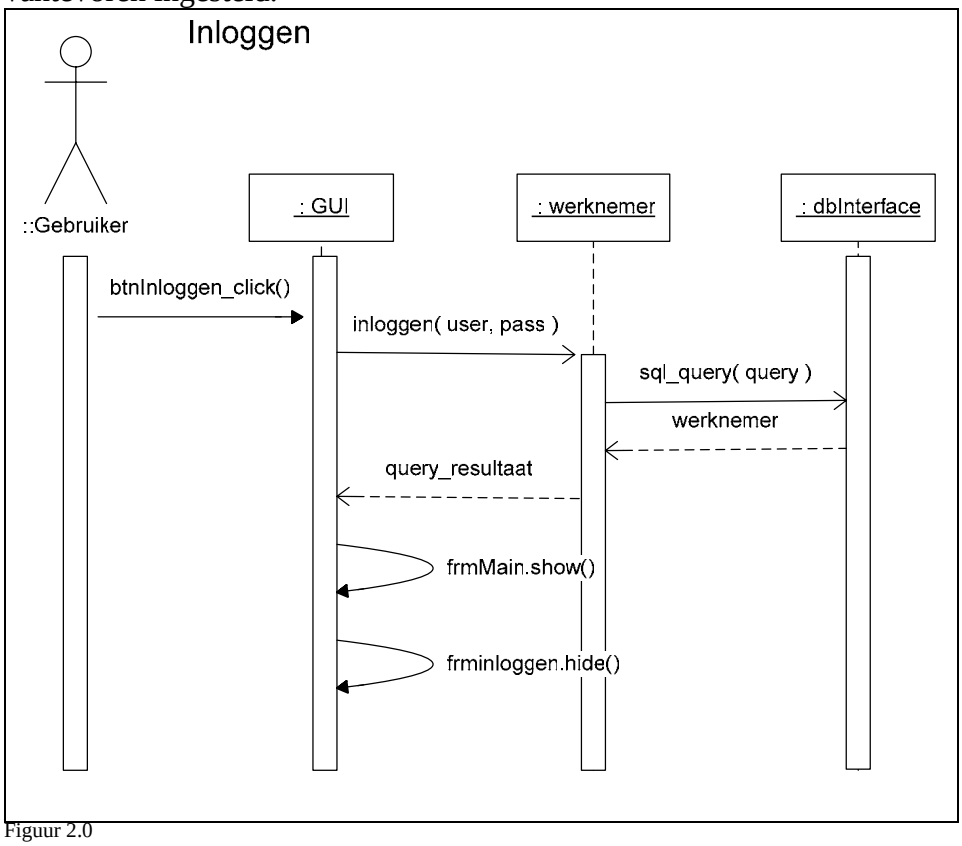
2.2. Contactmoment client

Dit subsysteem, eerste contactmoment client (afgekort tot Contactmoment client), is het eerste subsysteem in een reeks van vier. De use-cases uit de definitiefase worden hier per subparagraaf in functioneel detail ontworpen.

Het functionele ontwerp zal bestaan uit een uitbreiding van de use-case beschrijvingen, waarin vooral de reactie van het systeem wordt uitgebreid. Hiernaast wordt ook een sequence-diagram opgesteld waarin de gebruikte objecten herkenbaar worden en waar nodig wordt een tekstuele toelichting gegeven.
De sequence-diagrammen zijn opgesteld in het geval van een succesvolle uitvoering. De eventuele uitzonderingen die kunnen optreden worden in het hoofdstuk 'Ontwerp software bouweenheden' besproken.

2.2.1. Inloggen

De eerste use-case inloggen, maakt het mogelijk om de applicatie te kunnen gebruiken. Voordat het systeem gebruikt kan gaan worden, moet de gebruiker middels een geldige combinatie van gebruikersnaam en wachtwoord worden gecontroleerd.
De gebruikersnaam bestaat uit de naam van de medewerker en het wachtwoord is vantevoren ingesteld.



Figuur 2.0

In figuur 2.0 is het sequencediagram te zien voor de functie inloggen. Hierin wordt duidelijk dat wanneer de gebruiker op de button 'inloggen' klikt, het proces in werking wordt gesteld. De naam van de medewerker en het bijbehorende wachtwoord worden middels de functie inloggen van de klasse medewerker omgezet naar een geldige SQL query.

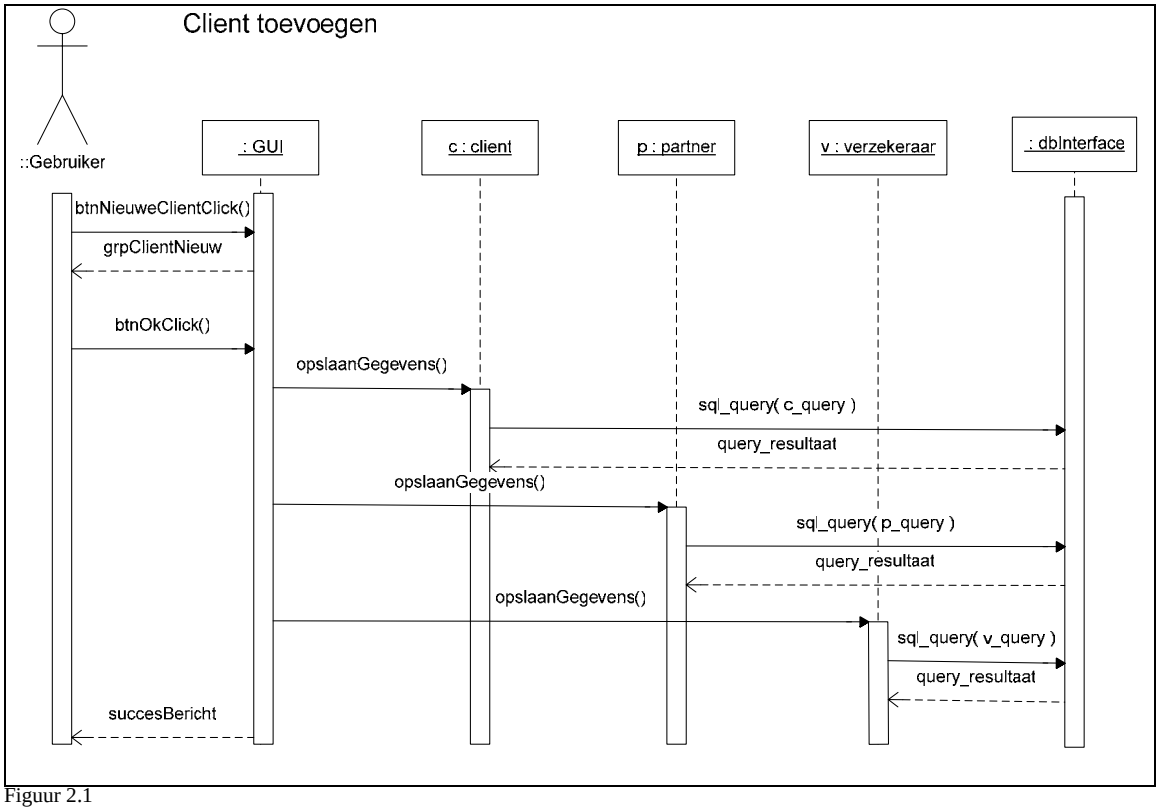
Ter illustratie en toelichting is in dit sequencediagram de database-interface opgenomen. Met behulp van de opgestelde query, kan de functie ‘sql_query’ uitgevoerd worden, die een ‘query_resultaat’ tot gevolg heeft.

Dit query_resultaat wordt vervolgens door de GUI geïnterpreteerd en zal op basis daarvan het hoofdscherm van de applicatie tonen en het inlogscherm verbergen.

De inhoud van de functies en procedures waaronder het omzetten naar een mysql_query zal bij het ontwerp van de software-bouweenheden naar voren komen.

2.2.2. Cliënt toevoegen

Het toevoegen van de cliënt zoals al in de definitiefase behandeld wordt hieronder toegelicht met een sequencediagram.



Figuur 2.1

Om de cliënt succesvol toe te kunnen voegen, moet uiteraard eerst worden ingelogd. Dit is vanzelfsprekend, aangezien dit in de use-case beschrijvingen al tekstueel is toegelicht. De gebruiker van het systeem zal uiteraard alle gegevens die benodigd zijn voor een nieuwe cliënt, naar eigen inzicht moeten invullen. Dit eigen inzicht wordt uiteraard gestuurd door de velden die ingevuld moeten worden, echter zal de applicatie geen foutmelding geven als een veld niet ingevuld is.

De use-case beschrijving die in de definitiefase is ontwikkeld is hieronder ter informatie opgenomen.

Use-case	Client toevoegen
Preconditie	-
Postconditie	De cliënt is ingevoerd en de bijbehorende sub-cases zijn ook uitgevoerd.
Beschrijving	1. De gebruiker dient in te loggen 2. De gebruiker vult de personalia in 3. De gebruiker werkt het contactmoment bij 4. De gebruiker registreert de voornemens 5. De gebruiker registreert de toxische factoren 6. De gebruiker stelt de 'a terme' in, (bevallingstermijn) 7. De gebruiker vult de anamnese in

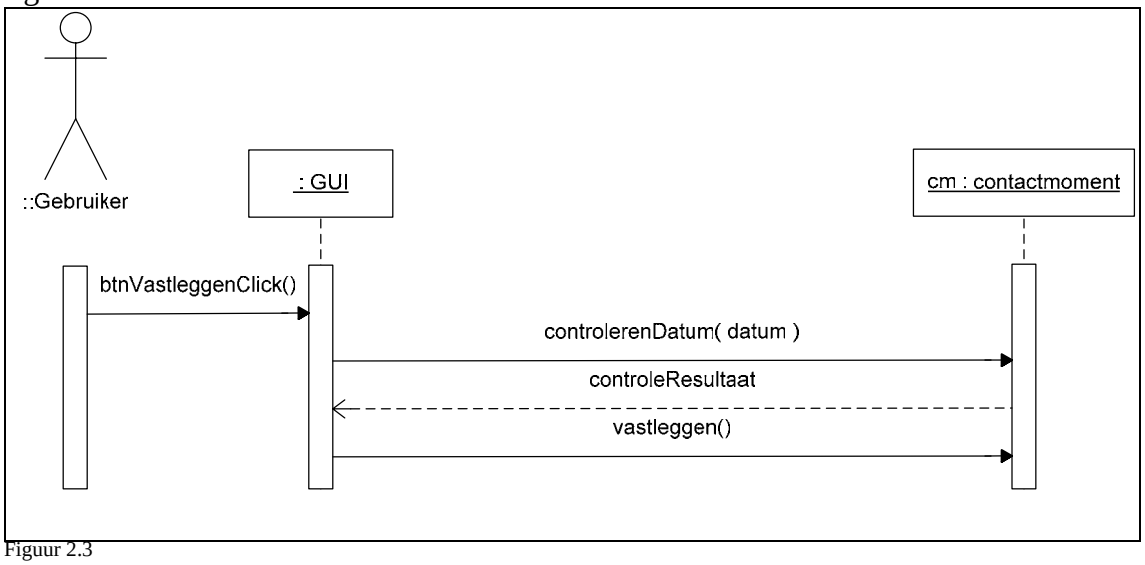
Figuur 2.2

In figuur 2.2 is te zien dat de sub use-cases ook als actie zijn opgenomen in de use-case beschrijving. Deze use-cases worden in de volgende paragrafen behandeld. Wat vooral van belang is bij deze use-case is de functie opslaanGegevens(), die nog niet eerder is behandeld.

Deze functie heeft als doel de benodigde gegevens op te slaan in een instantie van de klasse. De instantie maakt een sql_query van de actie en geeft deze tenslotte aan de instantie van de klasse database interface (dbInterface). Het resultaat van de functie sql_query is zoals in de vorige pilot is behandeld altijd van het type TDatasource. Het resultaat van de functie opslaanGegevens() is altijd van het type Boolean (True of False).

2.2.3. Contactmoment bijwerken

Deze sub use-case wordt uitgevoerd nadat de personalia is vastgelegd, zoals te zien is in figuur 2.3.



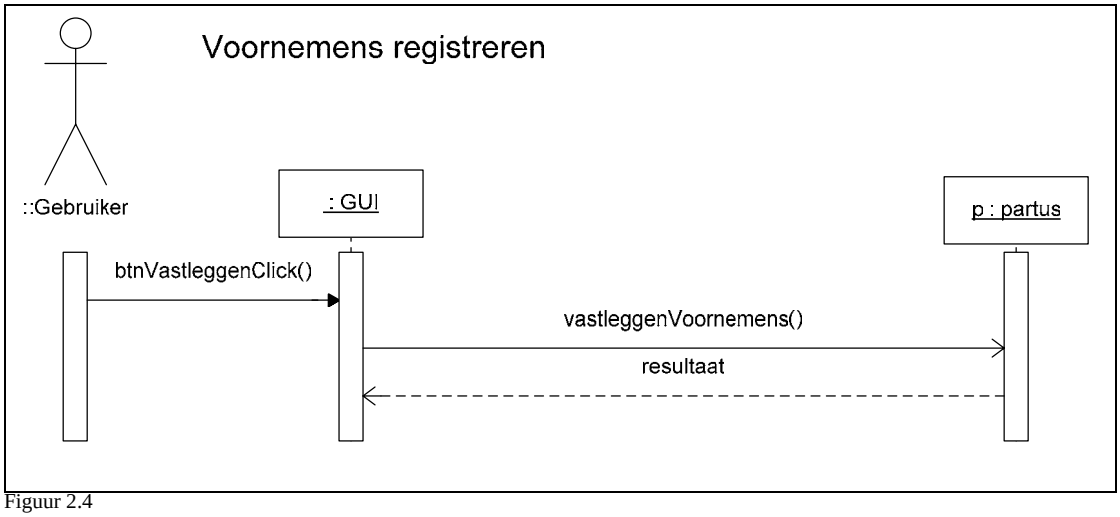
Figuur 2.3

In figuur 2.3 is te zien dat zodra de button vastleggen wordt aangeklikt het proces in werking wordt gesteld. De datum van het contactmoment dient te worden gecontroleerd, zodat deze niet in verleden ligt. De datum van het contactmoment is vrij cruciaal, op basis hiervan worden vervolgafspraken gemaakt.

De klasse dbInterface is vanaf nu bij ieder sequencediagram weggelaten, aangezien dit geen toegevoegde waarde heeft op het sequencediagram.

2.2.4. Voornemens registreren

Het registreren van de voornemens houdt in dat de wijze van kraamzorg, plaats van bevalling, de voeding en de voorbereiding op de zwangerschap wordt vastgelegd.



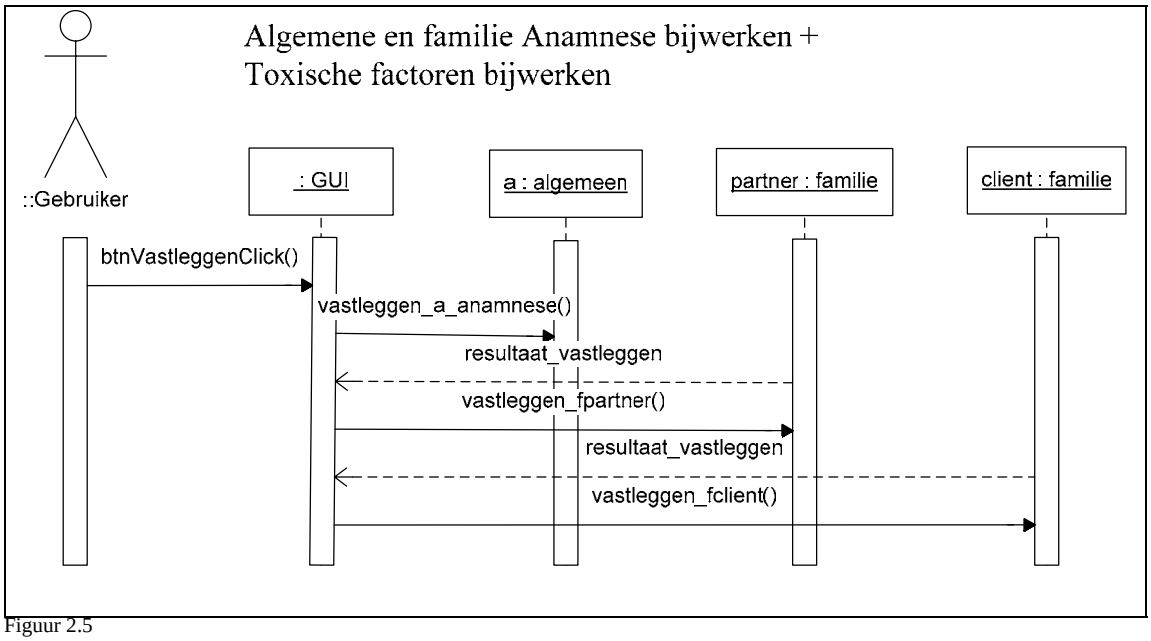
Figuur 2.4

Bij deze klasse wordt wederom een variant op de functie vastleggen() gebruikt, namelijk vastleggenVoornemens(). De naam van de functie is hier veranderd, omdat het niet een klasse voornemens betreft, maar de klasse partus. Deze klasse bevat naast vele andere attributen ook die van de voornemens.

De functie heeft echter geen argumenten, aangezien in dit stadium nog niet is vastgelegd hoe de variabelen aan de functie worden meegegeven. Dit zal naar alle waarschijnlijkheid een array worden van een onbekende grootte, zodat in het geval van null-waarden de functie geldig blijft.

2.2.5. Toxische factoren bijwerken

De toxische factoren bijwerken is een use-case die onder andere het alcohol-gebruik en de mate van roken registreert. Aangezien het gebruik hiervan hoort tot het geschiedenis (anamnese) van de client, is deze ook verwerkt in het sequence-diagram.



Figuur 2.5

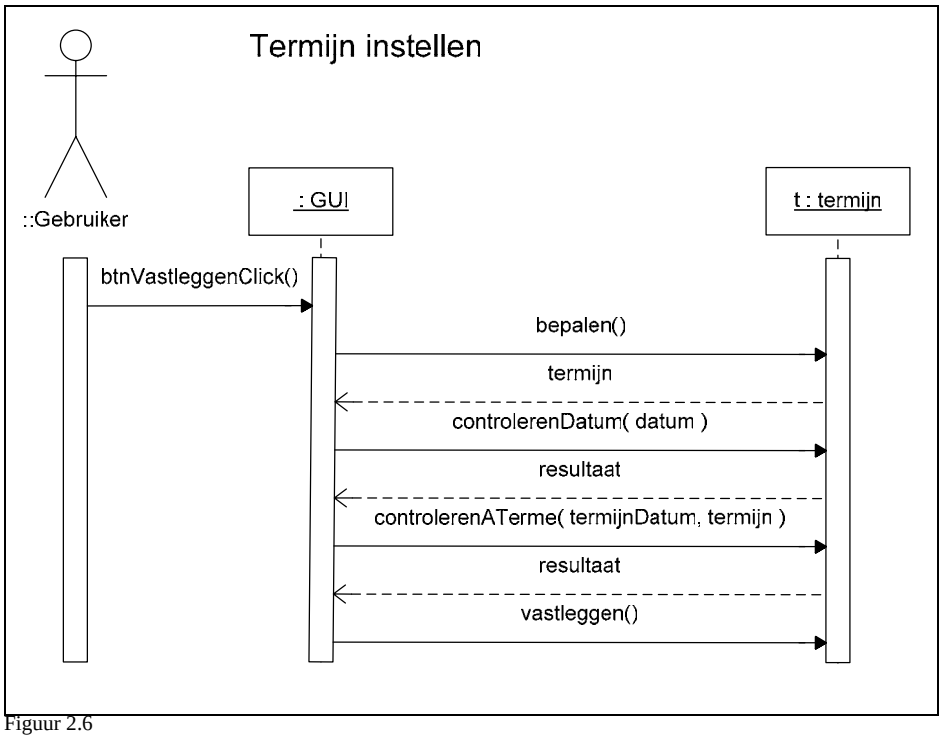
De anamnese houdt in dat naast eventuele verslavingen ook eventuele ziektes moeten worden geregistreerd. Dit in verband met overdracht aan het kind en complicaties die kunnen optreden bij de bevalling. In sommige gevallen wordt hier gekozen voor het doorsturen naar de gynaecoloog om complicaties te voorkomen die niet door een verloskundige mogen worden behandeld.

Naast het klikken op de knop vastleggen wordt de algemene anamnese (drugs en ziektes van de client) vastgelegd. Vervolgens wordt de familie anamnese vastgelegd. Deze zijn apart voor de partner en de client. Het vastleggen van de familie anamnese houdt in dat erfelijke ziektes kunnen worden geconstateerd die worden doorgegeven aan het kind.

2.2.6. Termijn instellen

Het instellen van een termijn is een procedure die op basis van de laatste menstruatie berekent na hoeveel weken een zwangere vrouw kan bevallen. Uiteraard kan dit afwijken, maar in de meeste gevallen bevalt de vrouw vanaf 38 tot 42 weken vanaf de laatste menstruatie. De afwijking kan ook liggen aan dat de vrouw niet meer weet wanneer de laatste menstruatie was, of dat cyclus van de vrouw heel onregelmatig is. In onderstaand sequencediagram wordt het menstruatietermijn berekend, de datum gecontroleerd en vervolgens de ‘A terme’ datum gecontroleerd.

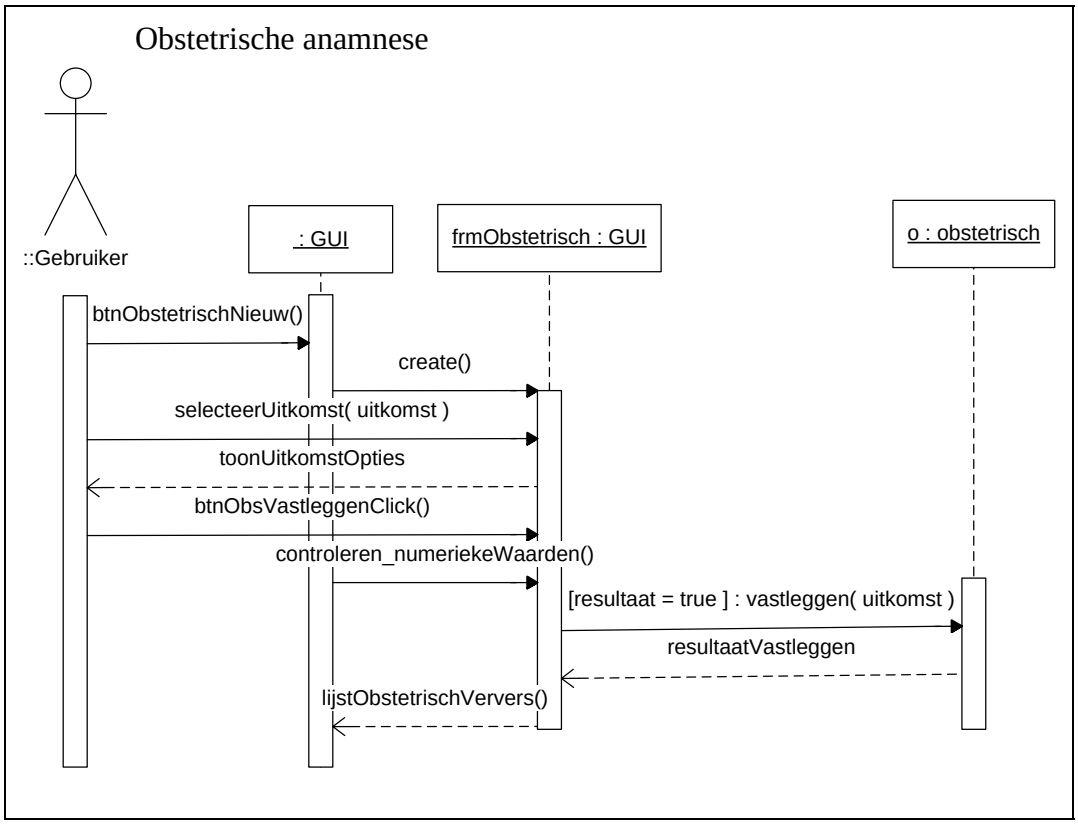
A terme is de datum dat de vrouw uitgerekend is. A terme is vertaald “op dat moment”.



Figuur 2.6

2.2.7. Anamnese bijwerken

Naast de familie en algemene anamnese is e rook nog de obstetrische anamnese. Dit is een combinatie van de familie en algemene anamnese voor voorgaande zwangerschappen. Indien de client geen eerdere zwangerschappen heeft doorlopen (abortus of volledige zwangerschap) is de obstetrische anamnese niet van toepassing.



Figuur 2.7

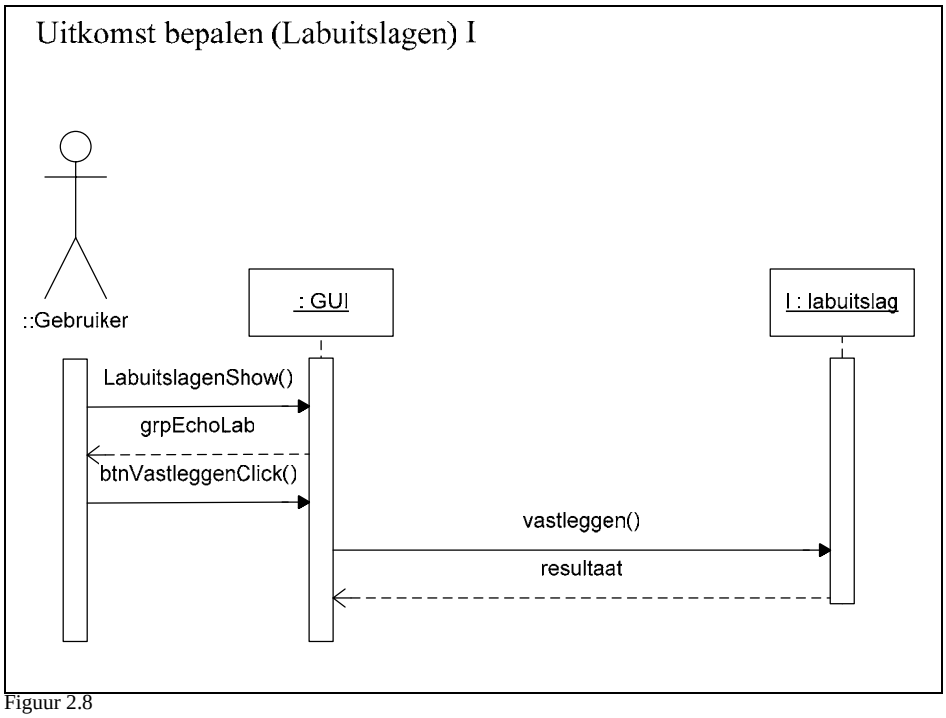
Voor de obstetrische anamnese kan worden ingevuld wordt een nieuw venster geopend door de GUI na het aanklikken van de knop ‘ObstetrischNieuw’. De uitkomst van een eerdere zwangerschap is bijvoorbeeld een abortus of een doorgaande zwangerschap (volledig, inclusief partus).

Na het selecteren van de uitkomst verschijnen er extra opties, die extra toelichting geven op het verloop en de uitkomst van de zwangerschap.

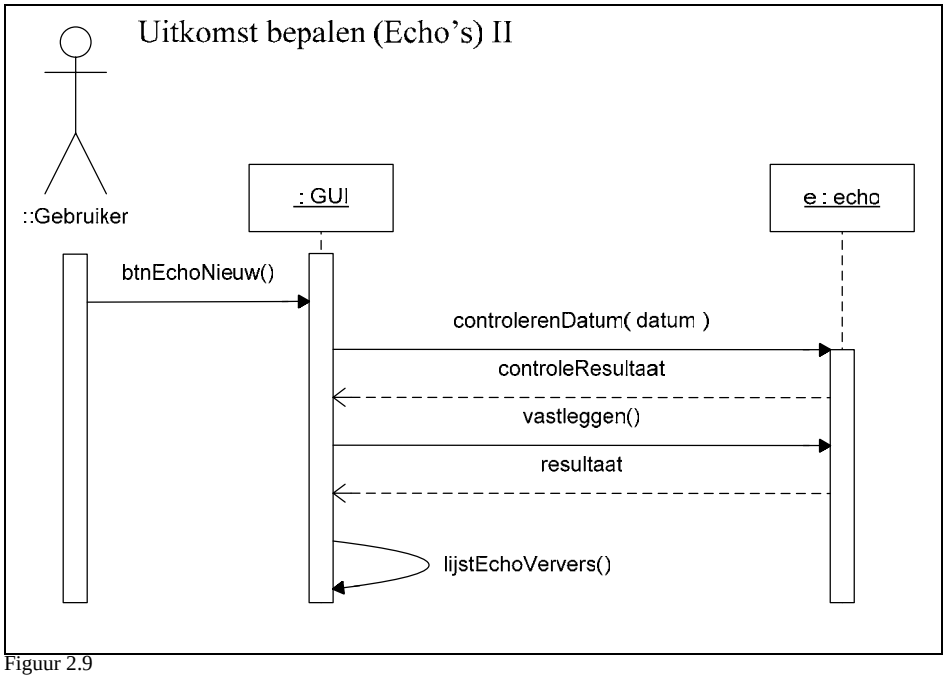
2.2.8. Uitkomst bepalen

De use-case uitkomst bepalen is procedure die verloskundigen gebruiken om te bepalen hoe het verloop en wat het resultaat van de zwangerschap zal zijn. Het bepalen van de uitkomst wordt gedaan op basis van de kennis van de verloskundige in combinatie met de gemaakte echo’s en de labuitslagen van de client. De uitkomst van een echo kan uiteraard ook heel negatief zijn, bijvoorbeeld een ‘spontane abortus’.

In figuur 2.8 is het diagram van de labuitslagen te zien is en in figuur 2.9 het diagram voor het toevoegen van echo’s.



Figuur 2.8

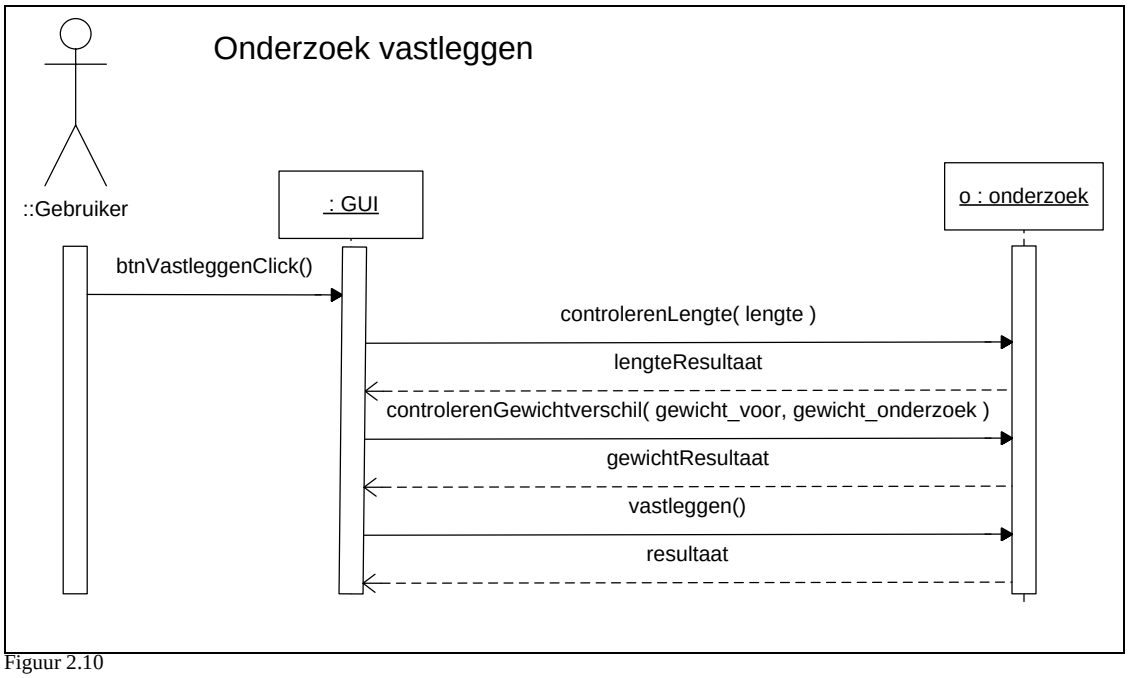


Figuur 2.9

2.2.9. Onderzoek vastleggen

Vastleggen van een onderzoek houdt in dat een drietal cliëntgegevens worden geregistreerd:

- De lichaamslengte
- Het gewicht voor de zwangerschap
- Het gewicht bij het eerste contactmoment



Figuur 2.10

Wat in figuur 2.10 te zien is, is dat de lengte en het gewicht moeten worden gecontroleerd, zodat er geen vreemde getallen in de database terechtkomen. Met vreemde getallen wordt bedoeld dat een komma in het gewicht op lengte verkeerd geplaatst wordt en dat een eenduidige eenheid in gewicht (kilogrammen) en lengte (meters) wordt gebruikt.

2.3. Zwangerschap

Het subsysteem zwangerschap, is een subsysteem dat bedoeld is om de gegevens van de uitkomst te registreren. Logischerwijs bevat dit subsysteem alleen use-cases die betrekking hebben op een partus / abortus en het resultaat van de zwangerschap, een baby.

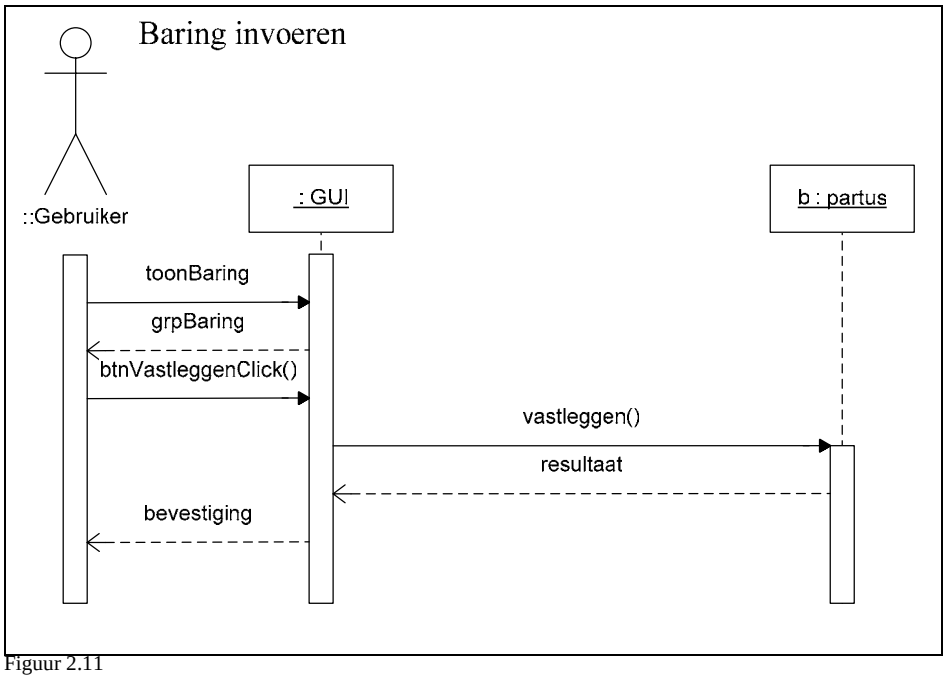
Net als in het vorige subsysteem zullen de use-cases per subparagraaf worden toegelicht. Uitzondering is dat niet bij iedere use-case een sequence-diagram is opgenomen. Een sequencediagram is alleen nuttig als deze een toegevoegde waarde op de content van de use-case heeft. Dit is in sommige use-cases niet het geval, aangezien het principe voor bijna iedere use-case in dit subsysteem hetzelfde is.

In deze paragraaf is voor de use-case ‘Baring invoeren’ een diagram opgenomen en een voor ‘Zoeken’. De andere use-cases zijn tekstueel toegelicht, op basis van het diagram van ‘Baring invoeren’.

2.3.1. Baring invoeren

De begrippen baring, partus en zwangerschap worden veelal door elkaar heen gebruikt. Net als de verloskundigen dat doen, is door de aanpassing aan de omgeving bij mij ook het geval. Om verwarring te verkomen zal hier kort worden toegelicht wat de verschillen en overeenkomsten zijn.

De woorden partus en baring hebben dezelfde betekenis, het enige verschil is dat partus Latijns is voor een baring. In geval van baring wordt een succesvolle zwangerschap bedoeld. Een zwangerschap hoeft niet perse een succesvolle uitkomst te hebben. Een vrouw kan zwanger zijn, zonder een partus na 40 weken. Dit wordt dan een zwangerschap met negatieve uitkomst.



Figuur 2.11

Bij het invoeren van een baring gaat het om het invoeren van de gegevens omtrent de partus. Aangezien dat in een sequencediagram niet goed weer te geven is, is het proces weergegeven en het object dat de gegevens zal bezitten.

2.3.2. Gegevens kraambed invoeren

Het invoeren van de kraambedgegevens houdt in dat er geregistreerd wordt hoe de client (moeder) zich gedraagt na de bevalling. Met gedrag wordt bedoeld of er medicijnen benodigd zijn om de gezondheid van de client te garanderen (lactatieremmers bijvoorbeeld)

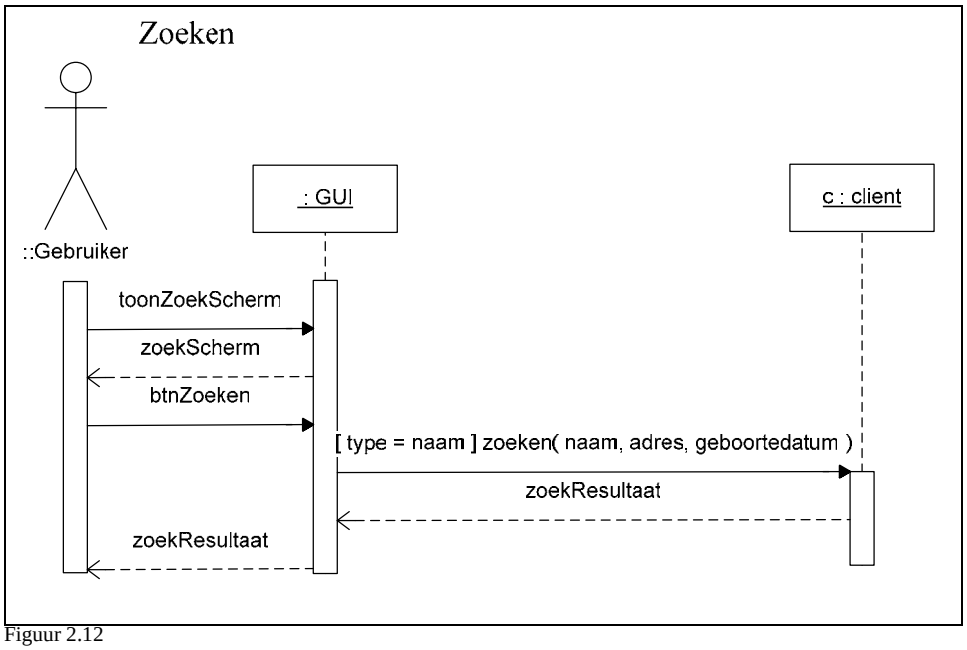
2.3.3. Babygegevens invoeren

De gegevens van de baby zijn de cruciale gegevens zoals de conditie van de baby na de bevalling en wat voor soort voeding deze krijgt. De conditie wordt na een week nogmaals gecontroleerd en als nacontrole ingevoerd.

2.3.4. Zoeken

De use-case zoeken is bedoeld om te kunnen zoeken naar de cliënten die in het systeem zijn opgenomen. Het zoeken van deze cliënten is bedoeld om gegevens te kunnen bijwerken en er kan op unieke gegevens zoals het zwangerschapsnummer, de naam, het adres en de geboortedatum van de client.

Hieronder is het sequencediagram opgenomen die het zoeken naar een cliënt op het naam adres en woonplaats weergeeft.



Figuur 2.12

De functie zoeken() van een instantie van de klasse client wordt aangeroepen als het type ‘naam’ is. Het zoekresultaat wordt teruggegeven aan de GUI en deze voert daarop operaties uit en toont het juiste resultaat aan de gebruiker.

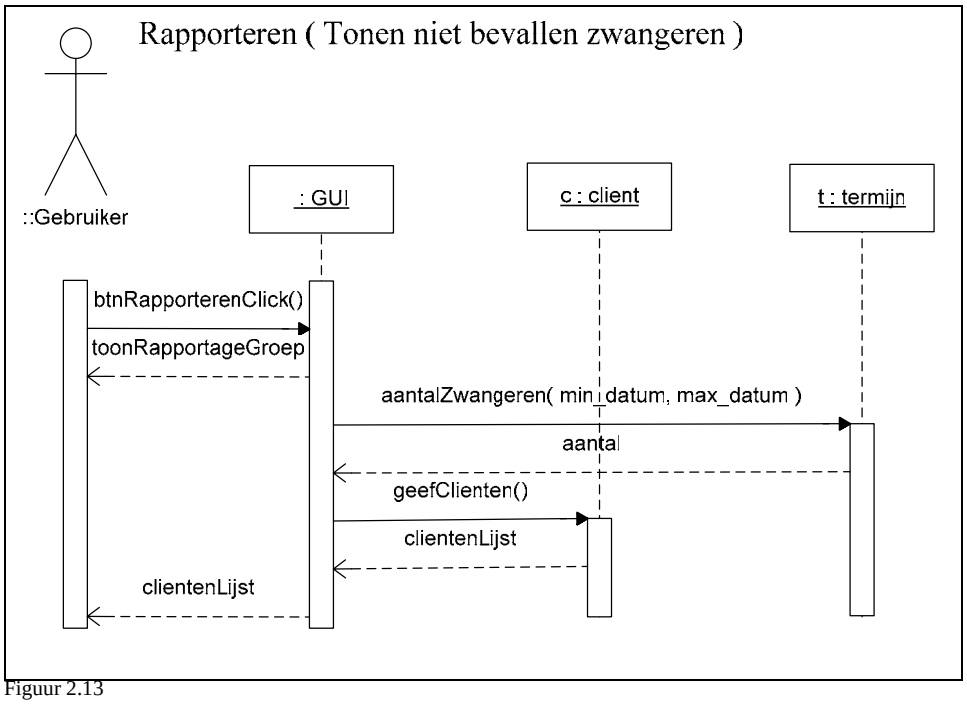
2.4. Administratie

Het subsysteem administratie bezit de use-cases ‘Rapporteren’ en ‘Uitvoeren LVR-Controle’. De laatste use-case is bedoeld voor de financiële taken die de praktijk moet uitvoeren. Deze financiële taken vallen buiten het systeem en worden ook niet hier behandeld.

De eerste use-case wordt toegelicht door middel van een sequence-diagram. De tweede use-case is komen te vervallen zoals zal worden toegelicht in de desbetreffende subparagraaf.

2.4.1. Rapporteren

Rapporteren is een ruim gekozen naam voor een kleine functie. Deze functie rapporteert alleen de vrouwen die deze periode (38 weken >= huidige datum <= 42 weken) zouden moeten gaan bevallen. Hiermee kan tijdig worden ingegrepen indien noodzakelijk.



Figuur 2.13

In figuur 2.13 is te zien dat de functie aantalZwangeren een resultaat teruggeeft aan de GUI, deze haalt vervolgens cliënten op en retourneert deze aan de gebruiker.

2.4.2. Uitvoeren LVR-Controle

De use-case uitvoeren LVR-Controle, zou in het toekomstige systeem een knop moeten zijn die de database controleert op gegevens die voor de LVR (Landelijke Verloskunde Registratie) vereist zijn.

De opdrachtgever vond deze functie achteraf helemaal niet nodig, omdat de gegevens die de LVR nodig heeft op een zwangerschapskaart moeten worden gezet. Een kopie van die zwangerschapskaart wordt vervolgens naar de LVR toegestuurd. Waarbij een knop die deze controle uitvoert overbodig maakt.

3.

Pilotontwikkelplan

Het pilotontwikkelplan beschrijft hoe de pilot gaat worden gebouwd. Per pilotdeel worden één of meer bouweenheden gedefinieerd. Bouweenheden zijn componenten van de pilot die door hun aard en reikwijdte achtereenvolgens of parallel kunnen worden ontwikkeld.
In paragraaf 3.1 worden er pilotdelen opgesteld en onderverdeeld in bouweenheden en wordt aangegeven in welke volgorde de bouweenheden worden ontwikkeld. Paragraaf 3.2 beschrijft de ontwikkelaanpak.

3.1. Pilotdelen en bouweenheden

De methode IAD schrijft voor om allereerst de pilotdelen te definiëren. Dit zijn de delen die als componenten kunnen worden beschouwd en afzonderlijk kunnen worden geïmplementeerd. Binnen die pilotdelen worden de bouweenheden vastgesteld, dit stappen die bepaalde onderdelen van de pilot bevatten die bij de bouw samengevoegd het deel van de pilot vormen.

Om vast te kunnen stellen wat precies als pilotdeel wordt gedefinieerd wordt allereerst opgesomd wat er allemaal in deze pilot als resultaat moet worden opgeleverd.

In het vorige hoofdstuk is te zien dat de eerste drie gedefinieerde subsystemen op functionele inhoud zijn toegelicht. Dit is echter niet het enige wat deze pilot moet gaan opleveren. De volgende producten zullen als resultaat worden opgeleverd:

- Klassen en bijbehorende operaties
- Operationele subsystemen één tot en met drie.
- Gebruikershandleiding per subsysteem

Naast deze drie grootschalige producten moet ook de functionele systeemeisen worden gedekt. Hoewel deze drie producten ook als drie pilotdelen zoude kunnen worden gedefinieerd, is er gekozen voor een iets andere aanpak. Het eerste en tweede product worden wel degelijk een pilotdeel. Echter zal het derde product worden gecombineerd met het ontwikkelen van de subsystemen. Naast die twee wordt een extra pilotdeel gedefinieerd die een aantal afspraken vast moet leggen. Dit leidt tot de volgende drie pilotdelen:

1. Opzetten basisstructuur en omgeving
2. Ontwikkelen klassen en bijbehorende operaties
3. Ontwikkelen subsystemen

In de aankomende subparagrafen worden de bouweenheden per pilotdeel gedefinieerd.

3.1.1. Opzetten basisstructuur en omgeving

Dit pilotdeel is opgezet om structuur in de applicatie aan te brengen. Dit is vooral bedoeld om de toekomstige applicatie overdraagbaar te maken, zoals deze is gedefinieerd in de regels van software engineering.

Bovendien is het van belang om structuur aan te brengen, aangezien een applicatie van deze omvang snel kan leiden tot een verzameling functies zonder enige samenhang. Op basis hiervan zijn de volgende bouweenheden gedefinieerd

1. Definiëren basisstructuur
2. Definiëren constanten en globale variabelen
3. Definiëren standaard voor toelichting in de code

3.1.2. Ontwikkelen klassen en bijbehorende operaties

De klassen die zijn gedefinieerd worden omgezet naar object in Delphi. Als bouweenheden zijn het ontwerp van de objecten en de bouw van de functies apart gedefinieerd. Wat leidt tot de volgende twee bouweenheden.

4. Omzetten klassen naar objecten
5. Ontwikkelen functies en procedures

3.1.3. Ontwikkelen subsystemen

Dit pilotdeel bestaat uit de bouweenheden die subsystemen implementeert. Naast de drie bouweenheden is een extra bouweenheid opgenomen waarin de gebruikershandleiding wordt ontwikkeld en opgesteld.

6. Ontwikkelen functies Contactmoment cliënt
7. Ontwikkelen functies subsysteem Zwangerschap
8. Ontwikkelen functies subsysteem Administratie
9. Opstellen gebruikershandleiding subsystemen

3.2. Ontwikkelaanpak

De ontwikkelaanpak is de aanpak die aangeeft hoe de pilotdelen en bouweenheden ontwikkeld gaan worden. De hiervoor gekozen aanpak is de iteratiestrategie. Deze aanpak zorgt ervoor dat de ontwikkeling van de bouweenheden iteratief verloopt.

Om te testen of de bouweenheden voldoen aan de gestelde eisen, zijn per bouweenheid een aantal testspecificaties opgesteld. Door het uitvoeren van de tests kan per bouweenheid worden bepaald of het gewenste resultaat wordt geleverd. Het kan zich voordoen dat uit de testresultaten blijkt dat de bouweenheid niet voldoet aan de gestelde eisen. Op basis van die resultaten zal een oplossing moeten worden bedacht. Deze oplossing kan eventueel betekenen dat de ontwikkeling van de bouweenheid nogmaals moet worden doorlopen.

4. Ontwerp software-bouweenheden

Het doel van dit hoofdstuk is om de technische structuur van de bouweenheden in het vorige hoofdstuk uit te breiden Naast de technische structuur worden ook de testspecificaties per bouweenheid gedefinieerd in dit hoofdstuk. Op basis de structuur en de testspecificaties moet het mogelijk om de bouweenheden te implementeren en te testen.

De opbouw van dit hoofdstuk begint met in de eerste paragraaf het ontwerp van de bouweenheden. In de daaropvolgende paragraaf 4.2 worden de testspecificaties per bouweenheid gedefinieerd.

4.1. Ontwerp bouweenheden

4.1.1. Bouweenheid 1: Definiëren basisstructuur

Allereerst wordt de basisstructuur van de ontwikkelomgeving gedefinieerd. Het resultaat van de pilot is een opzichzelfstaande executabel programma, waarin deze structuur zit verwerkt. De definitie is bedoeld om het overzicht te bewaren bij het ontwikkelen en debuggen van fouten.

Om die reden is ervoor gekozen om verschillende units te bouwen. De hoeveelheid units dat zal worden onderscheiden hangt af van de onafhankelijkheid van de unit. Een unit kan gezien worden als een apart component en om het overzicht te bewaren, is de indeling gebaseerd op het doel van de unit:

- Globale functies (global_functions.pas)
- Objecten met bijbehorende operaties (classes.pas)
- Het bijhouden van een logbestand (logging.pas)
- De database interface (database.pas)
- MD5 encryptie (md5.pas)

De inhoud van units is hierboven weergegeven en het merendeel is reeds besproken. De units global_functions en MD5 zijn in dit stadium nieuw toegevoegd. In de eerste unit worden de functies ondergebracht die zullen worden hergebruikt. Deze functies/procedures zijn opgenomen in het onderstaande overzicht.

```
function getSelectedComboItem( cmbBox: TComboBox; value: integer ): string;
function getSelectedRadioItem( rdoBox: TRadioGroup; value: integer ): string;
procedure groupActivate( groupBox: TGroupBox );
procedure groupDeactivate( groupBox: TGroupBox );
procedure deactivateAllGroups( grp1, grp2, grp3, grp4, grp5, grp6, grp7, grp8,
grp9: TGroupBox );
function WindowExists(AppWindowName, AppClassName: string): Boolean;
function isStrEmpty( const testString: string ): Boolean;
function strToArr( const str: string; var typeArray : array of char ):
boolean;
function arrToStr( const typeArray: array of const ): string;
```

Hierboven zijn de functies opgenomen die in het kader van de unit 'Globale functies' worden ontwikkeld. De functies zullen nu kort worden toegelicht door middel van pre- en postcondities

getSelectedComboItem

Deze functie geeft de inhoud terug van het item dat geselecteerd is. Waarbij het argument 'value' aangeeft wat deze inhoud representeert. De text van het item kan worden geretourneerd of de index van het geselecteerde item kan worden teruggegeven.

Functienaam	getSelectedComboItem
Resultaat Type	String
Argumenten	cmbBox: TComboBox value: integer;
Preconditie	cmbBox not null value = [0 | 1]
Postconditie	Het geselecteerde item is teruggegeven als resultaat zoals in het argument value is aangegeven
Uitzonderingen	1. Er is geen item geselecteerd in de gegeven cmbBox. 2. Er zijn geen items aanwezig in de gegeven cmbBox.
Oplossing uitzonderingen	Het resultaat = -1

getSelectedRadioItem

Deze functie geeft de inhoud terug van het radioitem dat geselecteerd is. Hierbij geldt hetzelfde als voor bij de vorige functie. Het argument 'value' geeft aan welke inhoud wordt teruggegeven.

Functienaam	getSelectedRadioItem
Resultaat Type	String
Argumenten	rdoBox: TRadioGroup value: integer;
Preconditie	rdoBox not null value = [0 | 1]
Postconditie	Het geselecteerde item is teruggegeven als resultaat zoals in het argument value is aangegeven
Uitzonderingen	1. Er is geen item geselecteerd in de gegeven rdoBox.
Oplossing uitzonderingen	Het resultaat = -1

groupActivate, groupDeactivate, deactivateAllGroups

Aangezien de applicatie bestaat uit een verzameling van groepen met knoppen, labels en tekstvelden, moeten deze groepen geactiveerd en gedeactiveerd worden. Het activeren en deactiveren van de groepen wordt gerealiseerd door het te verbergen.

Functienaam	groupActivate
Resultaat Type	-
Argumenten	groupBox: TGroupBox;
Preconditie	groupBox not null
Postconditie	De gegeven groupBox is visible
Uitzonderingen	-
Oplossing uitzonderingen	-

Functienaam	groupDeactivate
Resultaat Type	-
Argumenten	groupBox: TGroupBox;
Preconditie	groupBox not null
Postconditie	De gegeven groupBox is invisible
Uitzonderingen	-
Oplossing uitzonderingen	-

Functienaam	deActivateAllGroups
Resultaat Type	-
Argumenten	Alle groepen van het type GroupBox
Preconditie	Alle groepen zijn uniek
Postconditie	Alle groepen zijn invisible
Uitzonderingen	-
Oplossing uitzonderingen	-

WindowExists

Deze functie gaat na of de applicatie al is gestart op het huidige werkstation. Dit wordt gedaan door de klasse en applicatienaam als argumenten te vergelijken met de proceslijst van windows. Het resultaat is van het type boolean, true als de applicatie gevonden is, false indien dit niet zo is.

Functienaam	WindowExists
Resultaat Type	Boolean
Argumenten	AppWindowName: string AppClassName: string
Preconditie	-
Postconditie	Het programma is gevonden in de proceslijst van Windows.
Uitzonderingen	Het programma is niet gevonden, aan de hand van de gegeven argumenten
Oplossing uitzonderingen	Result = false

isStrEmpty

Deze functie controleert of de string die gegeven is als argument leeg is. Delphi beschouwd strings als een array van char waarbij het eindkarakter een \0 is. Om te controleren of een string leeg is, wordt gecontroleerd of de lengte van de string gelijk is aan 0. Indien dit niet het geval is wordt gecontroleerd of het adres van de string 'nil' is.

Functienaam	isStrEmpty
Resultaat Type	Boolean
Argumenten	testString: string
Preconditie	-
Postconditie	De teststring is leeg
Uitzonderingen	De teststring is niet leeg
Oplossing uitzonderingen	Result = false

strToArr

Deze functie converteert een string naar een array. Dit klinkt overbodig aangezien een string al een array of char is, maar omdat er met typen wordt gewerkt is zo'n functie noodzakelijk. Een string is namelijk een array of char zonder bepaalde grootte. Aangezien de database een maximale grootte definieert voor sommige velden, moet dat bij de attributen van de objecten ook worden doorgevoerd.

Een naam is bijvoorbeeld van het type TNames. Dat een array van char is met een maximale lengte, namelijk 'C_SIZE_NAMES'. Om een variabele van het type string met een variabele van dit type te vergelijken resulteert in een foutmelding. Het zijn namelijk niet dezelfde typen.

Deze functie maakt om die reden van een string een array met een bepaalde gegeven lengte, wat kopiëren en vergelijken mogelijk wordt.

Functienaam	strToArr
Resultaat Type	Boolean
Argumenten	Str: string **var** typeArray: array of char
Preconditie	De string is not null Het typeArray is niet statisch
Postconditie	De string is omgezet naar de gegeven array en het resultaat is true
Uitzonderingen	De string is niet omgezet.
Oplossing uitzonderingen	Het resultaat is false

arrToStr

Deze functie doet het omgekeerde van de vorige functie. Deze converteert een statisch array van het type array of const naar een string.

Functienaam	arrToStr
Resultaat Type	String
Argumenten	constArray : array of const
Preconditie	-
Postconditie	De array is omgezet naar een string met een \0 karakter. De string wordt teruggeven als resultaat
Uitzonderingen	De string is niet omgezet.
Oplossing uitzonderingen	Het resultaat is nil

De laatste unit is een unit die de MD5-functie implementeert. Delphi kent zelf geen MD5-encryptie en deze moet dus geheel zelf worden ontwikkeld. De aanpak hiervoor is om Request For Comment 1321 te bestuderen en te implementeren. Deze RFC beschrijft de standaard voor het ontwikkelen van het “Message-Digest 5 Algorithm”.

De volgende procedures en functies moeten worden ontwikkeld aan de hand van de RFC 1321.

```
procedure MD5Init(var Context: MD5Context);
procedure MD5Update(var Context: MD5Context; Input: pChar; Length:
longword);
procedure MD5Final(var Context: MD5Context; var Digest: MD5Digest);

function MD5String(M: string): MD5Digest;
function MD5File(N: string): MD5Digest;
function MD5Print(D: MD5Digest): string;

function MD5Match(D1, D2: MD5Digest): Boolean;
```

Voor toelichting op de ontwikkeling van het MD5 algoritme wordt verwezen naar de RFC 1321.

4.1.2. Bouweenheid 2 Definiëren constanten en globale variabelen

Allereerst zullen de constanten en de globale variabelen gedefinieerd worden. De constanten zijn waarden die binnen de applicatie niet kunnen worden veranderd. Het zijn statische variabelen.

Een voorbeeld van zo’n constante is die van de combinatie van een ‘carriage return en line feed’, een harde return in een messageBox. Delphi kent hiervoor geen standaard en vandaar dat er een is aangemaakt.

De globale variabelen zijn de variabelen die in de gehele applicatie beschikbaar zijn. Aangezien er maar één instantie van de klasse database-interface mag zijn, moet die instantie globaal worden gedefinieerd.

Zoals al in de eerste bouweenheid naar voren is gekomen, is het noodzakelijk dat de attributen van de objecten een vast type krijgen. Deze typen moeten dezelfde lengte zijn als de velden in de tabellen.

```
const
  C_SIZE_NAMES = 150;
  C_SIZE_DATES = 10;
  C_SIZE_ZIPCODE = 6;
  C_SIZE_TEL = 11;
  C_SIZE_SMALLNAME = 50;
  C_SIZE_ZWNR = 6;

type

  TNames = array[0..C_SIZE_NAMES] of char;
  TDates = array[0..C_SIZE_DATES] of char;
  TZipCode = array[0..C_SIZE_ZIPCODE] of char;
  TTel = array[0..C_SIZE_TEL] of char;
  TSmallName = array[0..C_SIZE_SMALLNAME] of char;
  TZwnr = array[0..C_SIZE_ZWNR] of char;
```

Een voorbeeld van zo'n type definiëring voor de velden van client is hierboven weergegeven.

4.1.3. Bouweenheid 3 Standaardisering codecommentaar.

Om de applicatie overdraagbaar te maken en duidelijk leesbaar voor eventuele toekomstige programmeurs, moet gekeken worden hoe code op de juiste wijze van commentaar moet worden voorzien. Er is geen vaste regelgeving omtrent het toevoegen van commentaar op programmeercode.

Door het Universitair Dienstencentrum voor Informatica en Telematica in Leuvens zijn er een aantal standaarden voor het geven van commentaar ontwikkeld voor de taal PHP. Hoewel deze taal in vele opzichten verschilt van Delphi, is het geven van commentaar onafhankelijk te beschouwen van de programmeertalen.

Naast deze bron is ook het document van de "Orange County Delphi Users Group" beschouwd. In dit document, dat ook online beschikbaar is, staan de standaarden voor het programmeren in Delphi. Naast een aantal programmeertips staat er ook een standaard in voor het commentaar geven in programmeercode.

De standaard die in het document is terug te vinden komt overeen met een van de standaarden die staat gedefinieerd in het universitaire onderzoek. Op basis van deze twee documenten wordt de standaard die in het document van "Orange County Delphi Users Group" is terug te vinden, gehanteerd als standaard voor dit project.

De standaard die daarin naar voren komt wordt als volgt geformuleerd:

1. Unit and Function Prologs

Each unit (Pascal source file) should begin with a comment block, or "prolog" that generally describes the contents of the unit. Each class definition and each function (procedure/method) within a unit should have a separate function prolog.

4.1.4. Bouweenheid 4: Omzetten klassen naar objecten

De klassen uit het klassendiagram zijn pas bruikbaar in Delphi als deze naar zogeheten objecten zijn omgezet. De objecten bevatten dezelfde attributen als de klassen, waar de attributen van een soortgelijke type zijn als in de database. In de eerste bouweenheid zijn de standaarden hiervoor ontwikkeld en kunnen deze voor ieder attribuut gebruikt worden.

De objecten in Delphi moeten voldoen aan een bepaalde structuur. Ze moeten worden gedefinieerd in het ‘interfacegedeelte’ en er dient onderscheid gemaakt te worden tussen de publieke, privé en beschermde functies.

In deze bouweenheid worden alleen de klassen omgezet naar een object, deze omzetting zal aan de volgende voorwaarden moeten voldoen:

- De naamgeving van de objecten beginnen met een ‘T’ gevolgd door de naamgeving vastgesteld in het klassendiagram.
- De attributen zijn publiek gedefinieerd, zodat alle units de attributen kunnen wijzigen en opvragen.
- Alle functies en procedures worden opgenomen als zogeheten ‘dummy’s’. Dummyfuncties zijn functies die geen aanvullende code hebben, naast de definitie van de functie.

4.1.5. Bouweenheid 5: Ontwikkelen functies en procedures

Als alle klassen zijn omgezet naar objecten, kan het ontwikkelen van de functies en procedures van de objecten beginnen. In deze bouweenheid is een onderscheid gemaakt tussen de twee verschillende soorten functies en procedures. De functies die in het kader van de subsystemen zijn opgesteld in de vorm van een sequencediagram en de soort functies die nog niet eerder in dit document ter sprake zijn gekomen.

De laatste groep functies, bestaat uit een viertal functies die ieder object minimaal bezit. Deze worden in deze bouweenheid ontwikkeld en gerealiseerd:

- Vastleggen
- Wijzigen
- Verwijderen
- Opvragen

Deze functies verzorgen het omzetten naar een geldige query die door de interface kan worden uitgevoerd.

Functienaam	Vastleggen
Resultaat Type	Boolean
Argumenten	-
Preconditie	De verplichte attributen in het object zijn ingevuld.
Postconditie	De gegevens in het object zijn succesvol naar de database geschreven
Beschrijving	1. Er wordt gecontroleerd of de verplichte gegevens in de database ook in het object zijn ingevuld. 2. De query voor het vastleggen van het object wordt opgesteld. 3. De query wordt uitgevoerd. 4. Het resultaat van de query wordt teruggegeven als Boolean
Uitzonderingen	1a. Object gegevens zijn niet correct of onvolledig ingevuld
Oplossing uitzonderingen	1a. Indien de niet ingevulde velden niet verplicht zijn, worden deze in de database door een NULL-waarden vervangen. Indien de velden verplicht zijn (NOT NULL) wordt deze melding aan de gebruiker teruggegeven

Functienaam	Wijzigen
Resultaat Type	Boolean
Argumenten	-
Preconditie	Alle verplichte velden van het object zijn ingevuld
Postconditie	De velden in de database zijn gewijzigd
Beschrijving	1. Alle noodzakelijk velden worden gecontroleerd 2. De query voor het wijzigen van de gegevens wordt opgesteld. 3. De query wordt uitgevoerd 4. Het resultaat wordt als Boolean teruggegeven
Uitzonderingen	1a. Object gegevens zijn niet correct of onvolledig ingevuld
Oplossing uitzonderingen	1a. Indien de niet ingevulde velden niet verplicht zijn, worden deze in de database door een NULL-waarden vervangen. Indien de velden verplicht zijn (NOT NULL) wordt deze melding aan de gebruiker teruggegeven

Functienaam	Verwijderen
Resultaat Type	Boolean
Argumenten	-
Preconditie	-
Postconditie	De gegevens van het object zijn uit de database verwijderd
Beschrijving	1. Er wordt gecontroleerd of de unieke waarde in het object is ingevuld 2. De query voor het verwijderen uit de database wordt opgesteld 3. De query wordt uitgevoerd. 4. Het resultaat wordt als Boolean teruggegeven.
Uitzonderingen	1a. De unieke waarde is niet ingevuld. 3a. Een constraint staat niet toe dat een veld uit de database wordt verwijderd.
Oplossing uitzonderingen	1a. De functie kan niet worden voortgezet en ‘false’ wordt teruggegeven. 3a. Indien een veld niet mag worden verwijderd door een

	constraint, wordt dit aan de gebruiker doorgegeven.

Functienaam	Opvragen
Resultaat Type	Boolean
Argumenten	-
Preconditie	Het unieke veld dat in de database is bepaald is in het object opgenomen.
Postconditie	De gegevens uit de database zijn opgehaald en in het object vastgelegd.
Beschrijving	1. Aan de hand van het unieke nummer in het object wordt de query opgesteld om alle waarden uit de database op te halen 2. De query wordt uitgevoerd 3. De resultaten van de query worden in het object geplaatst.
Uitzonderingen	3a. Er zijn gegevens in de database aanwezig die voldoen aan het opgegeven unieke nummer.
Oplossing uitzonderingen	-

4.1.6. Bouweenheid 6: Ontwikkelen functies subsysteem Contactmoment cliënt

In deze bouweenheid worden de functies ontwikkeld van het subsysteem contactmoment cliënt. Met het ontwikkelen van de functies worden de use-cases die betrekking hebben op het subsysteem bedoeld.

Alle acties met betrekking tot de use-cases en alle betrokken operaties die in de grafische user interface zijn ontwikkeld, moeten na het implementeren van deze bouweenheid het gewenste resultaat geven.

De basis voor deze bouweenheid wordt gevormd door de use-case beschrijvingen, de sequencediagrammen en de voorgaande bouweenheden. De functies en procedures die in het sequencediagram zijn toegepast, worden hieronder in deze bouweenheid in detail beschreven.

Hoewel de genoemde procedures in deze bouweenheid veel overeenkomsten vertonen met de procedure ‘vastleggen’ gedefinieerd in bouweenheid 5, is er een groot verschil. De procedure in bouweenheid 5 implementeert het omzetten van de gegevens in het object naar een query. De overige procedures leggen de gegevens vast in het object in plaats van in de database.

Object	Werknemer
Functienaam	Inloggen
Resultaat Type	Boolean
Argumenten	Username : string Password : string
Preconditie	De username bevat een string met de voornaam, tussenvoegsel en de achternaam van de werknemer Het password is het wachtwoord van de werknemer
Postconditie	De gegevens van de werknemer zijn vastgelegd in het object werknemer
Beschrijving	1. Het argument Password wordt omgezet naar een

	MD5 encrypt wachtwoord. 2. Er wordt een query opgezet die de gegevens van de werknemer selecteert en voldoet aan de argumenten. 3. De query wordt uitgevoerd 4. Uit het aantal resultaten wordt opgemaakt of de query succesvol was. 5. De resultaten worden uit de datasource in het object geplaatst 6. True wordt als resultaat van de functie teruggegeven.
Uitzonderingen	Er is geen werknemer gevonden die voldoet aan criteria gegeven in de argumenten
Oplossing uitzonderingen	Resultaat uitzondering tonen aan de gebruiker en vastleggen in het logbestand (beveiligingsovertreding)

Object	Client
Functienaam	opslaanGegevens
Resultaat Type	Boolean
Argumenten	gegevensArray : array of char
Preconditie	De gegevensArray is een verzameling van alle gegevens van client in de vorm van een array
Postconditie	De gegevens in de array zijn vastgelegd in het object client
Beschrijving	1. Na het klikken op de button 'vastleggen' worden de gegevens in de tekstvelden van de GUI in een array geplaatst 2. De array wordt bij de aanroep van de functie als argument meegegeven 3. De lengte van de array is variabel en wordt vastgesteld door het aantal velden op te vragen middels de Length()-functie 4. Als de lengte van de array gelijk is aan het aantal velden van het object, wordt de array sequentieel uitgelezen. 5. Vervolgens wordt er gecontroleerd of de verplichte velden in het object mogen in de array geen lege waarden zijn. 6. Het resultaat van de functie is true
Uitzonderingen	5a. Een verplicht veld in het object is als leeg veld ingevuld
Oplossing uitzonderingen	5a. De array wordt niet opgeslagen in het object en de gebruiker wordt ingelicht over het foutief ingevulde veld.

Object	Partner
Functienaam	opslaanGegevens
Resultaat Type	Boolean
Argumenten	gegevensArray : array of char
Preconditie	De gegevensArray is een verzameling van alle gegevens van partner in de vorm van een array
Postconditie	De gegevens in de array zijn vastgelegd in het object partner
Beschrijving	1. De gegevens van de partner zijn in het opgegeven array als argument meegegeven 2. De lengte van de array is gelijk aan het aantal attributen in het object. 3. De verplichte velden in het object bevatten geen lege waarde in de array 4. De array wordt sequentieel uitgelezen en opgeslagen in het object

	5. ‘True’ wordt teruggegeven als resultaat van de functie.
Uitzonderingen	3a. De array bevat lege waarden, die in het object verplicht zijn gesteld.
Oplossing uitzonderingen	3a. Informeer de gebruiker over het ontbreken van de verplichte waarde(n)

Object	Contactmoment
Functienaam	controlerenDatum
Resultaat Type	Boolean
Argumenten	datum : string
Preconditie	De datum is niet leeg
Postconditie	De datum voldoet aan de eisen die gesteld zijn aan het Datumtype voor de database.
Beschrijving	1. De opgegeven datum wordt gecontroleerd op notatiewijze en het tijdstip ten opzichte van de huidige datum 2. Het resultaat van deze controle wordt als resultaat in de vorm van een boolean teruggegeven.
Uitzonderingen	1a. De notatiewijze is incorrect of de datum is in het verleden
Oplossing uitzonderingen	1a. De boolean wordt als false teruggegeven en de gebruiker wordt toegelicht over de mislukte controle.

Object	Partus
Functienaam	vastleggenVoornemens
Resultaat Type	Boolean
Argumenten	gegevensArray : array of char
Preconditie	De gegevensArray is gevuld met de velden in de volgorde zoals ze ook in de database zijn aangebracht
Postconditie	De gegevens zijn succesvol in het object geplaatst en het resultaat van de functie is de boolean ‘true’
Beschrijving	1. De gegevens van de partner zijn in het opgegeven array als argument meegegeven 2. De lengte van de array is gelijk aan het aantal attributen in het object. 3. De verplichte velden in het object bevatten geen lege waarde in de array 4. De array wordt sequentieel uitgelezen en opgeslagen in het object 5. ‘True’ wordt teruggegeven als resultaat van de functie.
Uitzonderingen	3a. De gegevensArray bevat lege velden
Oplossing uitzonderingen	3a. Informeer de gebruiker over het ontbreken van de verplichte waarde(n)

Object	Algemeen
Functienaam	vastleggen_a_anamnese
Resultaat Type	Boolean
Argumenten	gegevensArray : array of char
Preconditie	-
Postconditie	Het object algemeen is aangevuld met de algemene anamnese
Beschrijving	1. De gegevens van de partner zijn in het opgegeven array als argument meegegeven 2. De lengte van de array is gelijk aan het aantal attributen in het object.

	3. De verplichte velden in het object bevatten geen lege waarde in de array 4. De array wordt sequentieel uitgelezen en opgeslagen in het object 5. 'True' wordt teruggegeven als resultaat van de functie.
Uitzonderingen	3a. De gegevensArray bevat lege velden
Oplossing uitzonderingen	3a. Informeer de gebruiker over het ontbreken van de verplichte waarde(n)

Object	Familie
Functienaam	vastleggen_fpartner
Resultaat Type	Boolean
Argumenten	gegevensArray : array of char
Preconditie	-
Postconditie	Het object is aangevuld met de familie anamnese van de partner
Beschrijving	1. De gegevens van de partner zijn in het opgegeven array als argument meegegeven 2. De lengte van de array is gelijk aan het aantal attributen in het object. 3. De verplichte velden in het object bevatten geen lege waarde in de array 4. De array wordt sequentieel uitgelezen en opgeslagen in het object 5. 'True' wordt teruggegeven als resultaat van de functie.
Uitzonderingen	3a. De gegevensArray bevat lege velden
Oplossing uitzonderingen	3a. Informeer de gebruiker over het ontbreken van de verplichte waarde(n)

Object	Familie
Functienaam	vastleggen_fclient
Resultaat Type	Boolean
Argumenten	gegevensArray : array of char
Preconditie	-
Postconditie	Het object is aangevuld met de familie anamnese van de cliënt
Beschrijving	1. De gegevens van de partner zijn in het opgegeven array als argument meegegeven 2. De lengte van de array is gelijk aan het aantal attributen in het object. 3. De verplichte velden in het object bevatten geen lege waarde in de array 4. De array wordt sequentieel uitgelezen en opgeslagen in het object 5. 'True' wordt teruggegeven als resultaat van de functie.
Uitzonderingen	3a. De gegevensArray bevat lege velden
Oplossing uitzonderingen	3a. Informeer de gebruiker over het ontbreken van de verplichte waarde(n)

Object	Termijn
Functienaam	bepalen
Resultaat Type	string
Argumenten	-

Preconditie	-
Postconditie	De termijn van de zwangere is bepaald
Beschrijving	1. Controleren van de laatste menstruatie ten opzichte van de huidige datum 2. Controleren van de menstruatiecyclus in het gegeven aantal dagen 3. Op basis van de laatste menstruatie en de menstruatie cyclus wordt een datum vastgesteld waarin de client kan gaan bevallen 4. Deze termijn wordt teruggegeven als string
Uitzonderingen	1a. De laatste menstruatie datum is niet ingevuld of is incorrect (te ver in het verleden) 2a. De menstruatiecyclus moet binnen een minimaal en een maximaal aantal dagen vallen en niet aangegeven is dat deze onbekend is.
Oplossing uitzonderingen	1a/2a Terugkoppelen aan de gebruiker

Object	Termijn
Functienaam	controlerenDatum
Resultaat Type	Boolean
Argumenten	datum : string
Preconditie	Het argument datum is de datum waarop de ‘A Terme’ is vastgesteld
Postconditie	De datum is correct in verhouding met de ‘A Terme’
Beschrijving	1. De datum wordt gecontroleerd op notatiewijze 2. Er wordt gecontroleerd of op basis van de A Terme datum deze datum kan zijn ingevuld. 3. Het resultaat is ‘true’
Uitzonderingen	1a. Notatie van de datum is incorrect 2a. De verkregen datum is in verhouding met de A-terme datum ongeldig
Oplossing uitzonderingen	1a/2a. De gebruiker wordt ingelicht over de ongeldigheid van de data

Object	Termijn
Functienaam	controlerenATerme
Resultaat Type	Boolean
Argumenten	termijnDatum : string termijn : string
Preconditie	Verkregen termijnDatum is bepaald door de functie bepalen van het object Termijn.
Postconditie	De notatie van de datum is correct en het resultaat van de functie is ‘true’. De termijnDatum is correct opgesteld aan de hand van het verkregen termijn.
Beschrijving	1. De datum wordt gecontroleerd op notatie en of deze niet in het verleden is en of deze klopt met de verkregen termijn. 2. Het resultaat is ‘true’
Uitzonderingen	1a De datum is incorrect of in het verleden
Oplossing uitzonderingen	1a Resultaat is ‘false’

Object	Echo
Functienaam	controlerenDatum
Resultaat Type	Boolean

Argumenten	datum : string
Preconditie	-
Postconditie	De datum is correct gezien het stadium van de zwangerschap
Beschrijving	1. De echo's worden alleen in bepaalde weken ten opzichte van de 'A terme' datum verkregen, dit wordt eerst gecontroleerd. 2. Het resultaat is 'true'
Uitzonderingen	1a. De echo's bevinden zich buiten het standaardstadium
Oplossing uitzonderingen	1a. De echo's worden vastgelegd en dit wordt gemeld aan de gebruiker.

Object	Onderzoek
Functienaam	controlerenLengte
Resultaat Type	Boolean
Argumenten	lengte : real
Preconditie	-
Postconditie	De lengte valt binnen de geldige waarden en het resultaat is 'true'.
Beschrijving	1. Er wordt gecontroleerd of de lengte in meters is. (0.4 > lengte < 3) 2. Het resultaat is 'true'
Uitzonderingen	1a De lengte is niet in meters, groter dan 3 of kleiner dan 0.4.
Oplossing uitzonderingen	1a Er wordt gecontroleerd of deze in centimeters is weergegeven (50 > lengte < 300). Deze lengte wordt omgezet naar meters. 1b De lengte is ongeldig en de gebruiker wordt op de hoogte gesteld.

Object	Onderzoek
Functienaam	controlerenGewichtverschil
Resultaat Type	Boolean
Argumenten	gewicht_voor : real gewicht_onderzoek : real
Preconditie	-
Postconditie	De gewichten zijn in kilogrammen en zijn correct ten opzichte van elkaar. Het resultaat is 'true'
Beschrijving	1. De gewichten worden gecontroleerd op notatiewijze (kilogrammen) 2. De gewichten worden ten opzichte van elkaar gecontroleerd. Het verschil kan niet groter zijn dan 20 kilogram. 3. Het resultaat is 'true'
Uitzonderingen	1a De gewichten zijn niet in kilogram (20 > gewicht < 500) 2a Het verschil in gewichten komt niet overeen. (verschil > 50 kilogram)
Oplossing uitzonderingen	1a Het resultaat is 'false' en de gebruiker wordt ingelicht. 2a Het resultaat is 'false' en de gebruiker wordt ingelicht.

4.1.7. Bouweenheid 7: Ontwikkelen functies subsysteem Zwangerschap

Na de bouw van het subsysteem 'contactmoment client', kon worden begonnen met het ontwerpen van de functionaliteiten in het kader van het subsysteem 'zwangerschap'.

Deze bouweenheid omvat de beschrijvingen van de functies die in de sequencediagrammen van de functionele structuur van de pilot globaal zijn gedefinieerd.

Object	Client
Functienaam	zoeken
Resultaat Type	TDataSource
Argumenten	naam : string adres : string (straatnaam) geboortedatum : string
Preconditie	Minimaal een van de argumenten bevat zoekcriteria
Postconditie	Het aantal gevonden cliënten in de database is in de teruggeven als resultaat.
Beschrijving	1. De functie controleert of minimaal een van de argumenten niet leeg is. 2. Er wordt een query opgesteld die zoekt naar de cliënten die voldoen aan de opgegeven argumenten 3. De query wordt uitgevoerd. 4. De resultaten indien meer dan 0 worden teruggegeven als resultaat
Uitzonderingen	4. Het aantal gevonden cliënten die voldoen aan de criteria is 0
Oplossing uitzonderingen	4. Er wordt 'nil' teruggegeven in plaats van een TDataSource

Naast deze functie, zijn de functionaliteiten voor het vastleggen, wijzigen, verwijderen en opvragen reeds in de vijfde bouweenheid ontwikkeld.

4.1.8. Bouweenheid 8: Ontwikkelen functies subsysteem Administratie

Na het ontwikkelen van de functies voor het subsysteem 'zwangerschap', zijn de functies voor het subsysteem 'Administratie' ontwikkeld. Deze bouweenheid omvat de beschrijvingen van de functies die geïmplementeerd worden in het kader van dit subsysteem. Deze functies zijn opgesteld op basis van de sequencediagrammen beschreven in de functionele structuur.

Object	Termijn
Functienaam	aantalZwangeren
Resultaat Type	integer
Argumenten	min_datum : string max_datum : string
Preconditie	De minimale datum ligt niet in de toekomst ten opzichte van de maximale_datum
Postconditie	Het aantal zwangere vrouwen in het stadium van de minimale datum tot de maximale datum is teruggegeven als resultaat
Beschrijving	1. De minimale datum moet in het verleden liggen ten opzichte van de maximale datum 2. Er wordt gecontroleerd of de data voldoen aan de eisen gesteld aan het type datum, in de database. 3. De query wordt opgesteld, die zoekt naar de zwangeren in de opgegeven periode. 4. De query wordt uitgevoerd 5. Het aantal resultaten van de query wordt teruggeven als resultaat.
Uitzonderingen	5a. Er zijn geen zwangeren in die periode
Oplossing uitzonderingen	5a. Een melding aan de gebruiker waarin de minimale datum

	en de maximale datum nogmaals weergegeven wordt en het aantal resultaten wordt vermeld.

Object	Client
Functienaam	geefClienten
Resultaat Type	TDatasource
Argumenten	Het aantal zwangeren in de gegeven periode is groter dan 0.
Preconditie	-
Postconditie	De zwangeren cliënten zijn in de eerder bepaalde periode in het resultaat verwerkt.
Beschrijving	1. De functie gebruikt de data die opgegeven zijn als argumenten bij de functie aantalZwangeren 2. De query wordt opgesteld die de cliënten ophaalt uit de database. 3. De query wordt uitgevoerd 4. Het resultaat van deze query wordt direct een resultaat teruggegeven aan de functie.
Uitzonderingen	-
Oplossing uitzonderingen	-

4.1.9. Bouweenheid 9: Opstellen gebruikershandleiding subsystemen

Na het ontwerp en de bouw van de functies en procedures van de applicatie, kan worden begonnen met het opstellen van de gebruikershandleiding voor de subsystemen. Om een geschikte gebruikershandleiding op te stellen die een toegevoegde waarde heeft voor het gebruik van de applicatie, is het type handleiding in overleg met de opdrachtgever vastgesteld.

In overleg met de opdrachtgever is besloten om een quick-reference guide te maken. De handleiding die voor de huidige applicatie bestond werd alleen door de opdrachtgever gebruikt. De quick-reference guide moet klein zijn en zal minimaal moeten bestaan uit de volgende vier punten:

- Overzicht van alle functies van de applicatie
- De locatie en het doel van iedere functie
- Alle schermen moeten worden opgenomen en iedere optie in ieder scherm moet worden toegelicht
- Een inhoudsopgave en een symbolenlijst

4.2. Testspecificaties

Na implementatie van iedere bouweenheid dienen deze te worden getest aan de hand van de opgestelde testspecificaties. Zoals vastgesteld is in het plan van aanpak van het project zullen de bouweenheden van de derde en vierde pilot volgens het black-box principe worden getest. Deze manier van testen bekijkt op basis van gegeven invoer of de verwachte uitvoer ook de daadwerkelijke uitvoer is. Hieronder zijn de testspecificaties opgesteld die na de bouw van iedere bouweenheid worden uitgevoerd.

4.2.1. Bouweenheid 1

Bouweenheid 1 levert een basisstructuur op waarin het MD5 algoritme en de globale functies zijn geïmplementeerd. Om te testen of deze na de bouw de verwachte uitvoer leveren zijn de volgende testspecificaties opgesteld.

Case #	**Test**	**Invoer**	**Verwachte uitvoer**
1.0	getSelectedComboItem geeft een geselecteerd combo-item terug als string	Selecteer een veld uit een combobox en voer de functie uit met value 0	Het geselecteerde item wordt als string teruggegeven
1.1	getSelectedComboItem geeft een geselecteerd combo-item terug als index	Selecteer een veld uit een combobox en voer de functie uit met value 1	Het geselecteerde item wordt als index teruggegeven
1.2	getSelectedRadioItem geeft een geselecteerd radio-item terug als string	Selecteer een item uit een radiogroup en voer de functie uit met value 0	Het geselecteerde item wordt als string teruggegeven
1.3	getSelectedRadioItem geeft een geselecteerd radio-item terug als index	Selecteer een item uit een radiogroup en voer de functie uit met value 1	Het geselecteerde item wordt als index teruggegeven
1.4	groupActivate zet de gegeven groep op de voorgrond	groupActivate(grpClient)	De grpClient wordt op de voorgrond weergegeven
1.5	groupDeactivate verbergt de groep	groupDeactivate(grpClient)	De grpClient wordt verborgen
1.6	deactivateAllGroups verbergt alle groepen	deactivateAllGroups(alle groepen)	De groepen zijn allemaal verborgen
1.7	WindowExists zoekt naar een bepaalde applicatie in de proceslijst van Windows	WindowExists(‘W!se Delivery’, ‘TForm’)	Het window wordt gevonden.
1.8	isStrEmpty controleert of een string daadwerkelijk leeg is	testString := ‘’; isStrEmpty(testString)	Het resultaat van de functie is true.
1.9	strToArr converteert de gegeven string naar een gegeven array.	testString := ‘Henk’; voornaam: TNames; strToArr(testString, voornaam);	voornaam bevat de naam ‘Henk’
1.10	arrToStr converteert een gegeven array naar een string	testString := arrToSrr(voornaam)	testString is ‘Henk’
1.11	MD5Print() geeft een geldige MD5 encryptie	MD5Print(MD5String(‘hallo’)	MD5 ‘hallo’ = 598d4c200461b815

		)	22a3328565c25f7c

4.2.2. Bouweenheid 2

Bouweenheid 2 levert een lijst met constanten een globale variabelen. Om te testen of de globale en de constanten goed gedefinieerd zijn, is voor iedere variant een case opgesteld.

Case #	**Test**	**Invoer**	**Verwachte uitvoer**
2.0	De klasse TDb is globaal gedefinieerd en benaderbaar	TDb.sql_query('Select * from medewerker');	TDatasource is niet nil.
2.1	Een enter in een tekstring is te gebruiken in een berichtvenster.	showMessage('Dit is een test'+CRLF+'nieuwe regel')	Toont een berichtvenster met daarin de twee gescheiden regels.

4.2.3. Bouweenheid 3

Deze bouweenheid definieert de standaardisering voor het commentaar. Het doel hiervan is dat iedere functie in de applicatie voorzien is van commentaar volgens de standaard. Om te testen of hieraan is voldaan zijn alle functies gecontroleerd op de onderstaande punten:

- Controleren of iedere unit met een 'comment block' begint.
- Controleren of iedere functie/procedure binnen die unit ook een 'comment block' bevat.

4.2.4. Bouweenheid 4

Bouweenheid 4 moet het omzetten van de klassen naar objecten in Delphi realiseren. Om te testen of de omzetting correct is verlopen, is een steekproef genomen voor het object TClient. Hieronder zijn de testspecificaties opgenomen voor dit object.

Case #	**Test**	**Invoer**	**Verwachte uitvoer**
4.0	De naamgeving is correct ingevoerd	Client := TClient.Create	Client is een instantie van TClient
4.1	De attributen in de objecten zijn globaal gedefinieerd	Client.voornaam = 'Cees'	De Client.voornaam is 'Cees'.
4.2	De functies en procedures zijn als dummy aan te roepen.	Client.vastleggen()	De aanroep is geldig.

4.2.5. Bouweenheid 5

In de vijfde bouweenheid worden de standaardfuncties en -procedures van de objecten gerealiseerd. Deze standaardfuncties stellen de queries op die uitgevoerd moeten worden door de interface. Om te testen of deze functies voldoen aan de gestelde eisen, zijn de testspecificaties opgesteld voor iedere functie eigendom van het object TClient.

Case #	**Test**	**Invoer**	**Verwachte uitvoer**
5.0	De functie vastleggen legt de cliëntgegevens vast in de database.	Client.vastleggen()	De cliëntgegevens staan in de database.
5.1	De functie wijzigen, wijzigt de cliëntgegevens in de database.	Client.wijzigen()	De cliëntgegevens zijn gewijzigd in de database.
5.2	De functie verwijderen, verwijdert de cliëntgegevens uit de database.	Client.verwijderen()	De cliëntgegevens zijn verwijderd uit de database.
5.3	De functie opvragen, haalt de cliëntgegevens uit de database op.	Client.nummer := 1; Client.ophalen()	De cliëntgegevens van client met nummer ‘1’, zijn vastgelegd in het object Client.

4.2.6. Bouweenheid 6

In deze bouweenheid worden de functies van het subsysteem ‘Contactmoment Client’ geïmplementeerd. Door de onderstaande testcases uit te voeren kan worden getest of deze functies voldoen aan de gestelde eisen. Deze testcases zijn opgesteld op basis van de use-cases, indien zich een fout voordoet in een van de testcases, zal dieper worden ingegaan op de functies specifiek voor die use-case.

Case #	**Test**	**Invoer**	**Verwachte uitvoer**
6.0	Een verloskundige kan inloggen	De verloskundige logt in met naam en wachtwoord	De verloskundige is ingelogd
6.1	Een verloskundige kan een client toevoegen	De verloskundige voert de gegevens van een client in en klikt op de button vastleggen	De cliëntgegevens zijn vastgelegd in de database.
6.2	Een verloskundige kan een	De verloskundige klikt op de button	Het contactmoment is opgeslagen in de

	contactmoment bijwerken	vastleggen	database
6.3	Een verloskundige kan de voornemens registreren	De verloskundige voert de voornemens in en klikt op de button vastleggen.	De voornemens zijn in de database vastgelegd.
6.4	De toxische factoren kunnen worden bijgewerkt	De verloskundige vult de algemene anamnese en de familieanamnese in	De toxische factoren en anamnese is vastgelegd in de database.
6.5	Het termijn kan worden ingesteld	De verloskundige voert de gegevens in en klikt op de button vastleggen	Het termijn is berekend en opgeslagen in de database.
6.6	De obstetrische anamnese kan worden bijgewerkt	De verloskundige voert de nodige gegevens in en selecteert de uitkomst en klikt op vastleggen	De obstetrische anamnese is in de database vastgelegd.
6.7.1	De uitkomst kan worden bepaald (labuitslagen)	De verloskundige voert de noodzakelijk labgegevens in en klikt op de button vastleggen	De labuitslagen zijn in de database vastgelegd.
6.7.2	De uitkomst kan worden bepaald (echo's)	De verloskundige klikt op de knop 'nieuwe echo' na het invoeren van de gegevens	De echo is vastgelegd in de database.
6.8	Het onderzoek kan worden vastgelegd	De verloskundige voert de gegevens in en klikt op vastleggen	De onderzoeksgegevens zijn in de database opgeslagen.

4.2.7. Bouweenheid 7

De zevende bouweenheid omvat de implementatie van het subsysteem 'Zwangerschap'. Om te garanderen dat de functies correct zijn geïmplementeerd, zijn de volgende testcases opgesteld.

Case #	**Test**	**Invoer**	**Verwachte uitvoer**
7.0	De baring kan worden ingevoerd	De baringgegevens van een client worden ingevoerd en vastleggen wordt aangeklikt	De baringgegevens zijn vastgelegd in de database.

7.1	De gegevens van het kraambed kunnen worden ingevoerd.	De gegevens worden ingevuld en er wordt op vastleggen geklikt	De kraambedgegevens zijn in de database opgeslagen
7.2	De gegevens van de baby kunnen worden ingevoerd.	De babygegevens worden ingevoerd en er wordt op vastleggen geklikt	De babygegevens zijn ingevoerd in de database.
7.3	De verloskundige kan zoeken naar cliënten	De criteria wordt opgegeven en er wordt op zoeken geklikt	De resultaten die voldoen aan de criteria worden getoond

4.2.8. Bouweenheid 8

Bouweenheid 8 omvat de implementatie van het subsysteem ‘Administratie’. Om te testen of de implementatie hiervan succesvol is uitgevoerd zijn onderstaande cases opgesteld

Case #	**Test**	**Invoer**	**Verwachte uitvoer**
8.0	De verloskundige kan het aantal niet bevallen zwangeren opvragen	De verloskundige drukt op de knop rapporteren	Een lijst met nog niet bevallen zwangeren wordt getoond

4.2.9. Bouweenheid 9

Tot slot zal in de laatste bouweenheid de gebruikershandleiding worden opgesteld. Om te controleren of deze aan de gestelde eisen voldoet, is in samenwerking met de opdrachtgever gekeken of de quick-reference guide voldoet aan de onderstaande eisen:

1. Bevat een overzicht van alle functies van de applicatie
2. Beschrijft de locatie en het doel van iedere functie
3. De schermen zijn opgenomen met een combinatie van een grafische en tekstuele toelichting
4. Een inhoudsopgave en een symbolenlijst is aanwezig.

5. Invoeringsprocedure pilot

Wanneer de bouweenheden van de pilot zijn geïmplementeerd en de testcases per bouweenheid zijn uitgevoerd, kan worden overgegaan tot het invoeren van de pilot. Dit hoofdstuk beschrijft de procedure van deze pilot die dient te worden gevolgd bij de invoering. In de eerste paragraaf wordt het plan van invoering beschreven en in paragraaf 5.2 de testspecificaties van deze invoeringsprocedure.

5.1. Invoeringsplan

Bij de invoering van deze pilot is een invoeringsprocedure opgesteld bestaande uit twee stappen. Bij deze twee stappen wordt er van uitgegaan dat deze pilot als onderdeel van de applicatie wordt ingevoerd.

1. Controleren operationele software
   Software:
   - MySQL
   - MyODBC

2. Controleren of er een netwerk is aangelegd, zodat de applicatie correct kan opereren in de huidige omgeving.

3. Invoeren nieuwe applicatie met operationele subsystemen

5.2. Testspecificaties invoeringsprocedure

Hieronder is een lijst met testspecificaties opgesteld die ter controle dienen te worden uitgevoerd om na te zien op de correcte invoering van de pilot.

Testspecificatie Invoeringsplan			
Case #	**Test**	**Invoer**	**Verwachte uitvoer**
10.0	De MySQL-database is operationeel	Opvragen van lijst met actieve processen	MySQL is actief
10.1	MyODBC is juist ingesteld en werkt correct	Uitvoeren testfunctie van MyODBC	MyODBC werkt correct
10.2	Het netwerk is aangelegd en operationeel	Opvragen lijst met netwerkadapters	Het netwerk is operationeel

6.

Testen en beoordelen pilot

Na de implementatie van iedere bouweenheid zijn deze getest aan de hand van de specificaties die in hoofdstuk 4 van dit document zijn opgesteld. De resultaten van deze uitgevoerde tests worden in dit hoofdstuk beschreven. In de eerste paragraaf worden de resultaten en omgeving beschreven en in paragraaf 6.2 wordt het eindoordeel van de pilot beschreven.

6.1. Testomgeving

De testcases voor iedere bouweenheid zijn op een lokaal systeem uitgevoerd. Deze lokale omgeving bevat dezelfde specificaties als die in de vorige pilot zijn opgesteld:

- Athlon XP 2400+
- 1280 MB RAM

De software die op dit systeem aanwezig was, is hieronder opgenomen:

- Borland Delphi Enterprise Edition
- Windows XP besturingssysteem
- Microsoft Office XP
- MySQL DBMS
- MyODBC

6.1.1. Testen Bouweenheid 1

Hieronder zijn in het kader van de eerste bouweenheid “Definiëren basisstructuur”, de onderstaande resultaten van de testcases opgenomen.

Case #	Test	Invoer	Verwachte uitvoer	Resultaat
1.0	getSelectedComboItem geeft een geselecteerd combo-item terug als string	Selecteer een veld uit een combobox en voer de functie uit met value 0	Het geselecteerde item wordt als string teruggegeven	OK
1.1	getSelectedComboItem geeft een geselecteerd combo-item terug als index	Selecteer een veld uit een combobox en voer de functie uit met value 1	Het geselecteerde item wordt als index teruggegeven	OK
1.2	getSelectedRadioItem geeft een geselecteerd radio-item terug als string	Selecteer een item uit een radiogroup en voer de functie uit met value 0	Het geselecteerde item wordt als string teruggegeven	OK
1.3	getSelectedRadioItem geeft een geselecteerd radio-item terug als index	Selecteer een item uit een radiogroup en voer de functie	Het geselecteerde item wordt als	OK

		uit met value 1	index teruggegeven	
1.4	groupActivate zet de gegeven groep op de voorgrond	groupActivate(grpClient)	De grpClient wordt op de voorgrond weergegeven	OK
1.5	groupDeactivate verbergt de groep	groupDeactivate(grpClient)	De grpClient wordt verborgen	OK
1.6	deactivateAllGroups verbergt alle groepen	deactivateAllGroups(alle groepen)	De groepen zijn allemaal verborgen	OK
1.7	WindowExists zoekt naar een bepaalde applicatie in de proceslijst van Windows	WindowExists(‘W!se Delivery’, ‘TForm’)	Het window wordt gevonden.	OK
1.8	isStrEmpty controleert of een string daadwerkelijk leeg is	testString := ‘’; isStrEmpty(testString)	Het resultaat van de functie is true.	OK
1.9	strToArr converteert de gegeven string naar een gegeven array.	testString := ‘Henk’; voornaam: TNames; strToArr(testString, voornaam);	voornaam bevat de naam ‘Henk’	OK
1.10	arrToStr converteert een gegeven array naar een string	testString := arrToSrr(voornaam)	testString is ‘Henk’	OK
1.11	MD5Print() geeft een geldige MD5 encryptie	MD5Print(MD5String(‘hallo’))	MD5 ‘hallo’ = 598d4c200461b815 22a3328565c25 f7c	OK

6.1.2. Testen Bouweenheid 2

In bouweenheid 2 “Definiëren constanten en globale variabelen” zijn de constanten en globale variabelen gedefinieerd. Hieronder zijn de testresultaten van de testcases voor deze bouweenheid opgenomen.

Case #	**Test**	**Invoer**	**Verwachte uitvoer**	**Resultaat**
2.0	De klasse TDb is globaal gedefinieerd en benaderbaar	TDb.sql_query(‘Select * from medewerker’);	TDatasource is niet nil.	OK
2.1	Een enter in	showMessage(‘Dit	Toont een berichtvenster	OK

	een tekstring is te gebruiken in een berichtvenster.	is een test'+CRLF+'nieu we regel')	met daarin de twee gescheiden regels.	

6.1.3. Testen Bouweenheid 3

Voor bouweenheid 3 "Standaardisering codecommentaar", zijn geen black-box testcases opgesteld, maar is getest of aan de onderstaande eisen is voldaan.

Case #	Test	Resultaat
3.0	Controleren of iedere unit met een 'comment block' begint.	Mislukt
3.1	Controleren of iedere functie/procedure binnen die unit ook een 'comment block' bevat.	OK

Mislukte case 3.0:
Niet iedere unit was voorzien van een comment block. Hoewel de functies/procedures allemaal zijn voorzien van commentaar, was dit bij de units van het hoofdscherm niet het geval.
De oplossing hiervoor was om het commentaar alsnog toe te voegen, waarna case 3.0 als succesvol uitgevoerd kan worden beschouwd.

6.1.4. Testen Bouweenheid 4

In Bouweenheid 4 "Omzetten klassen naar objecten", worden de operaties uitgevoerd om de klassen om te zetten naar objecten. De onderstaande resultaten zijn deze van de opgestelde testcases uit hoofdstuk 4.

Case #	Test	Invoer	Verwachte uitvoer	Resultaat
4.0	De naamgeving is correct ingevoerd	Client := TClient.Create	Client is een instantie van TClient	OK
4.1	De attributen in de objecten zijn globaal gedefinieerd	Client.voornaam = 'Cees'	De Client.voornaam is 'Cees'.	OK
4.2	De functies en procedures zijn als dummy aan te roepen.	Client.vastleggen()	De aanroep is geldig.	OK

6.1.5. Testen Bouweenheid 5

Hieronder zijn de resultaten van de testcases opgenomen die in het kader van de vijfde bouweenheid “Ontwikkelen functies en procedures” zijn uitgevoerd.

Case #	Test	Invoer	Verwachte uitvoer	Resultaat
5.0	De functie vastleggen legt de cliëntgegevens vast in de database.	Client.vastleggen()	De cliëntgegevens staan in de database.	OK
5.1	De functie wijzigen, wijzigt de cliëntgegevens in de database.	Client.wijzigen()	De cliëntgegevens zijn gewijzigd in de database.	OK
5.2	De functie verwijderen, verwijdert de cliëntgegevens uit de database.	Client.verwijderen()	De cliëntgegevens zijn verwijderd uit de database.	OK
5.3	De functie opvragen, haalt de cliëntgegevens uit de database op.	Client.nummer := 1; Client.ophalen()	De cliëntgegevens van client met nummer ‘1’, zijn vastgelegd in het object Client.	OK

6.1.6. Testen Bouweenheid 6

Bouweenheid 6 “Ontwikkelen functies subsysteem Contactmoment cliënt“ omvat na invoering het eerste functionele subsysteem. Om te controleren of het subsysteem voldoet aan de gestelde eisen, zijn de testcases uitgevoerd. De resultaten hiervan zijn in de volgende tabel opgenomen.

Case #	Test	Invoer	Verwachte uitvoer	Resultaat
6.0	Een verloskundige kan inloggen	De verloskundige logt in met naam en wachtwoord	De verloskundige is ingelogd	OK
6.1	Een verloskundige kan een client toevoegen	De verloskundige voert de gegevens van een client in en klikt op de button vastleggen	De cliëntgegevens zijn vastgelegd in de database.	OK
6.2	Een verloskundige kan een contactmoment	De verloskundige klikt op de button vastleggen	Het contactmoment is opgeslagen in de database	OK

	bijwerken			
6.3	Een verloskundige kan de voornemens registreren	De verloskundige voert de voornemens in en klikt op de button vastleggen.	De voornemens zijn in de database vastgelegd.	OK
6.4	De toxische factoren kunnen worden bijgewerkt	De verloskundige vult de algemene anamnese en de familieanamnese in	De toxische factoren en anamnese is vastgelegd in de database.	OK
6.5	Het termijn kan worden ingesteld	De verloskundige voert de gegevens in en klikt op de button vastleggen	Het termijn is berekend en opgeslagen in de database.	OK
6.6	De obstetrische anamnese kan worden bijgewerkt	De verloskundige voert de nodige gegevens in en selecteert de uitkomst en klikt op vastleggen	De obstetrische anamnese is in de database vastgelegd.	OK
6.7.1	De uitkomst kan worden bepaald (labuitslagen)	De verloskundige voert de noodzakelijk labgegevens in en klikt op de button vastleggen	De labuitslagen zijn in de database vastgelegd.	OK
6.7.2	De uitkomst kan worden bepaald (echo's)	De verloskundige klikt op de knop 'nieuwe echo' na het invoeren van de gegevens	De echo is vastgelegd in de database.	OK
6.8	Het onderzoek kan worden vastgelegd	De verloskundige voert de gegevens in en klikt op vastleggen	De onderzoeksgegevens zijn in de database opgeslagen.	OK

6.1.7. Testen Bouweenheid 7

Na het ontwikkelen van bouweenheid 7 "Ontwikkelen functies subsysteem Zwangerschap", moet het mogelijk zijn om de functies van het tweede subsysteem zonder probleem te kunnen uitvoeren. Om op het gewenste resultaat na te zien, zijn de bouweenheden getest. Hieronder zijn de resultaten van deze tests opgenomen.

Case #	Test	Invoer	Verwachte uitvoer	Resultaat
7.0	De baring kan worden ingevoerd	De baringgegevens van een client worden ingevoerd en vastleggen wordt aangeklikt	De baringgegevens zijn vastgelegd in de database.	OK
7.1	De gegevens van het kraambed kunnen worden ingevoerd.	De gegevens worden ingevuld en er wordt op vastleggen geklikt	De kraambedgegevens zijn in de database opgeslagen	OK
7.2	De gegevens van de baby kunnen worden ingevoerd.	De babygegevens worden ingevoerd en er wordt op vastleggen geklikt	De babygegevens zijn ingevoerd in de database.	OK
7.3	De verloskundige kan zoeken naar cliënten	De criteria wordt opgegeven en er wordt op zoeken geklikt	De resultaten die voldoen aan de criteria worden getoond	OK

6.1.8. Testen Bouweenheid 8

Bouweenheid 8 “Ontwikkelen functies subsysteem Administratie” omvat de functies voor het laatste subsysteem. Door het vervallen van de “LVR-controle”, is de functionaliteit van dit subsysteem beperkt. De resultaten van de uitgevoerde testcase zijn opgenomen in deze paragraaf.

Case #	Test	Invoer	Verwachte uitvoer	Resultaat
8.0	De verloskundige kan het aantal niet bevallen zwangeren opvragen	De verloskundige drukt op de knop rapporteren	Een lijst met nog niet bevallen zwangeren wordt getoond	OK

6.1.9. Testen Bouweenheid 9

De laatste bouweenheid “Opstellen gebruikershandleiding subsystemen” bevat het opstellen van een quick-reference guide. In samenwerking met de opdrachtgever is gecontroleerd of de opgestelde ‘guide’ voldeed aan de gestelde eisen. De resultaten van deze controle zijn hieronder opgenomen.

Case #	Test	Resultaat
9.0	Bevat een overzicht van alle functies van de applicatie	OK
9.1	Beschrijft de locatie en het doel van iedere functie	OK
9.2	De schermen zijn opgenomen met een combinatie van een grafische en tekstuele toelichting	OK
9.3	Een inhoudsopgave en een symbolenlijst is aanwezig.	OK

6.2. Beoordeling pilot

In deze paragraaf wordt beoordeeld of de pilot voldoet aan de gestelde eisen, door na te gaan of voldaan is aan de functionele systeemeisen uit definitiestudie.

Code	Systeemeis	Voldaan
M1	Het systeem moet de gegevens op de oude zwangerschapskaart kunnen realiseren.	Ja
M2	De gegevens van cliënten die verwerkt worden in het systeem moeten grotendeels variabel zijn. (Doktoren, ziekenhuizen, ziektes, etc)	Ja
M3	Het moet mogelijk zijn om te zoeken naar de cliënten op eventueel unieke gegevens	Ja
M4	Het moet mogelijk zijn om ziekenhuizen bij te werken en er in te zoeken	Nee, onderdeel subsysteem 4
M5	Ziekenhuizen, verzekeraars, gynaecologen, doktoren, plaatsnamen, achternamen en voornamen. Deze gegevens moeten allemaal in een ‘zelfgroeiende’ lijst kunnen worden geselecteerd, zodra het systeem vereist dat deze worden ingevoerd.	Ja
S1	Het systeem zal middels een functie de cliëntdata moeten kunnen exporteren naar een locatie buiten het systeem.	Nee, onderdeel subsysteem 4

S2	De gebruiker en/of beheerder kan alleen toegang krijgen tot het systeem middels een gebruikersnaam en wachtwoord	Ja
S3	Er moet een mogelijkheid zijn om de gegevens te rapporteren	Ja
C1	De LVR (Landelijke Verloskunde Registratie) vereist dat er periodiek een formulier wordt toegezonden met het aantal cliënten en extra gegevens, deze moet door het systeem kunnen worden gegenereerd.	Nee, deze functie is komen te vervallen
C2	De instellingen binnen de applicatie moeten kunnen worden veranderd door een beheerder	Nee, onderdeel van subsysteem 4
C3	Gebruikers van het systeem moeten kunnen worden toegevoegd	Nee, onderdeel van subsysteem 4

In dit stadium is het mogelijk om de dagelijkse handelingen die een verloskundige verricht te ondersteunen. De systeemeisen die niet behaald zijn in dit stadium, met uitzondering van C1, worden gedekt door het vierde en tevens laatste subsysteem.

De systeemeis C1 is komen te vervallen na de beslissing om het laatste subsysteem niet te ontwikkelen in het kader van dit project.

Ondanks dat binnen het kader van dit project het vierde subsysteem niet zal worden uitgevoerd, kan door het niet behalen van de systeemeis C1 deze pilot als onsuccesvol afgerond worden beschouwd.

Wellicht zal in een later stadium, die buiten het kader van dit project valt, de vierde pilot alsnog worden uitgevoerd. Waarna het mogelijk wordt om voor de systeemeis C1 alsnog de nodige functionaliteiten te ontwikkelen.

Conclusie

In dit document is een ontwerp gemaakt van de functionaliteiten van de eerste drie subsystemen, zoals gedefinieerd inde definitiestudie.
Na de implementatie van iedere bouweenheid zijn de opgestelde testspecificaties uitgevoerd. De resultaten hiervan wijzen erop dat functionaliteiten naar verwachting opereren. Uit de controle op de dekking van de systeemeisen is gebleken dat in dit stadium niet wordt voldaan aan de gestelde eisen.

De functionaliteit “LVR-Controle” is komen te vervallen, doordat het laatste subsysteem niet meer zal worden uitgevoerd binnen het kader van dit project. Hierdoor is de volgende systeemeis niet gedekt.

- De LVR (Landelijke Verloskunde Registratie) vereist dat er periodiek een formulier wordt toegezonden met het aantal cliënten en extra gegevens, deze moet door het systeem kunnen worden gegenereerd.

Wellicht zal in een later stadium de vierde pilot alsnog worden uitgevoerd. Waarna het mogelijk wordt om voor de systeemeis alsnog de nodige functionaliteiten te ontwikkelen.

Bijlage A: Plan van aanpak pilot 4 ‘Subsystemen één tot en met drie’

Inleiding

In dit document worden de onderdelen van de vierde pilot toegelicht en in een planning opgenomen. Dit plan van aanpak is een detailplanning voor de vierde pilot en toevoeging op het plan van aanpak voor het gehele project.
Allereerst zullen de resultaten die de pilot oplevert worden beschreven en vervolgens zal een planning gemaakt worden voor de uitvoering van de pilot.
Voor informatie over kwaliteitsborging en de projectorganisatie wordt verwezen naar het plan van aanpak voor het gehele project.

Producten en activiteiten pilotontwikkeling

Hieronder volgt een overzicht van de activiteiten die worden uitgevoerd tijdens de ontwikkeling van de eerste pilot. Per activiteit worden deelactiviteiten beschreven en er wordt aangegeven wat het resultaat is van deze activiteit.

- Specificeren globaal-functionele structuur pilot
    - Bepalen functionele reikwijdte pilot;
    - Resultaat: ontwerp van functionele structuur.
- Opstellen pilotontwikkelplan
    - Definiëren pilotdelen
    - Definiëren bouweenheden
    - Voorbereiden beoordeling en testen pilotdelen;
    - Resultaat: ontwikkelplan voor pilot.
- Ontwerpen software-bouweenheden
    - Specificeren integratie en testen bouweenheden
    - Opstellen detailspecificaties
    - Opstellen testspecificaties
    - Resultaat: gedetailleerd ontwerp pilotdelen.
- Bouwen software-bouweenheden
    - Coderen pilotdelen
    - Voorbereiden component-test
    - Uitvoeren component-test
    - Corrigeren component
    - Resultaat: implementatie van pilot.
- Opstellen invoeringsprocedure pilot
    - Ontwerpen invoeringsprocedure
    - Specificeren tests invoeringsprocedures
    - Testen invoeringsprocedures
    - Corrigeren invoeringsprocedures
    - Resultaat: beschrijving invoeringsprocedure.
- Beoordelen en testen pilot
    - Opzetten testomgeving
    - Beoordelen en testen pilot
    - Analyseren uitkomsten

- Corrigeren fouten
- Rapporteren over beoordelen en testen

Naast deze stappen zijn een aantal volgens IAD voorgeschreven ontwikkelstappen bewust overgeslagen. Deze stappen zullen geen toegevoegde waarde leveren op het ontwerp of de ontwikkeling van de pilot.

De overige activiteiten die IAD voorschrijft worden niet voor dit product uitgevoerd.

- Voorbereiden pilotontwerp-workshop
  Een workshop voor de pilot is reeds gehouden in de definitiefase.
- Specificeren globaal-technische structuur pilot
  De technische structuur van de pilot is onveranderd sinds de laatste pilot.
- Specificeren globaal-organisatorische inrichting
  De invloed op de organisatorische aspecten zijn beschreven in de definitiefase. Deze pilot levert geen extra veranderingen met zich mee..
- Aanpassen externe componenten
  Er hoeven geen externe componenten te worden aangepast.
- Wijzigen andere informatiesystemen
  Er is geen sprake van een ander informatiesysteem.
- Bouwen conversie-tools
  Deze pilot vereist geen conversie-tools
- Ontwerpen handmatige procedures
  De handmatige procedures zijn hier niet van toepassing.
- Integreren bouweenheden
  De bouweenheden worden bij implementatie direct geïntegreerd, doordat deze onderdeel zijn van de applicatie.
- Samenstellen opleidingsmateriaal pilot
  Er is geen behoefte aan opleidingsmateriaal voor enkel deze pilot.
- Samenstellen handleiding pilot
  Deze deelactiviteit is opgenomen als laatste bouweenheid van deze pilot

<u>Planning</u>
Hieronder volgt een planning van de uit te voeren activiteiten.

Product / Proces	Tijdsduur in uren
Specificeren globaal-functionele structuur pilot	70
Opstellen pilotontwikkelplan	8
Ontwerpen software-bouweenheden	43
Bouwen software-bouweenheden	56
Opstellen invoeringsprocedure pilot	4
Beoordelen en testen pilot	2
Totaal aantal uren	183