Afstudeerverslag

O.N.C. Easy PHP Script

Pijnacker, 17 oktober, 2004

Student:	C. Zimmerman
Studentnummer:	99007743
Examinatoren:	V.H.F. Hermans R.A Mantel
Opdrachtgever:	P. d'Anjou

REFERAAT

Trefwoorden:

Apache, Database, HTML, MySQL, Oranje Nassau College Zoetermeer, Intranet, PHP, Scripttaal, SQL, Webserver.

INHOUDSOPGAVE

1 INLEIDING

Dit verslag is geschreven door Coen Zimmerman, vierdejaars informaticastudent aan de Haagse Hogeschool in de richting IVIT4 (voorheen IKS). Het beschrijft de ontwikkeling van een scripttaal en de ontwikkelomgeving voor een systeembeheerdatabase voor het Oranje Nassau College te Zoetermeer, in de afstudeerperiode van mei 2004 tot en met oktober 2004.
Met het verslag wordt beoogd om inzicht te geven in de aanpak en uitvoering van het afstudeerproject.

Legenda
De meeste lezers zullen bekend zijn met gangbare ICT-vaktermen, modellerings-technieken en programmeertalen. Voor de overigen is in hoofdstuk 5 een verklaring opgenomen van een aantal termen en begrippen die in dit verslag zijn gebruikt.

Bijlagen
Bij dit verslag horen de volgende bijlagen:
- Plan van Aanpak, op basis waarvan de opdracht is uitgevoerd.
- Het definitiestudie-document.
- Het scripttaal-syntaxis ontwerpdocument.

Deze bijlagen zijn volledigheidshalve gegeven, al zijn ze niet noodzakelijk voor begrip van het verslag. Daarom, en om het verslag binnen hanteerbare grenzen te houden, zijn de bijlagen in een aparte band toegevoegd.

2 OMSCHRIJVING OPDRACHT

2.1 OPDRACHTGEVER, SITUATIE

Het Oranje Nassau College (ONC) is ontstaan in 1995 uit een fusie van het Oranje Nassau College en het Willem van Oranje College. Deze christelijke scholengemeenschap bestaat uit drie schoollocaties en een kantoorlocatie waarin de centrale directie is gehuisvest, alle in Zoetermeer.
De drie locaties bieden samen de onderwijsvormen van VMBO tot en met het VWO-gymnasium en er wordt lesgegeven aan ongeveer 2900 leerlingen.

Het ONC is een ICT-voorhoede school. Binnen diverse vakken speelt de ICT een belangrijke rol in de lesgeving, en de leerlingen worden gestimuleerd in het gebruik van internet en Kennisnet. Er is ook een leercentrum waar zij kunnen werken met gespecialiseerde technische software, zoals CAD/CAM toepassingen.

2.2 PROBLEEMSTELLING

De afdeling systeembeheer heeft onvoldoende gereedschappen voor registratie en beheer van de door het ONC gebruikte hard- en software, de storingsmeldingen, genomen acties, etc., hetgeen bij het groeiende ICT-gebruik als een gaandeweg groter probleem wordt herkend. Er wordt nu gebruik gemaakt van een op Access gebaseerd databasesysteem, maar dat wordt als onoverzichtelijk en niet gebruiksvriendelijk ervaren en is niet vanaf iedere locatie aanroepbaar.

Het ONC heeft onvoldoende programmeerkennis in huis en te weinig beschikbare mantijd om de bestaande programmatuur aan te passen of een nieuw systeem te ontwikkelen. Ook ontbreken de financiële middelen om de aanpassing of ontwikkeling in opdracht te geven aan een softwarehuis. Het huidige beheersysteem is overigens niet of nauwelijks gedocumenteerd, hetgeen de aanpassing ervan ernstig zou bemoeilijken.

2.3 GEKOZEN OPLOSSING

In overleg met ONC is gekozen voor de ontwikkeling van een nieuw beheersysteem, gebaseerd op een relationele database en zodanig opgezet dat het gegevensbeheer eenvoudig in eigen beheer kan worden aangepast en uitgebreid.
Een essentieel onderdeel hiervan is de ontwikkeling van een robuuste scripttaal waarmee op relatief eenvoudige wijze databaseaanpassingen kunnen worden uitgevoerd, en invoerschermen, rapporten en andere databasegerelateerde routines worden gemaakt.

De aanschaf van een kant-en-klaar product is uitgesloten. De precieze toepassingen staan nog niet vast en de benodigde aanpassingen ervan zouden te ingewikkeld zijn voor de ONC-systeembeheerders, en te duur om uit te besteden.

2.4 UITGEVOERDE OPDRACHT

In deze opdracht heb ik de gewenste systeembeheerdatabase en een PHP-gebaseerde scripttaal geschreven, waarmee alle noodzakelijke database- en andere functies op een simpele en consistente wijze gegenereerd en gebruikt kunnen worden. Tevens heb ik een oplossingsbepalende aanzet gegeven tot de implementatie van dit systeem.

2.5 AANPAK

2.5.1 Uitvoeringsfasen

Voor de opdrachtuitvoering ben ik uitgegaan van de volgende drie fasen: definitiestudie, ontwikkeling en implementatie.

Definitiestudie
In deze eerste fase heb ik, na uitgebreide interviews met de systeembeheerders en overleg met de opdrachtgever, vastgesteld aan welke eisen het nieuwe beheersysteem (ook wel genoemd het *systeem* of *product*) zou moeten voldoen. Daarnaast heb ik de verschillende ONC-systemen en -toepassingen, waaronder de huidige Access-beheer-toepassing, onderzocht om een goed beeld te krijgen van het kader waarin het product zou moeten functioneren.
Op basis van de bevindingen heb ik een concept-pilotplan geschreven waarin ik de opzet van het produkt met de benodigde pilots, en het ontwikkeltraject heb vastgelegd. Ik heb daarbij besloten het produkt op te delen in de bekende categorieën: basis, comfort en luxe, zodat ik de gebruikelijke ontwikkelvolgorde kon toepassen.

Ontwikkeling
Ik ben deze fase begonnen met het stuk voor stuk uitwerken van de pilots. Voor elke pilot heb ik een detailconcept geschreven en dit toegevoegd aan de definitiestudie. Vervolgens heb ik voor de pilots, op basis van de detailconcepten, de code geschreven. De referentie-documentatie is gelijktijdig met de code gemaakt, zodat elke functie meteen al beschreven was. De handleidingen voor gebruik en beheer worden pas na de voltooiing van de programmatuur geschreven, zodat deze overeenstemmen met het uiteindelijke product.

Implementatie
De kern van het beheersysteem, met database, parser, de meest belangrijke scripts e.d, zal ik op details nog aanpassen en completeren, waarna ik het systeem, met documentatie, systeemcode en andere programmatuur, als een compleet pakket aan de opdrachtgever zal overhandigen.

Met de nog door het ONC toe te voegen scripts zal het systeem vervolgens worden geïnstalleerd en, na inventarisatie en opname van de gegevens in de database, in gebruik worden genomen.
Met de opdrachtgever is afgesproken dat ik dit proces zal begeleiden en ondersteunen.

2.5.2 Methoden en technieken

Ontwikkelmethode

Ik heb gekozen voor de ontwikkelmethode IAD, vooral vanwege de goede ervaringen met IAD bij integrerende practica op de Haagse Hogeschool. Hoewel deze methode vaak meer werk heeft gekost dan vereist voor een succesvol project, is het gebruik ervan mij goed bevallen. Het is mij opgevallen dat IAD een prima voortgangscontrole biedt. De iteratieve aanpak zorgt voor een duidelijk stappenplan, waarmee een project prettig in onafhankelijk te maken modulen opgedeeld kan worden.

Ik heb niet strikt volgens de voorschriften van IAD gewerkt, maar hiervan afgeweken waar ik het in dit project logisch en verantwoord vond. Zo schrijft IAD bijvoorbeeld een verificatie- en validatiefase per pilot voor. Voor dit project leek het mij verstandig de pilots klein te houden, maar niet voor elke losse pilot een volledige testfase uit te voeren, dit om de vaart erin te houden en de 'papierwinkel' te beperken. Hierbij heeft de onderlinge afhankelijkheid van de pilots een rol gespeeld. Wanneer de werking van een pilot afhankelijk is van een voorgaande pilot, dan heb ik deze uiteraard wel eerst gevalideerd. Ook de 'workshop' technieken heb ik achterwege gelaten omdat deze alleen nuttig zijn bij omvangrijke gebruikersgroepen. Wel heb ik goed gebruik gemaakt van de zogeheten 'Juicy Bits First' (JBF) aanpak. Deze strategie heeft de ontwikkelvolgorde van de pilots en systeemeisen bepaald.

3 UITVOERING OPDRACHT

3.1 DEFINITIE

3.1.1 Werkwijze

Aanpassing
Tijdens de uitvoer van het project is de Definitiestudie steeds bijgewerkt met de aanpassingen op concepten, het pilotplan etc., zodat het document steeds consistent met het product en het plan is gehouden. Uitzondering op deze regel zijn de systeemeisen, die ik niet in volgorde of nummering heb aangepast om verwarring te voorkomen.

In de loop van het project zijn keuzes en veranderingen in de prioritering, planning of afbakening steeds gemaakt in overleg met goedkeuring van de opdrachtgever.

3.1.2 Interviews

Interviews en contactmomenten
In de eerste vier weken van het ontwikkeltraject heb ik zesmaal een interview met de opdrachtgever gehouden. Het plan was aanvankelijk minstens één gesprek per week te houden, om zo de voortgang te controleren. Ik had al verwacht dat dit aantal tijdens de definitiestudie wat hoger zou liggen dan tijdens de ontwikkelfase. Dit is niet meer dan logisch, omdat vooral in deze eerste projectfase veel informatie wordt verzameld.

Aanpak van de interviews
De interviews bij de opdrachtgever heb ik volgens een vast patroon gevoerd. Tevoren heb ik de vragen verzameld en in een logische volgorde gegroepeerd, de vragen uitgewerkt in hele zinnen en de groepen en vragen genummerd.
Tijdens de interviews heb ik deze lijsten vraag voor vraag afgewerkt. Hierbij ben ik soms van de volgorde afgeweken wanneer dit in het gesprek logisch was. Wanneer ik de indruk had dat een vraag niet goed werd begrepen, maakte ik de implicaties ervan met een voorbeeld duidelijk.
Door de gekregen antwoorden en de gemaakte afspraken zorgvuldig terug te koppelen tijdens de interviews heb ik zeker gesteld dat de informatie en inzichten wederzijds goed zijn begrepen.
Na de interviews heb ik nog diezelfde dag de behandelde vragen, conclusies en verdere gespreksresultaten uitgeschreven in een Word-document en dit naar de opdrachtgever gestuurd. Op deze manier hadden wij beiden een volledig beeld van de besproken stof en een laatste verificatie van het wederzijds begrip.

Nadat ik het eerste interview op deze wijze had uitgevoerd en vastgelegd, heb ik de opdrachtgever gevraagd wat hij van de interviewaanpak vond. Hoewel hij half serieus grapte over de hoeveelheid papierwerk, was hij wel degelijk te spreken over deze aanpak, zodat ik deze tijdens het project heb volgehouden.

De gesprekken met de opdrachtgever en de beheerders zijn soepel verlopen. Ik kan het goed vinden met de opdrachtgever, en met zijn technische expertise en achtergrond

was het geen probleem om bijvoorbeeld de wat technischer aspecten van bepaalde keuzes te bespreken. Ook was het niet lastig te achterhalen wat er precies geëist werd; de opdrachtgever wist in de meeste gevallen precies wat hij wilde. In twijfelgevallen hebben we de mogelijkheden aan de hand van onze ervaring besproken: in zijn geval vooral beheertechnisch en gericht op het onderwijs, in mijn geval vooral op het gebied van programmering, webdesign en intranetapplicaties.

Interviews inhoudelijk
Bij de eerste interviews heb ik me vooral gericht op de globale eisen en de afbakening van het project, vooral om te zien of het beeld dat ik me van het benodigde systeem had gevormd wel overeen kwam met bij de werkelijkheid. Daarnaast heb ik achterhaald hoe de technische omgeving eruit zag en welke programmatuur ik het beste zou kunnen gebruiken.
De organisatorische aspecten van het project zijn in de interviews vrijwel niet meer aan bod gekomen omdat deze al tijdens de opdrachtomschrijving waren vastgesteld. Slechts in specifieke gevallen, zoals de gebruikersrechten van leerlingen of storingsmeldingen, is hierop doorgegaan. De interviewvragen waren daarom hoofdzakelijk technisch van aard.
Naarmate de globale vorm van het project duidelijker werd, spitsten de interviews zich meer toe op details. Vooral vanaf het moment dat ik de eerste opzet van het systeem-concept aan de opdrachtgever heb getoond (interview 4) zijn de vragen duidelijk technisch specifieker geworden.

Ervaringen op de Haagse Hogeschool, maar ook daarbuiten, hebben mij geleerd dat het in algemene zin vragen naar de 'eisen' vaak een verkeerd beeld oplevert van wat er werkelijk gewenst is. Het is voor een opdrachtgever moeilijk om 'vanuit het niets' een complete en juiste opsomming te geven wat er precies moet gebeuren; als deze daarentegen wordt benaderd vanuit een specifiek kader en met de juiste manier van doorvragen, worden veel betere resultaten verkregen. Ik heb dan gerichte vragen met betrekking op bepaalde systeemdelen of situaties gesteld en ben zo, met de nodige terugkoppeling, gaandeweg tot een compleet en correct eisenpakket gekomen.

3.1.3 Vooronderzoek

Bestaand systeem
Het huidige systeem bestaat uit een Access database met een specifieke gebruikers-interface. Access leent zich echter niet voor deze toepassing. De database kan niet door meerdere gebruikers en vanaf alle locaties worden aangesproken en ondanks de beschikbare 'wizards' is het aanpassen en uitbreiden te ingewikkeld.
De inhoud van de database kon al dan niet worden overgenomen uit het bestaande systeem. Voor het Legermuseum Delft heb ik (gedurende mijn stageperiode) een aantal soortgelijke bestanden op deze manier verwerkt. Het kostte mij destijds enkele dagen programmeerwerk om een zeer omvangrijk gegevensbestand te converteren.
Daarnaast had de opdrachtgever nog niet besloten of hij de bestaande gegevens wel over wilde zetten. Ik wilde de opdrachtgever niet voor het blok zetten voordat deze een bruikbaar prototype van de invoerschermen had gezien, omdat dit een goed beeld zou geven van de hoeveelheid werk die het invoeren zou vergen.
Uiteindelijk is besloten de gegevens niet over te zetten, de opdrachtgever wil de hard- en software inventariseren en de inhoud van het nieuwe systeem hierna invullen.

Fronter
Ik heb Fronter bekeken, een zogeheten "E-learning toepassing", gericht op kennis-uitwisseling en samenwerking. Fronter wordt op het ONC gebruikt en komt in aanmerking voor onderzoek omdat het een internet- dan wel intranet-gebaseerd content-management systeem is. Het biedt een omgeving waarin leerlingen en docenten gebruik kunnen maken van gedeelde documenten, agenda's en archieven.
De beheerders- en docentenkant van Fronter had mijn interesse, omdat deze qua gebruikersrechten en functionaliteit het meest overeenkwam met de wensen van de opdrachtgever. De opzet van het systeem is echter te inflexibel gebleken, en is teveel gericht op het gebruik door docenten en leerlingen, en niet op de meer algemeen inzetbare beheertaken.

Principekeuze gebruikersinterface en database
De opdrachtgever heeft op dit vlak zeer uitgesproken eisen gesteld, met als uitgangs-punt dat het systeem op alle ONC-locaties en op elke LAN-PC - mits geautoriseerd - aan te spreken moet zijn.
Wij zijn het er ook snel over eens geworden dat de enige plausibele vervanging voor het huidige systeem zou bestaan uit een database, aangesproken door een webgebaseerde interface. Dat een database gebruikt zou worden, spreekt voor zich; het gaat om een uitgebreide set van gegevens, die voortdurend aan verandering onderhevig zijn. Alleen databases zijn voldoende flexibel en zijn in de regel prima benaderbaar door de gewenste interface.
De keuze voor een webinterface ligt voor de hand, ondanks de bekende nadelen aan het gebruik hiervan: men is gebonden aan het gebruik van browsers, de mogelijkheden van invul- en selectievelden zijn beperkt, en het schrijven van dynamische programma's is ingewikkelder omdat er alleen tijdens het laden van een (intranet-) pagina contact kan worden gemaakt met de database. Dit maakt het schrijven van interfaces wel lastiger, maar hier staat tegenover dat een webinterface die beschikbaar wordt gesteld op een centrale server, toegankelijk is voor iedere computer in het netwerk zonder dat er installaties voor moeten worden verricht. Al met al geven de voordelen hier de doorslag.

Keuze programmeertaal
Al vroeg in de definitiefase heb ik voor de te gebruiken programmatuur een keuze gemaakt uit drie talen die geschikt zijn voor het maken van internet- en intranet-applicaties, namelijk Perl, PHP en ASP.
Perl viel af door een nogal gecompliceerde syntax (zelfs lastig in het gebruik voor de gevorderde programmeur) en biedt geen bijzondere voordelen in deze toepassing.
In de vergelijking met ASP hebben mijn ervaringen met PHP de doorslag gegeven. De C-gelijkende syntaxis, platformonafhankelijkheid en de wijze waarop de taal aansluit bij verschillende webservers en databasemanagementsystemen, zijn mij altijd prima bevallen en maken PHP voor toepassingen als deze zeer geschikt.
De opdrachtgever had geen voorkeur, zolang er maar geen gebruik zou worden gemaakt van een obscure en slecht ondersteunde ontwikkelomgeving. Bij ASP en PHP is dit niet aan de orde; beide worden door uitgebreide, actieve sites en gebruikers-groepen op het internet ondersteund en besproken.

Keuze database
Voor de database heb ik als criteria gehanteerd: eenvoudig, snel, goedkoop, uitgebreid, nog steeds in ontwikkeling en krachtig genoeg voor de toepassing, rekening houdend met realistische groeiverwachtingen.
De keuze viel al snel op MySQL. In de vier jaar dat ik bekend ben met dit systeem is er volop aan gewerkt en is MySQL een volwaardig systeem geworden. Hiervoor geldt dus hetzelfde als voor de gekozen programmeertaal.
Krachtigere en veel duurdere DBMS'en als Oracle of MSSQL Server zijn vooral nuttig voor gecompliceerde, omvangrijke databanken en datawarehousing, maar voor eenvoudig content management en typische intranettoepassingen zijn deze systemen veel te uitgebreid. De opdrachtgever heeft mij ook te kennen gegeven dat projecten waarvoor zo'n DBMS nodig zou zijn, niet in de ICT-plannen van het ONC passen. Hij heeft zich dan ook geheel verenigd met de keuze voor MySQL.

Keuze scripttaal
Door de opdrachtgever zijn enige belangrijke uitgangspunten voor het beheersysteem gesteld:

- Het systeem moet gemakkelijk en in eigen beheer aangepast en uitgebreid kunnen worden.
- Het systeem moet opgenomen kunnen worden in het nog te ontwikkelen ONC-intranet.

Dat het systeem eenvoudig aan te passen moet zijn, is een logische eis. Een school is een dynamische omgeving met steeds nieuwe gebruikers van systemen; de docententeams en natuurlijk ook de klassen en het leerlingenbestand veranderen jaarlijks en daarmee ook de eisen, wensen aan en ideeën over ICT-toepassingen in de organisatie. Het is dus niet handig een star geprogrammeerde applicatie in te zetten, zeker niet in een vroeg stadium van het gebruiksproces.

De geplande intranetontwikkeling legt nadruk op de behoefte aan goed aanpasbare applicaties. Met dat in gedachten heb ik de volgende applicatiemogelijkheden overwogen:

a. Een applicatie waarin met menu's, invulvelden en voorbeeldlijsten een (intranet-) pagina wordt gegenereerd.
b. Een scripttaal waarmee pagina's worden gedefinieerd
c. Een set aanpasbare voorbeeldpagina's, die gemakkelijk kunnen worden aangepast voor verschillende gebruikssituaties.
d. Een set opmaak-templates en een interactieve cursus voor het gebruik van een webscripttaal (zoals ASP of PHP) die hiermee te gebruiken is.

Oplossing d. is onmiddellijk afgevallen, omdat hiervoor een uitgebreide programmeertaal moet worden aangeleerd, hetgeen de opdrachtgever juist wil vermijden.
Hetzelfde gold voor optie c. Hoewel het mogelijk is om in zulke applicaties duidelijk aan te geven waar aanpassingen kunnen worden gemaakt, brengt het veel beveiligingsrisico's met zich mee. Daarbij is het oplossen van fouten in een taal als ASP of PHP vrijwel onmogelijk zonder voldoende kennis van programmeren in het algemeen en van de gekozen taal in het bijzonder.

Na verder breinstormen met de opdrachtgever is oplossing a. afgevallen omdat een ontwikkelsysteem met veel invoerschermen lastig uit te breiden is, moeilijk overzichtelijk te houden, en ongemakkelijk in het gebruik, doordat de vele invoerschermen veel interactiviteit met de gebruiker vereisen.

Hiermee viel de keuze op b.: een scripttaal. Een goed ontworpen scripttaal is zeer flexibel en krachtig in het gebruik en eenvoudig aan te leren. De genoemde beveiligingsrisico's en problemen met het opsporen van fouten kunnen erin worden ontweken. De taal is eenvoudig te gebruiken voor de leek en nog steeds krachtig in handen van een expert. Het zou een prima tussenstap kunnen vormen naar het eventueel zelf ontwikkelen van programmatuur binnen het ONC, omdat men precies kan zien hoe een simpel scriptcommando wordt vertaald naar systeem-broncode.

3.1.4 Systeemeisen

In de eerste weken van de definitiestudie heb ik mij vooral gericht op het inventariseren en uitwerken van de systeemeisen. Dit was nodig omdat het systeemconcept en de pilots gebaseerd moeten zijn op een volledige en correcte eisenlijst.
Ik heb er ook op gelet dat de gestelde eisen binnen de perken zouden blijven door ze te toetsen aan de afgesproken afbakening van het project; Ik ken het probleem van de "opstapelende wensen". Gelukkig heb ik ook wat dit betreft steeds op één lijn met de opdrachtgever gezeten, waardoor het afwegen en kiezen van eisen en prioriteiten zonder problemen is verlopen.

Ik heb de definitieve eisen verzameld en in volgorde van prioriteit voor de opdrachtgever gegroepeerd en genummerd. Ik heb verder de eisenlijst verdeeld in de eisen gerelateerd aan de scripttaal en de opzet van het systeem, en de eisen met betrekking tot hetgene dat hiermee gemaakt zou gaan worden.

De eisen aan het informatiesysteem voor de systeembeheerders bleken relatief eenvoudig te verwerken; de typische database-eisen en wensen komen tenslotte in talrijke systemen aan de orde en zijn daarom niet nieuw voor me.
De eisen aan de scripttaal spraken echter minder voor zich. Ik heb vooral de te vage eisen uit de lijst geweerd, omdat die lastig te concretiseren zouden zijn. Ook sommige, erg specifieke eisen heb ik weggelaten, omdat deze niet bij de werkelijkheid zouden aansluiten.

Alle eisen aan de scripttaal zijn niet vanaf het begin bekend geweest. Hiertoe moeten namelijk ook de eisen aan de producten die met de taal geschreven zouden worden, precies bekend zijn. De eerder genoemde eenvoud van deze producten houdt niet in dat de eisen eraan onveranderlijk zijn. Omdat het de bedoeling is dat de taal juist gebruikt wordt voor latere uitbreiding en aanpassingen, is het scriptsysteem zo opgezet dat het flexibel in gebruik en eenvoudig aan te passen is.

Om te voorkomen dat ik door deze beperkte systeemeisenlijst voor problemen zou komen te staan, bijvoorbeeld door de complexiteit van een functie te onderschatten, heb ik steeds de scriptfuncties vergeleken met de taalelementen van bestaande (script)talen, zoals PHP, BASIC en mIRCScript. De opzet van deze talen en mijn programmeer-ervaring zijn een goede ondersteuning bij dit project geweest.

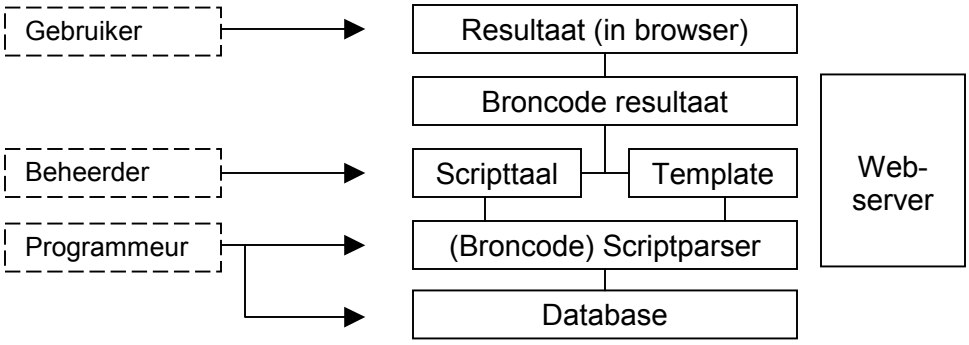
3.1.5 Systeemconcept

3.1.5.1 Globale aanpak

Het systeemconcept is gebaseerd op de gestelde systeemeisen en bestaat uit twee delen:

- *Het scriptsysteem.*
  Dit is het gereedschap waarmee de andere systeemdelen worden gemaakt. Binnen dit systeem vallen de scriptparser en de bijbehorende interface. Het voorziet geheel in de eisen met betrekking tot de aanpasbaarheid en flexibiliteit van het systeem.
- *Het beheersysteem.*
  Met dit systeem worden de gegevens (zoals hard- en software, gebruikers en nieuwsberichten) vastgelegd en bewerkt. Het systeem wordt gebouwd met het scriptsysteem en voorziet in de gestelde functionele eisen.

3.1.5.2 Hiërarchische structuur van het systeem



Met deze schematische weergave van de gegevensverwerking heb ik de gelaagdheid van het systeem aangegeven, gezien vanuit de verschillende rollen van personen die met het systeem zouden werken:

- Gebruikers kunnen alleen het resultaat van de scriptparser in hun webbrowser-sessie zien. De gegevens zijn dus pas vanaf een bepaald punt, namelijk de omzetting naar HTML broncode, in een browser op te vragen.
- Beheerders krijgen toegang tot de scripts en templates.
- Systeemprogrammeurs kunnen bij de meest basale elementen van het systeem, namelijk de code van de scriptparser zelf en de database.

In het diagram zijn de directe koppelingen tussen de systeemdelen met verbindingslijnen aangegeven. Zo is de scriptparser met de database verbonden, omdat deze de gegevens hier direct uithaalt. Het resultaat dat de webserver naar de browser van een gebruiker stuurt, heeft een directe koppeling met het beeld dat deze gebruiker vervolgens ziet.
In een apart blok, "Webserver", heb ik aangegeven hoe de webserverapplicatie de verschillende gegevens overkoepelt. Alle stappen, van het draaien van de scriptparser tot de presentatie van de gebruikerspagina's, vinden dus op de webserver plaats.

3.1.5.3 Databaseontwerp

Klassendiagram
Bij dit project heb ik voornamelijk klassendiagrammen gebruikt voor het modelleren van de gegevensstructuur. Het gebruik van klassen voor de representatie van systeem-elementen en gegevenssets maakt het verhalen van het globale concept naar een databaseontwerp zeer eenvoudig. In de meeste gevallen zou zelfs de één-op-één vertaling naar een databasetabel met bijbehorende attributen mogelijk moeten zijn.
Met het klassendiagram zijn ook de verbanden tussen de klassen duidelijk aangegeven, zodat deze direct vertaalbaar waren naar de relaties in de relationele databaseomgeving.

Als eerste stap naar het systeemconcept heb ik de kandidaatklassen verzameld uit verschillende documenten en gespreksinformatie. Na filtering hiervan (redundante, te vaag benoemde en irrelevante klassen) bleven de klassen over die in aanmerking kwamen voor opname in het klassendiagram.

Gezien de voornaamste functie van de database: de opslag van alle gebruikte soft- en hardware binnen het ONC, lagen de klassen als software en hardware voor de hand. Verder zijn klassen opgenomen die in eerdere brainstormsessies met de systeem- en applicatiebeheerders zijn benoemd.

Met de geselecteerde klassen heb ik een klassendiagram met de nodige verbindingen gemaakt (dit alles volgens de UML standaarden) en de attributen aan de klassen toe-gevoegd. Het klassendiagram is bijgewerkt gehouden gedurende de hele definitie-studiefase.
Omdat het klassendiagram genoeg inzicht geeft voor de te maken database, heb ik ervan afgezien om een meer gedetailleerd ERD (Entity Relationship Diagram) te maken.

Het databaseontwerp, gebaseerd op het klassendiagram, bestaat uit een relationeel representatiemodel waarin ik duidelijk de primaire en vreemde sleutels heb aangegeven, en een implementatiemodel voor invoer in de MySQL database.

3.1.5.4 Scriptsysteem

Bij het scriptsysteem is geen sprake van een typische database, zodat ik hiervoor geen klassendiagram heb opgesteld. Als er al databasetabellen zouden worden gebruikt, zou de toepassing hiervan beperkt zijn tot de opslag van instellingen en andere secundaire gegevens. Daarbij komt dat de opdrachtgever voor de scripts, applicatie-instellingen e.d. liever zou werken met tekstbestanden dan met een database en bijbehorende interface.

In het definitiestudie-document heb ik, waar zinvol, de aanpak en de gemaakte keuzen verklaard. Dit heb ik ook gedaan voor de gekozen alternatieve aanpak van het script-systeem.
Ik heb de techniek achter het systeem, namelijk de bestand- en directorystructuur, systeemhiërarchie en dergelijke, vooral aan de hand van schematische weergaven en diagrammen in kaart gebracht.
Waar sprake is van typische applicatie-elementen, zoals de interface van de parserapplicatie of back-upgereedschap, heb ik hiervoor navigatiediagrammen, use-cases en andere meer gestandaardiseerde UML modellen gebruikt. Voor de meer ingewikkelde functies in de code van de parser heb ik gebruik gemaakt van pseudo-code.

Modellering scripttaal
De scripttaal heb ik beschreven zoals dit voor programmeertalen op sites (PHP) en in boeken (BASIC, C) wordt gedaan. De syntaxis heb ik beschreven aan de hand van taalelementen als operators, statements en functies, met duidelijke voorbeelden van de beschrijving van de taalregels. Hiermee is de script-documentatiestructuur ook voor programmeurs herkenbaar geworden.

Run-time uitvoeren of compileren
Het vertalen van scripts naar uitvoerbare broncode (van scripttaal naar PHP) kan in principe gebeuren op twee manieren: run-time parsing en compilatie. In het eerste geval wordt een script pas tijdens de gebruikersaanroep in PHP-code vertaald, in het tweede geval moet het script door de beheerder zijn gecompileerd, voordat de gebruikers het betreffende script kunnen aanspreken.

De voor- en nadelen van beide methoden:

Run-time parsing	Vooraf compileren
Voordelen: • Aanpassen scripts heeft onmiddellijk resultaat. Het testen van de scriptwerking is eenvoudiger.	*Voordelen*: • De beheerder bepaalt wanneer scripts beschikbaar zijn voor de gebruikers. • De traagheid bij het compileren is alleen merkbaar voor de beheerder.
Nadelen: • Opvragen systeempagina's is vertraagd (het script wordt dan immers verwerkt). • Scriptfouten zijn gelijk zichtbaar voor de gebruiker.	*Nadelen*: • Telkens wanneer een aanpassing wordt gemaakt in een script, moet het eerst gecompileerd worden voor het kan worden gebruikt.

Om een verantwoorde keuze mogelijk te maken, heb ik beide methoden voor de opdrachtgever beschreven en in concept uitgewerkt.

Hierbij heb ik aangegeven dat de gekozen methode bepaalt in welk systeemgedeelte de uitvoer van de scriptparser ligt, hetgeen belangrijke gevolgen heeft voor de beveiliging en voor de wijze waarop foutmeldingen worden gepresenteerd. Bij run-time parsing zijn deze foutmeldingen namelijk niet alleen zichtbaar voor de beheerders, maar ook voor alle gebruikers. Ook geeft run-time parsing een hogere serverbelasting, omdat scripts bij elke opvraag worden vertaald. De opdrachtgever heeft om deze redenen gekozen voor compileren.

Directory-structuur
In de definitiestudie heb ik een uiteenzetting gegeven van de directorystructuur van de bij het scriptsysteem behorende bestanden en van de plaatsing van bestanden op de webserver. De gebruikte directorynamen zijn in een tabel verklaard, dit om verwarring te voorkomen, omdat het hier directories betreft en geen termen zoals gedefinieerd in de modeldictionary, waarin juist klassennamen worden beschreven.

3.1.5.5 Concepten voor de systeemintegratie

Interface scripttaal en database
Ik heb elke tabel (met uitzondering van de koppelingstabellen, waarin dit niet van toepassing is), voorzien van een veld 'code', waarmee een tabelregel uniek te identificeren is. Dit vormt een onwrikbaar uitgangspunt voor de scripttaal, waardoor scripts automatisch elke databasetabel op dezelfde manier kunnen aanspreken.

Gebruikersinterface
Per iteratie van de bouwcyclus heb ik detailconcepten opgesteld en heb deze aan de definitiestudie toegevoegd.
Met use-cases heb ik de gebruikersinteractie verduidelijkt bij ondoorzichtige elementen van het systeem; in combinatie met de navigatiediagrammen is er zo een duidelijk beeld van het systeem ontstaan.
Met schermontwerpen heb ik de navigatieconcepten gevisualiseerd en getoetst; gedetailleerde versies hebben als grafisch ontwerp gediend, in een vorm die de opdrachtgever eenvoudig kon spiegelen aan zijn eigen ideeën.

3.1.5.6 Pilotplan

In het pilotplan, het slotgedeelte van het definitiestudiedocument, heb ik een precieze beschrijving gegeven van de verschillende pilots, met een korte beschrijving van de inhoud en de systeemeisen waaraan door de pilot invulling wordt gegeven.

De scriptparser vormt de basis voor het beheersysteem en is het meest complexe en omvangrijke deel ervan. De kernpilot was om die reden: een werkende scriptparser. Ik heb deze kernpilot nog in tweeën gesplitst om te voorkomen dat de iteratieslag te groot en langdurig zou worden.

In een "functiepilot" heb ik alle gedeelde functies verzameld. De noodzaak en relevantie van zulke functies is ontstaan uit de geplande functionaliteit van de verschillende kern- en systeempilots. Daarom heb ik de functiepilot - als enige - parallel aan de andere systeemdelen ontwikkeld.

Per pilot heb ik aangegeven of de inhoud ervan is geclassificeerd als *basis*, *comfort* of *luxe*. Uitgangspunt was dat in ieder geval de basisfuncties binnen de gestelde projecttijd zouden worden uitgewerkt. Comfort-pilots zouden het gebruikt sterk veraangenamen, maar zijn niet cruciaal voor de werking van het systeem. Luxe-pilots zijn niet vereist voor een succesvolle oplevering, maar wel gewenst.
Deze indeling sluit aan bij de aangegeven prioriteit per systeemeisengroep.

De pilotdocumentatie is tijdens de ontwikkeling van de pilots geschreven, zodat deze steeds zoveel mogelijk beschikbaar en up-to-date was.

3.2 SYSTEEMBOUW

3.2.1 Werkwijze bij het programmeren

Voor het programmeren van de eenvoudige functies en systeempagina's was het voldoende om de beginsituatie en het gewenste eindresultaat te beschrijven.
Bij de meer ingewikkelde modulen en routines, zoals het ontleden van script-expressies in de parser, heb ik de programmering in logische, 'behapbare' stappen verdeeld. Waar sommige van de stappen nog te ondoorzichtig bleken, heb ik deze nog verder in tussenstappen opgesplitst.

De leesbaarheid van de programmacode is mede bepalend voor een vlotte ontwikkeling en later voor het onderhoud. Ik heb daarvoor bij het programmeren steeds de volgende regels gehanteerd;

- De naamgeving van variabelen en functies, maar ook de spatiëring en structuur van de code is zoveel mogelijk consistent gehouden.
- Bij het gebruik van accolades is indentatie in de code gebruikt, waardoor functies en andere codeblokken duidelijk herkenbaar zijn.
- De namen van directories, bestanden, variabelen en routines zijn logisch, weerspiegelen de functie en inhoud en zijn niet te kort of te lang.
- Met het groeperen van gedeelde functies is de code overzichtelijk gehouden en is voorkomen dat de programmabestanden te groot worden.
- Alle functies, en interessante of bijzondere stukken code, zijn voorzien van een beknopte uitleg van de context en het doel van de code.
- Constante factoren en waarden die in de code worden gebruikt, hebben herkenbare namen in hoofdletters en worden niet als waarde, maar altijd aan de hand van deze namen gebruikt.

Testen
De code van routines en functies heb ik steeds na het schrijven geverifieerd, waarbij is gelet op mogelijke array-boundary overschijvingen, eindloze lussen en dergelijke.
De code van de parser en de werking van de scripts heb ik per pilot getest aan de hand van de systeemeisen, waarbij vooral is gezocht naar fatale fouten.
Er is zowel functioneel (*black-box*) als structureel (*white-box*) getest om te voorkomen dat fouten onontdekt zouden blijven door het niet beschouwen van de systeemwerking.

Per pilot heb ik een testrapport opgesteld, met daarin de resultaten per systeem en de gevolgen daarvan voor het verloop en de planning van het ontwikkeltraject.

3.2.2 Scripttaal, parser

De scripttaal is O.E.P.S. gedoopt: Het ONC Easy PHP Script.
De parser voor deze scripttaal is ontwikkeld op basis van de twee kernpilots en bestaat in essentie uit een PHP-programma dat scripts vertaalt naar PHP-code, met daarin de verwijzingen naar de gedeelde functies en routines (de basis van pseudo-code stappen 1, 2 en 4 in de definitiestudie).
De parser voert de volgende stappen uit:

1. initialisatie
2. script inlezen
3. script controleren op syntaxis en semantiek, en vertalen
4. resultaat wegschrijven

De controlestap, de belangrijkste parserfunctie, wordt uitgevoerd voor elke regel van het ingeladen script en kijkt naar de syntaxis en semantiek, zoals het gebruik van al dan niet toegestane sleutelwoorden en functies.
Bij het wegschrijven van de resultaten wordt een HTML template toegepast, waarmee de opmaak van de PHP pagina wordt bepaald en waarin met een eenvoudig herkenningsteken wordt aangegeven waar de scriptresultaten moeten worden geplaatst.

Tijdens de vertaling hebben de codeblokken, ingeleid en afgesloten door een sleutelwoord als IF of PRINT, hun eigen adresruimte waar tijdelijke gegevens over het blok zoals een tekstbericht of de voorkeursuitlijning ervan, worden bewaard.
Via deze blokken wordt gecommuniceerd met de database, worden tabellen afgedrukt met TABLE, en worden gegevens opgehaald of weggeschreven met DATA, alles op een manier die weinig tot geen SQL-kennis vereist.

Expressieafhandeling
Voor de controle en vertaling van scriptexpressies, die binnen de 3e parserstap worden uitgevoerd, heb ik een uitgebreide functie geschreven waarin taalementen, zoals variabelen en functies, worden geïdentificeerd en gevalideerd.

Scriptregels, als bijvoorbeeld:

```
FUNCTIE("parameter1", 50 + $getal, "parameter" + "3")
```

worden door de scriptparser gecontroleerd op aspecten als:

- Het gebruik van variabelen, die gedeclareerd moeten zijn voordat ze worden gebruikt in een expressie.
- Het gebruik van operators als '/' of '-'. Wanneer een - (minus) operator wordt toegepast op tekstuele informatie, komt hier geen zinnig resultaat uit. Wanneer tekst en getallen op deze wijze gecombineerd worden, zal de parser dit detecteren.
  Een bijzonder geval betreft de '+' operator, die wordt gebruikt voor optellingen maar ook voor het concateneren van tekstuele waarden. Omdat dit voor de parser twee verschillende opdrachten zijn en binnen de scripttaal niet, wordt hier scherp op gelet.

- De juiste afbakening van tekstwaarden met een dubbel trema ("), omdat verkeerd gebruik ervan vreemde foutmeldingen geeft als het systeem deze fouten niet zou afvangen.
- Of het aantal parameters van de FUNCTIE wel klopt, omdat een verkeerde notatie, en een tekort of teveel aan parameters het parserresultaat zou corrumperen.
- Het voorkomen van onmogelijke berekeningen, zoals deling door nul.

De routine, die recursief werkt in gevallen waar expressies in expressies voorkomen (zoals parameters van functies of gebruik van haakjes), logt de gevonden fouten en geeft een werkende PHP vertaling van de scriptregel als resultaat.

Scriptfuncties en de parser-applicatie
Het beheer en het gebruik van gedeelde functies in de scripts worden overigens geregeld via een apart bestand waarin van alle functies het datatype, de bijbehorende parameters en de functiecode zijn opgeslagen. Ik heb dit zo georganiseerd om de latere uitbreiding en toevoeging van functies te vergemakkelijken.

De parser rapporteert de gevonden fouten in een lijst van waarschuwingen en foutmeldingen waarin regelnummers, en uitleg of mogelijke oplossingen worden aangegeven.

Een dynamisch menu, een HTML raamwerk, een aantal opmaakbestanden en plaatjes vormen samen de mal waarin de uit scripts gegenereerde PHP bestanden worden geplaatst, waardoor zij een logisch geheel vormen en door de gebruiker aan te spreken zijn.

Rond de scriptparser heb ik de parsertoepassing geschreven, waarmee de scripts worden beheerd en de instellingen kunnen worden aangepast. Tevens kan hiermee de database worden beheerd met behulp van invoerschermen, waarbij steeds wordt gecontroleerd of al dan niet gegevens verloren zouden gaan bij bepaalde keuzes (waarvoor dan wordt gewaarschuwd).
Ik heb tevens de schermen geschreven waarmee scripts gemaakt en bewerkt kunnen worden. Deze waren niet van tevoren gepland maar zijn in de praktijk, tijdens de bouw- en testfase, onmisbaar gebleken voor de volledige participatie van de opdrachtgever.

Late uitbreidingswens
Tijdens de bouw van de kernpilots heeft de opdrachtgever de wens geuit de database op gebruiksvriendelijke en veilige wijze via de scriptparser-applicatie te kunnen aanpassen. In overleg hebben wij besloten om deze uitbreiding voorrang te geven op de comfortdelen van het beheersysteem.

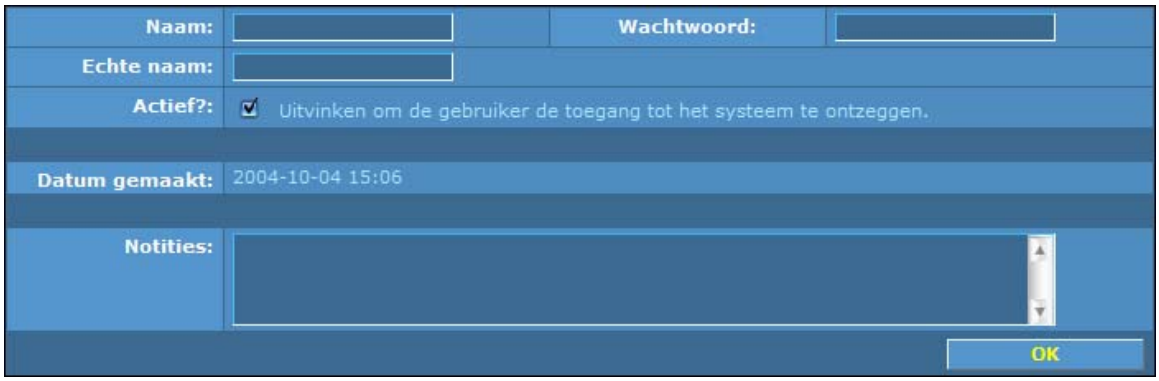
3.2.3 Scripts en het systeem

Gebruiker-sessies
Bij de aanmelding van een gebruiker worden diens naam en wachtwoord gevalideerd aan de hand van de gebruikersgegevens in de database. Vervolgens wordt een PHP-sessie gestart, waarin de gebruiker uniek geïdentificeerd wordt. Ik heb ervoor gezorgd dat bij elke paginawisseling de toegangsrechten van de gebruiker gecontroleerd worden om de afscherming ongevoelig te maken voor sessie-beveiligingslekken.

Ontwikkelde scripts
Voor bewerking van gebruikersgegevens, -groepen en -rechten heb ik scripts geschreven die dienen als voorbeeldscripts en als basis voor het beheersysteem. Tevens heb ik scripts geschreven voor het plaatsen en opvragen van verzoeken aan de afdeling systeembeheer, en voor het systeemlogboek en heb de daarvoor benodigde scriptfuncties geschreven.

In de elfde week van het project heb ik een aanpassing gemaakt in het pilotplan waarbij de onderwerpen en eisen van systeempilot 2, 3 en 4 zijn verschoven zodat de systeemdelen die eventuele parserconstructies en scriptfuncties vereisen eerder zouden worden geschreven.
Vervolgens heb ik de scripts geschreven voor het plaatsen, lezen en bewerken van berichten.

Invulformulieren
De invoerschermen en -formulieren worden in de scripts met een sleutelwoord en codeblok gedefinieerd, waarbij invulvelden en alle bijbehorende attributen door de scriptschrijver worden bepaald. In figuur 3.1 en 3.2 zijn respectievelijk een formulier en de bijbehorende scriptcode te zien.



Figuur 3.1: Formulier

```
FORM
     WIDTH 700, 120
     SUBMIT

     HEADER            "Naam:"
     FIELD TEXT
           NAME        "naam"
           VALUE $i_naam
           MAX_CHARS 20
     END FIELD

     HEADER            "Wachtwoord:"
     FIELD TEXT
           NAME        "password"
           VALUE       $i_password
           MAX_CHARS   20
           PASSWORD
     END FIELD

     NEXTROW

     HEADER            "Echte naam:"
     FIELD TEXT
           NAME        "echte_naam"
           VALUE       $i_echte_naam
           MAX_CHARS   50
     END FIELD

     NEXTROW

     HEADER            "Actief?:"
     FIELD CHECK
           NAME        "actief"
           VALUE       $i_actief
           TEXT  "Uitvinken om de gebruiker de toegang tot het..
     END FIELD

     EMPTYROW

     HEADER            "Datum gemaakt:"
     FIELD STATIC
           VALUE       NOW()
     END FIELD

     EMPTYROW

     HEADER            "Notities:"
     FIELD TEXT
           NAME        "notities"
           VALUE       $i_notities
           ROWS        4
           WIDTH       500
     END FIELD
END FORM
```

Figuur 3.2: Formulier-scriptcode

Databasecommunicatie
Communicatie met de database gebeurt eveneens met speciale sleutelwoorden en codeblokken, waarbij eenvoudige sleutelwoorden en complexe SQL opdrachten afzonderlijk of gecombineerd gebruikt kunnen worden. Ik heb ervoor gezorgd dat alle databasefuncties door de parser worden gecontroleerd, onder andere op het bestaan van de aangesproken databasetabel en kolommen, en of de opdracht geen MySQL-foutmeldingen teweeg brengt.

De gegevens die in het formulier uit figuur 3.1 en 3.2 worden ingevuld, worden met het volgende codefragment in de database weggeschreven:

```
DATA INSERT
   SOURCE          "gebruiker"
   PUT        $i_naam,          "naam"
   PUT        $i_echte_naam,    "echte_naam"
   PUT        J_N($i_actief),   "actief"
   PUT        $i_notities,      "notities"
   PUT        MD5($i_password),"wachtwoord"

   PUT        NOW(),            "datum_gemaakt"
END DATA
```

Figuur 3.3: Gegevens in database plaatsen

Voorbeeld script en PHP resultaat

Als voorbeeld het volgende script:

```
// ------------------------------------------------------
//          Test scriptpagina - 1
//          rev.no.: 3
// ------------------------------------------------------

// Initialisatie:
INIT
      TEMPLATE    "test.htm"
      TARGET      "systeembeheer/test.php"
      MIN_ACCESS  "read"
END INIT

HEADER            "Test script"

$teller = 0
$waarde = 5

WHILE $teller < $waarde

      PRINT
            "Optellen: " + $teller + " van de " + $waarde + ".."
      END PRINT

END WHILE
```

Figuur 3.4: Scriptvoorbeeld

De scriptparser genereert uit dit script het volgende PHP bestand:

```
<?php /*

      .----.
     /___.--'-.                  +-----------------------------------------------------
     C   '----'                  |   OEPS - Generated script                    Parser
     |       )   _______         +-----------------------------------------------------
     |     -|   /_     /         |    Source      : scripts/test.script
     '''----'  / )    /          |    Target      : systeembeheer/test.php
     /        \/'..'__/          |    Template    : templates/test.htm
    /        /   /               +-----------------------------------------------------
   /            /                |    Created     : 01/10/2004 14:23
              /\~~)_____________  |    Warnings    : 0
                \(_____________)  |    Compiletime : 0.9548 seconds
                 | | | | | | |   +-----------------------------------------------------
                 | | |(__) | |

*/ ?>
<?php
      // -- Automatic includes (functions & script_functions): --
      include_once("../scriptparser/shared/inc_basic_functions.php");
      include_once("../scriptparser/shared/inc_database.php");
      include_once("../scriptparser/shared/inc_script_functions.php");

      // -- Database login : --
      $page_global["db_link"] = db_connect();
      if (!$page_global["db_link"])
```

```
	{
		$fail_melding = "Er kon geen databaseverbinding worden gemaakt!";
		include("inc_fail.php");
		die;
	}

	// -- Global page settings : --
	$page_global["section"] =		"none";
	$page_global["access"] =		"read";
	$page_global["logged_in"] =		auth_check();

	// -- Authorisation procedures : --
	if (!$page_global["logged_in"])
	{
		$fail_melding = "Je moet ingelogd zijn om de pagina aan te roepen.";
		include("inc_fail.php");
		die;
	} elseif (!auth_gedeelte_min(auth_get_user(), "none", "r")) {
		$fail_melding = "Je hebt onvoldoende privileges om de pagina aan te roepen.";
		include("inc_fail.php");
		die;
	}

	// -- FORM submit controle : --
	if (isset($_POST["submitted"]))
	{
		$page_global["form_submitted"] =	$_POST["submitted"];
		$page_global["form_name"] =		$_POST["submitted_name"];
	} else {
		$page_global["form_submitted"] =	0;
		$page_global["form_name"] =		"";
	}

?>
<html>
<head>
    <title>Hoofdscherm</title>
    <link rel="stylesheet" href="shared/stijl.css" type="text/css">
</head>
<body>
<!-- | ------------------------ | -->
<?php
// --- HEADER (1) - SCRIPT LINE: 13 ---
echo "<h1>". str_html_script("Test script") ."</h1>";

// --- VARIABLE toewijzing - SCRIPT LINE: 15 ---
$scrvar_teller = 0;

// --- VARIABLE toewijzing - SCRIPT LINE: 16 ---
$scrvar_waarde = 5;

// --- WHILE - SCRIPT LINE: 18 ---
while ($scrvar_teller < $scrvar_waarde)
{
		// --- PRINT BLOK - SCRIPT LINE: 20 to 22 ---
		echo "<p class=\"default\" align=\"left\">";
		echo str_html_script("Optellen: " . $scrvar_teller . " van de " .
				$scrvar_waarde . "..") ."<br>\r\n";
		echo "</p>";

}
// --- END WHILE ---
?>
<!-- | ------------------------ | -->
</body>
</html>
```

Figuur 3.5: Scriptvoorbeeld: PHP-resultaat

3.2.4 Documentatie

Referentiemateriaal
Als referentiemateriaal heb ik lijsten gemaakt van alle sleutelwoorden, taalconstructies en functies met de verplichte en optionele parameters, en heb deze ruim voorzien van uitleg en voorbeelden. Tijdens het programmeren van de parser en de scripts heb ik de wijzigingen meteen in de referentiedocumentatie bijgewerkt zodat deze voortdurend up-to-date en compleet is gehouden.

Handleidingen
Ten behoeve van de gebruikers heb ik een handleiding geschreven waarin de werking van de parserapplicatie wordt uitgelegd, en de minder evidente systeemschermen aan de hand van voorbeelden worden beschreven.

Ten behoeve van de systeembeheerders heb ik een handleiding gemaakt voor installatie van het systeem en de benodigde serversoftware zoals de webserver, het DMBS en PHP; hiermee kunnen zij zo nodig het gehele systeem van de grond af opbouwen.

In een apart document heb ik alle programmabestanden van het systeem in hun samenhang beschreven. Hierin kan een systeemprogrammeur opzoeken waar eventuele aanpassingen of toevoegingen moeten worden gemaakt.

3.3 IMPLEMENTATIE

Op het moment van dit schrijven is het systeem nog niet in zijn geheel overhandigd en geïmplementeerd. Om te controleren of de installatie op het ONC-netwerk naar behoren kan verlopen en of de installatiehandleiding klopt, is wel een prototypeversie bij het ONC geïnstalleerd. Gebleken is dat het systeem ook daar in principe naar behoren werkt.

Het ONC overlegt nog met mij over onderhoudsmogelijkheden, waarbij wordt gekeken naar eventuele uitbreidingen, andere toepassingen en verdere ontwikkeling van het scriptsysteem.

4 RESULTATEN EN EVALUATIE

4.1 RESULTATEN

4.1.1 Scripttaal

Scriptparser, applicatie en scripttaal
De scripttaal is robuust, en eenvoudig te lezen, te begrijpen en te leren, en voldoet aan alle gestelde eisen. Het resultaat is zelfs boven verwachting van de opdrachtgever, die mij een groot compliment heeft gegeven: "Het werk ermee is leuk en bijna verslavend."

De scripttaal is tegelijkertijd krachtig en eenvoudig te leren, zoals ik vanaf het begin voor ogen heb gehad. Ter illustratie: de inhoud van een databasetabel kan worden aangeroepen met een enkele regel zonder verdere parameters, waarna de parser zelf een logische HTML-tabel met alle kolommen genereert, maar ook met een groot aantal sleutelwoorden en parameters waarmee de kolommen, opmaak e.d. expliciet worden gekozen en ingesteld.

Het vertalen van scripts blijkt aanzienlijk langer te duren dan verwacht: van één seconde voor kleine scripts tot bijna tien seconden voor omvangrijke scripts. Dit bevestigt de keuze om scripts vooraf te compileren; dergelijke wachttijden zijn bij interactief gebruik van het systeem verre van acceptabel, terwijl het bij de infrequente beheertaken geen probleem vormt. Als er toch voor run-time vertalen zou zijn gekozen, had de parser versneld moeten worden met cache-technieken e.d., en zouden ook beperkingen aan de scripttaal e.d. nodig zijn geweest.

Systeem
Een gedeelte van het beheersysteem is niet met de parser gemaakt, waaronder het dynamische menu waarin opties aan de hand van een configuratiebestand worden aangegeven en een indexpagina waarin een HTML raamwerk is aangegeven. Dit omdat de code hiervoor te complex is en te weinig voorkomt om speciale parserroutines te rechtvaardigen.
Het doorbreekt echter de uniformiteit van het systeem en als ik er tijd voor zou hebben, zou ik de scriptfuncties zeker uitbreiden om ook deze functies te ondersteunen.

Er zijn minder scripts geschreven dan aanvankelijk was gepland, maar de belangrijkste delen van het systeem zijn af en het systeem is, op een aantal eenvoudige scripts na, gebruiksklaar. Alle meer ingewikkelde en bijzondere delen van het systeem zijn geschreven, getest en opgeleverd. De parser is hiermee uitvoerig getoetst, en de resterende scripts zullen zonder problemen gemaakt kunnen worden.

4.1.2 Documentatie

Referentiemateriaal
De referentiedocumenten zijn gereed en compleet met alfabetische lijsten van functies en sleutelwoorden met een uitvoerige beschrijving, en vormen voor de scriptschrijvers een goed naslagwerk.

Handleidingen
De handleiding voor scriptprogrammeurs geeft een uitgebreide beschrijving van de scripttaalregels, met mogelijkheden en voorbeelden. Daarnaast zijn er voorbeeldscripts met veel commentaar die meer praktijkgerichte uitleg geven dan de handleiding zelf, waardoor gebruikers zelf kunnen kiezen op welke manier zij bekend willen raken met het systeem en de scripttaal.
De gebruikshandleiding van de scriptparserapplicatie is beknopt, maar beschrijft de applicatie volledig; de gegeven voorbeelden blijken goed bij te dragen tot het begrijpen van de applicatie.

Installatieprocedure
De installatieprocedure is ervaren als een goed stappenplan. Het is in de praktijk toegepast en getest met de meest recente versies van de benodigde softwarepakketten.

4.2 EVALUATIE, LEERFEITEN

De definitiestudie is geheel naar de wensen van de opdrachtgever en binnen de geplande tijd voltooid, met een duidelijk afgebakende scope, en met goede concepten als resultaat.

Het contact met de opdrachtgever is goed verlopen, interviews en andere besprekingen hebben steeds tot bruikbare resultaten geleid en misverstanden waren er zelden of nooit. De opdrachtgever is zeer te spreken over de opdrachtuitvoering, de verslag-legging en de bereikte resultaten.

Het projectplan en de prioriteiten in het pilotplan zijn tijdens de bouw op de volgende punten bijgesteld:

- Het was aanvankelijk de bedoeling om een kant en klaar systeem te maken, compleet met de benodigde scripts. Dit bleek niet haalbaar, zodat de prioriteit is gelegd op het ontwikkelen van een goede systeemkern en een breed inzetbare scriptparser, en het schrijven van alleen de belangrijkste en de meest ingewik-kelde scripts. Het ONC schrijft zelf de overige scripts, hetgeen geheel strookt met de bedoeling van het systeem en met de verwachte aanpassings-behoeften in de toekomst.
- Bij het plannen heb ik achteraf gezien een onjuiste luxe-comfort verdeling voor de pilots gebruikt. De scripts zijn later namelijk grotendeels als "luxe" bestempeld omdat zij door het ONC zelf geschreven kunnen worden, terwijl de "comfort"-functionaliteit vooral betrekking heeft op het comfortabel gebruik van de script-parser, en niet van de te verwerken scripts.
- Er was nog geen databasebewerkingsgereedschap gedefinieerd in het concept. In de eisen werd wel aangegeven dat het script alle databasebewerkingen moest ondersteunen, maar niet dat pagina's met deze functies nu ook daadwerkelijk geschreven moesten worden. Dit is nu wel opgenomen.

In de elfde week van de opdrachtuitvoering heb ik het projectplan hierop aangepast; dit plan, met planning en prioriteiten, is verder ongewijzigd aangehouden, inclusief het ontwerp en de programmering van het gereedschap voor de databasebewerking als deel van de scriptparserapplicatie.
De aanpassingen zijn in overleg met- en met instemming van de opdrachtgever gemaakt, en wel zo vroeg in het traject dat de kwaliteit van de producten er niet onder heeft geleden.

Prototyping is zeer nuttig gebleken bij het kiezen van een grafisch ontwerp voor het systeem maar veel minder voor het concept van de scripttaal, omdat de opdrachtgever het systeem pas goed heeft ervaren nadat er een bruikbaar en herkenbaar resultaat mee te verkrijgen was. Het had tot gevolg dat het prototype complex en omvangrijk is geworden, en de opdrachtgever pas in een laat stadium zijn mening en ideeën aan heeft kunnen geven.
Dit leverde geen echte problemen op omdat het scriptsysteem zo is gebouwd dat het gemakkelijk aan te passen is. Daarbij kwam dat het concept meteen al grotendeels voldeed aan de wensen van de opdrachtgever, zodat weinig aanpassingen gemaakt hoefden te worden.

Tenslotte
De gemaakte producten blijken een afdoende oplossing te bieden voor de vastgestelde problemen; ze sluiten aan bij de wensen en verwachtingen van de opdrachtgever en stimuleren verder onderzoek naar gebruiksmogelijkheden en uitbreidingen, zoals de implementatie van een intranet op de scholen.

5 LEGENDA

Access	Microsoft Access. Een database en interface ontwikkelomgeving.
ASP	Active Server Pages. Een server-side programmeertaal en ontwikkelomgeving voor webservers.
BASIC	Beginner's All Purpose Symbolic Instruction Code. Een eenvoudige programmeertaal (ooit bedoeld als opstap naar krachtiger talen als FORTRAN en ALGOL).
Black-box testing	Ook wel functioneel testen genoemd. Bij het kiezen van testgegevens wordt niet gekeken naar de programmacode, maar alleen naar de interface en resultaten.
DBMS	DataBase Management Systeem.
expressie	Een verzameling van sleutelwoorden, functies, operators, variabelen en waarden, waarmee waarden binnen de scripttaal worden aangegeven.
HTML	HyperText Markup Language. De taal waarin webpagina's worden geschreven, voor gebruik met het HyperText Transfer Protocol.
IAD	Iterative Application Development. Een projectmanagement-methode voor software-ontwikkeling.
mIRCScript	Scripttaal waarmee het (Internet Relay Chat) chatprogramma mIRC kan worden geautomatiseerd en geconfigureerd.
MySQL	Open-source DBMS.
Parser	Programma dat zorgt voor het vertalen van broncode naar machinecode. Parse betekent letterlijk "interpreteren", of "ontleden".
PHP	Hypertext Pre-Processor, een server-side programmeertaal en ontwikkelomgeving voor webservers.
Scope	Het werkingsgebied van een project, programmafunctie, e.d.
Scope-Creep	Ook wel bekend als *Feature-Creep*. Het verschijnsel van steeds toenemende eisen en wensen gedurende een ontwikkeltraject.
Server-side	Geeft in een client-server omgeving aan dat de programmatuur wordt uitgevoerd aan de serverkant.
SQL	Structured Query Language. Een taal waarmee databasegegevens kunnen worden opgevraagd en bewerkt.
Template	Letterlijk: een patroon of mal. In deze context: een bestand waarmee het uiterlijk van een webpagina wordt bepaald.
UML	Unified Modelling Language. Een taal voor objectgeörienteerde analyse en ontwerp.
White-box testing	Ook wel structureel testen genoemd. Testgegevens worden gekozen op het zo goed mogelijk testen van de programmacode.