Departement Handelswetenschappen en Bedrijfskunde
Gegradueerde in Toegepaste Informatica
Programmatie en Analyse

Logboek- en dosimetriebeheer van gammabestralingsexperimenten in eZ publish

CAMPUS
Geel

Wim Konings

Academiejaar 2004-2005

De houder van dit diploma is gerechtigd tot het voeren van de titel van Bachelor

Woord vooraf

De stap tussen naar school gaan en gaan werken is niet te onderschatten. In de eerste plaats is er een overgang van 20 uur aanwezig zijn op school naar 40 uur werken op het bedrijf. Dit zorgde al voor een kleine aanpassing waaraan ik moest wennen. Daarnaast komen de sociale vaardigheden nog meer aan bod.
Maar dankzij de goede werksfeer en de grote behulpzaamheid van mijn collega's verdwenen deze problemen in het niets. Daarom wil ik een woord van dank uiten voor iedereen die mij gesteund heeft in deze periode.

In de eerste plaats wil ik mijn externe begeleidster Marie-Laure Ruyssen bedanken. Door haar overtuigingskracht koos ik voor deze stageopdracht van de dienst Knowledge Management Knowledge Centre, waar ik in eerste instantie niet had voor gekozen. Ze zorgde voor een interessante en aangename stageopdracht.
Verder waren er Tom Couwberghs, Hans Melis en Kristof Coomans die altijd klaar stonden om mij te helpen. Ik kon altijd vertrouwen op hun ruime technische kennis.
Mijn dank gaat ook naar al de Knowledge Centre medewerkers: Mario Bens, Wendy Machiels, Kris Pennemans, Arnout Ulenaers en mijn collega-stagiair Gwen Vansonhoven. Ook deze mensen zorgden mede voor de aangename werksfeer die ik de voorbije 3 maanden heb gevoeld.

Naast de mensen op de werkvloer wil ik ook nog enkele andere mensen bedanken. Eerst wil ik mijn eindwerkbegeleidster Kristine Mangelschots bedanken voor het nalezen van mijn werk en de nuttige tips voor de presentatie. Ook mijn ouders wil ik bedanken omdat ze altijd voor me klaar staan. En als laatste wil ik mijn vriendin Inge bedanken voor haar steun en aanmoedigingen die ze me gaf.

Bedankt allemaal!

Wim Konings

Samenvatting

De stageopdracht van het Knowledge Management op het SCK bestond uit 2 grote delen.
Ten eerste moest ik een logboek maken waarin wetenschappers hun wetenschappelijke waarnemingen konden wegschrijven.
Ten tweede moest ik een functionaliteit creëren waarin de waarnemingen van een dosimetrie bijgehouden konden worden. De 2 delen moesten geïntegreerd worden in de portaalsite EMS die de planning van experimenten bijhield.

De functionaliteiten moesten worden gemaakt in een nieuwe module met behulp van het Content Management Systeem eZ publish. eZ publish is naast een webapplicatie ook een ontwikkelingsframework waardoor ik de module kon aanmaken. Bij elke view van de module codeerde ik een gebruiksvriendelijke interface zodat de wetenschappers hun waarnemingen correct en eenvoudig konden registreren.

Binnen EMS was er een datatype aangemaakt om de periodes van experimenten te beheren. Maar naar mijn opdracht toe was het datatype niet compleet. Ik heb enkele aanpassingen gemaakt zodat dit probleem opgelost werd.

Daarnaast werden er ook nog enkele template operators aangemaakt of gewijzigd naar de nieuwe opdracht toe. Deze template operators maken het mogelijk om een grafiek te genereren aan de hand van de invoerde waarnemingen. Zo heb ik ook de mogelijkheid gecreëerd om de ingevoerde waarnemingen te kunnen exporteren naar Excel.

In de toekomst zullen er waarschijnlijk nog enkele aanpassingen aan mijn module komen. Niet alle wensen van de eindgebruiker konden aangepast worden. Dit lag vooral aan de korte periode van mijn stage.

Inhoudstafel

Illustraties

Alfabetische lijst van gebruikte afkortingen en symbolen

API	Application Programming Interface
ASP	Active Server Pages
BR1	Belgian Reactor 1
BR2	Belgian Reactor 2
BR3	Belgian Reactor 3
CLI	Command Line Interface
CMS	Content Management System
CSS	Cascading StyleSheets
CSV	Comma Seperated Value
DBX	Database aBstraction eXtension
DNS	Domain Name System
DLL	Dynamic Link Library
DOM	Document Object Model
DOS	Disk Operating System
DTD	Document Type Definitions
EMS	Experiment Management System
ESA	European Space Agency
GPL	General Public License
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protcol
IMAP	Internet Message Access Protocol
KM KC	Knowlegde Management Knowledge Centre
LDAP	Lightweight Directory Access Protocol
MySQL	My Structured Query Language
NEMO	Netwerk of Excellence on Micro-Optics
NNTP	Network News Transfer Protocol
ODBC	Open DataBase Connectivity
OO	Object Oriëntatie
PDF	Portable Document Format
PHP	PHP: Hypertext Preprocessor
POP3	Post Office Protocol 3
PWR	Pressurised Water Reactor
RDBMS	Relationeel DataBase Management Systeem
SAX	Simple API for XML
SCK•CEN	Studiecentrum voor Kernenergie – Centre d'étude de l'Energie Nucléaire
SNMP	Simple Network Management Protocol
SQL	Structured Query Language
STK	Studiecentrum voor de Toepassingen van de Kernenergie
TCP/IP	Transmission Control Protocol/Internet Protocol
URL	Uniform Resource Locator
VITO	Vlaamse Instelling voor Technologisch Onderzoek
VUB	Vrije Universiteit Brussel
vzw	vereniging zonder winstoogmerk
WebDAV	Web-based Distributed Authoring and Versioning
XHTML	eXtensible HyperText Markup Language
XML	eXtended Markup Language
XSLT	eXtensible Stylesheet Language Transformations

Inleiding

In dit eindwerk geef ik u een overzicht van de zaken die ik verwezenlijkt heb gedurende mijn stage op het SCK•CEN. Ik heb getracht om de uitwerking van mijn opdracht in logische stappen te verdelen. Op die manier hoop ik u inzage te geven in het redeneringsproces over mijn opdracht.

Op het SCK•CEN gaan onderzoek en kennis hand in hand. Het is van groot belang om resultaten van onderzoeken bij te houden. Op die manier wordt er geen dubbel werk verricht en kan iedereen gebruik maken van de beschikbare resultaten. Voor het centraal beheren van resultaten zorgt o.a. het Knowledge Management, de dienst waar ik de afgelopen 3 maanden heb gewerkt.

Vorig jaar startte het Knowledge Management met een portaalsite om experimenten bij te houden en te beheersen. Dit was het Experiment Management System. Het ging een overzicht creëren voor de wetenschappers om hun gammabestralingsexperimenten beter te beheersen. Het SCK•CEN voert deze experimenten uit in het kader van de ruimtevaart. Gammabestralingsexperimenten benaderen het meeste de elektronenrijke omgeving in de ruimte. Zo zal men kunnen waarnemen wat de gevolgen zullen zijn van een stof in de ruimte.
Voorlopig worden de resultaten van de experimenten enkel bijgehouden in een schrift. Dit biedt echter weinig mogelijkheden om resultaten op te zoeken, te vergelijken en te interpreteren. Daarom werd er gevraagd om een systeem te maken waarin de resultaten centraal beheerd konden worden.
Zo is mijn opdracht tot stand gekomen.

De opdracht bestond uit het maken van een systeem dat logboek- en dosimetriebeheer van gammabestralingsexperimenten verschafte. In het begin leek dit een moeilijke opdracht te worden. Ik kende het programma niet waarmee het Knowledge Management werkte en ik had geen benul van de termen logboek, dosimetrie en gammabestralingsexperimenten.
Maar ik heb me niet laten afschrikken door deze opdracht. Het resultaat van mij opdracht kunt u lezen in dit eindwerk.

In het eerste hoofdstuk krijgt u een overzicht van de functie van het SCK•CEN en de functie van het Knowledge Management Knowledge Centre binnen het SCK•CEN. Verder krijgt u in hoofdstuk 2 een duidelijke beschrijving van het Experiment Management System zodat u mijn opdracht goed kunt begrijpen. In de volgende hoofdstukken komen nog enkele technologieën en toepassingen aan bod. Tot slot wordt u in de laatste hoofdstukken geconfronteerd met de concrete uitwerking van de opdracht. U zal in aanraking komen met de geprogrammeerde code die u nader wordt uitgelegd.

1 SCK•CEN

SCK•CEN is de afkorting van Studiecentrum voor Kernenergie – Centre d'étude de l'Energie Nucléaire. Het centrum is een federale instelling van openbaar nut.
Het werd in het leven geroepen om de nucleaire wetenschap en de nucleaire industrie te promoten.

Momenteel houdt het zich voornamelijk bezig met het verrichten van wetenschappelijk onderzoek naar veiligere en vreedzamere toepassingen van kernenergie voor industrieel en medisch gebruik. Daarnaast doet het SCK•CEN ook onderzoek voor de ruimtevaart, zoals voor de ESA, om te kijken wat de reactie van materialen in de ruimte zou zijn.

In dit hoofdstuk beschrijf ik o.a. de geschiedenis van het SCK•CEN. Verder vindt u uitleg over de dienst waar ik de voorbije 3 maanden heb gewerkt, het Knowledge Management Knowledge Centre. Tot slot worden er nog 2 diensten onder de loep genomen die belangrijk geweest zijn voor mijn stageopdracht, namelijk de afdeling BR2 en de dienst Instrumentatie.

1.1 De geschiedenis in een notendop

Toen Kongo nog een kolonie van België was, werd er veel uraniumerts ontgonnen om naar België te transporteren. Uranium werd in de jaren '30 in de eerste plaats gebruikt voor de aanmaak van radium voor medische toepassingen.

In 1942 startten de Verenigde Staten van Amerika met een project voor de ontwikkeling van een atoombom. Ze wilden hiervoor gebruik maken van de uraniumreserves van België.

Op 26 september 1944 bereikten de Verenigde Staten van Amerika en het Verenigd Koninkrijk een overeenkomst met België om een alleenrecht op de uraniumvoorraden te hebben. In ruil kreeg België de nodige kennis voor commerciële en niet-militaire toepassingen (GOVAERTS, 2002:6).
Toen de Amerikanen enkele jaren later deze verbintenis niet meer nakwamen, was het de wetenschapper Pierre Ryckmans die ijverde voor een Belgische instelling die de mogelijkheden van kernenergie ging onderzoeken.

In 1952 richtten enkele wetenschappers o.l.v. Pierre Ryckmans de vzw STK op. Dit was het Studiecentrum voor de Toepassingen van de Kernenergie. De nieuwe vzw ging, zoals de naam al zegt, op zoek naar nieuwe toepassingsmogelijkheden van kernenergie.

Om toekomstige uitbreiding te garanderen en om de onderzoeken veilig te laten verlopen werd er een groot afgelegen terrein gezocht. De uiteindelijke keuze van de locatie viel op Mol.
Het was het ideale terrein om zich te vestigen omwille van zijn grootte, voldoende afstand tot bewoonde gebieden en het stabiele klimaat.
Na de aankoop van het terrein werd er gestart met de bouw van hallen, laboratoria en andere gebouwen waar de activiteiten konden plaatsvinden. Er werden zelfs woningen gebouwd waar de werknemers konden verblijven.

Enkele jaren later werd er gestart met de bouw van een eerste reactor om het onderzoeksgebied van de kernenergie te verruimen. Deze reactor kende zijn voltooiing in 1956 en werd BR1 genoemd.

In 1957 verscheen in het Staatsblad het Koninklijk Besluit dat het Studiecentrum voor de Toepassingen van de Kernenergie een instelling van openbaar nut met rechtspersoonlijkheid werd. Het nieuwe statuut ging gepaard met een naamsverandering. Voortaan zou het STK door het leven gaan als het SCK•CEN: het Studiecentrum voor Kernenergie – Centre d'étude de l'Energie Nucléaire.
Dit nieuwe statuut maakte een gemengde financiering van de overheid en de industrie mogelijk. Het ministerie van Economische Zaken fungeerde als voogdij-instantie. Verder verwierf het SCK•CEN eigen inkomsten door onderzoek te verrichten voor de privé-industrie en de overheid.

Vanaf september 1957 werd er begonnen met de bouw van een tweede reactor. Het was pas in 1961 dat deze nieuwe reactor volledig operationeel was. De nieuwe reactor, BR2, bevat een hoge neutronenflux om het gedrag van allerlei materialen onder hoge bestraling na te gaan. De BR2 is nog steeds de meest performante onderzoeksreactor van West-Europa.

Begin 1954 overwoog België het idee om een elektriciteitscentrale te bouwen. In 1956 werd een bestelling geplaatst bij een Amerikaanse firma, die de reactoren ontwikkelde. Oorspronkelijk wilde men deze centrale in Schaarbeek installeren maar dit plan ging niet door. Uiteindelijk werd ook deze reactor in Mol gevestigd en was in 1962 bedrijfsklaar. De BR3 zou fungeren als een elektriciteitscentrale en als proefstation voor prototype splijtstoffen.

Vanaf de jaren '70 ging het SCK•CEN zijn statuten aanpassen om de opdrachten te verruimen buiten de nucleaire sector. In opdracht van de overheid, wetenschap en industrie ging het Studiecentrum onderzoek verrichten op het gebied van leefmilieu, energietoepassingen, materialen en brandstofcellen via elektrolyse. Toch bleef de nadruk liggen op de nucleaire activiteiten.

In 1991 werd het SCK•CEN gesplitst. Er werden nationale subsidies voor het centrum overgeheveld naar het Vlaamse Gewest. Om deze subsidies niet te verliezen werd er door het Vlaamse Gewest de VITO opgericht. De Vlaamse Instelling voor Technologisch Onderzoek zou zich meer concentreren op onderzoek rond het leefmilieu, energie, grondstoffen en materialen terwijl het SCK•CEN zich ging concentreren op onderzoek rond radioactief afval en ontmanteling en stralingsbescherming.

Momenteel telt het SCK•CEN ongeveer 600 medewerkers waarvan één derde houder is van een universitair diploma. De totale omzet bedraagt 80 miljoen euro per jaar. Ongeveer de helft van de omzet is afkomstig van subsidies van de overheid. Het SCK•CEN haalt 40% van de omzet uit contractwerk en dienstverlening . De overige 10% wordt indirect behaald door activiteiten voor de ontmanteling van vrijgegeven installaties.

1.2 Het Knowledge Management Knowledge Centre

De SCK•CEN bibliotheek werd opgericht in 1953 om voor het eigen personeel de nucleaire literatuur van over de hele wereld beschikbaar te stellen.
In de statuten van het SCK•CEN wordt de taak van de bibliotheek omschreven als het 'verzamelen en bijhouden van de wetenschappelijke en technische documentatie'.
De bibliotheek werd omgevormd tot een nationale nucleaire bibliotheek die een ruime wetenschappelijke collectie kon aanbieden voor universiteiten, bedrijven en de overheid.

Tot 2001 bleef de bibliotheek gemeenschappelijk met de VITO. Vanaf februari verhuisde een deel van de bibliotheek naar het SCK•CEN.
Toen werd het concept bibliotheek in een ruimer kader geplaatst. Er was enerzijds het Knowledge Centre en anderzijds het Knowlegde Management. Het uitlenen en beheren van de boeken, artikels en tijdschriften wordt verzorgd door het Knowlegde Centre. Maar het Knowledge Management had een andere taak. Wetenschappers pennen vaak hun ondervindingen neer in een rapport. Maar deze rapporten werden niet centraal beheerd waardoor de juiste informatie moeilijk te vinden was. Het Knowledge Management biedt de mogelijkheid aan wetenschappers om hun rapporten centraal op te slaan zodat iedereen deze kan raadplegen. Aan de hand van een portaalsite biedt het Knowledge Management iedereen de mogelijkheid om op een eenvoudige manier zijn gewenste informatie te vinden.

Maar het Knowledge Management biedt niet alleen intern informatie aan. Het zorgt ook voor een verspreiding van de kennis buiten het bedrijf.
Een voorbeeld hiervan is NEMO, het Network of Excellence on Micro-Optics.
Het is een samenwerkingsplatform tussen 30 Europese partners gedurende 4 jaar. De coördinatie van dit project gebeurt door de VUB, de University of Technology van Warschau en het ForschungsZentrum van Karlsruhe.
Al de partners van het Network of Excellence voeren onderzoek uit naar o.a. micro-optische vezels. De resultaten van de onderzoeken worden op de website gestructureerd en geïntegreerd. Het Knowledge Management zorgt voor het beheer van de website.

1.3 BR2 en BR3 in het licht

In de volgende paragrafen vindt u wat meer informatie over twee reactoren, namelijk de Belgian Reactor 2 en 3. Het waren de diensten van deze reactoren waarmee ik moest samenwerken om mijn stageopdracht te vervullen.

1.3.1 BR2

In 1957 verschenen de eerste plannen om een tweede reactor te bouwen. Na enkele jaren hard werken slaagde men erin om de BR2 af te maken.

Vanaf 1961 werd de onderzoeksreactor operationeel. De afdeling BR2 bestaat uit de reactor, drie bassins en de reactorcontrolezaal. Deze bevinden zich in een metalen gebouw, in de vorm van een overkoepelde cilinder waarvan de dichtheid regelmatig gecontroleerd wordt.

De reactor werkt volgens een schema van opeenvolgende cycli die een periode van stilstand en een periode van werking (meestal 21 dagen) omvatten. Het hoofddoel van de reactor BR2 is de bestraling van materialen onder hoge neutronenflux.

De BR2-reactor wordt vooral gebruikt voor materialenonderzoek. Zo kan men o.a. onderzoek verrichten voor de geneeskunde.
Dit onderzoek wordt niet alleen gedaan voor de eigen departementen van het SCK•CEN, maar ook voor externe klanten.
Verder wordt de reactor ook gebruikt voor de productie van radio-elementen.

1.3.2 BR3

In 1954 overwoog België de bouw van een elektriciteitscentrale. De keuze viel op een Pressurised Water Reactor (PWR). Dit was een reactor die gekoeld en gemodereerd werd door water onder druk.

Uiteindelijk geraakt deze nieuwe reactor, BR3 genaamd, klaar in 1962. De BR3 zou fungeren als demonstratie-eenheid voor de bouw en uitbating van een industriële elektriciteitscentrale en diende als proefstation voor prototype splijtstoffen (GOVAERTS, 1999:17).
De reactor werd aangesloten op het Belgische elektriciteitsnet om ons land van meer stroom te voorzien.

Momenteel bevindt de BR3-reactor zich nog in de dienst Instrumentatie van het departement Reactor Veiligheid. De BR3-reactor is de eerste drukwaterreactor die in Europa werd gebouwd en hij is ook de eerste die wordt ontmanteld. Voor het SCK•CEN is de ontmanteling van deze reactor een ideale gelegenheid om zijn expertise te tonen en om bijkomende wetenschappelijke en technologische kennis te ontwikkelen.

De ontmanteling van BR3 draagt bij tot een enorme ontwikkeling van robots en delegeerde werktuigen om de blootstelling van de werknemers aan radioactieve straling zo gering mogelijk te houden. Veel aandacht en onderzoek zijn gegaan naar het ontwerpen van telegeleide werktuigen die in een nucleaire omgeving ingezet kunnen worden. Zij moesten gebruiksvriendelijk en veilig zijn voor de werknemers en uiteraard vervaardigd worden uit materialen die bestand zijn tegen nucleaire stralingen.

De resultaten van dit onderzoek leverden belangrijke informatie op over de veroudering van gelijkaardige instrumenten in onze kerncentrales, iets waar in de toekomst alvast rekening mee gehouden zal worden.

2 EMS

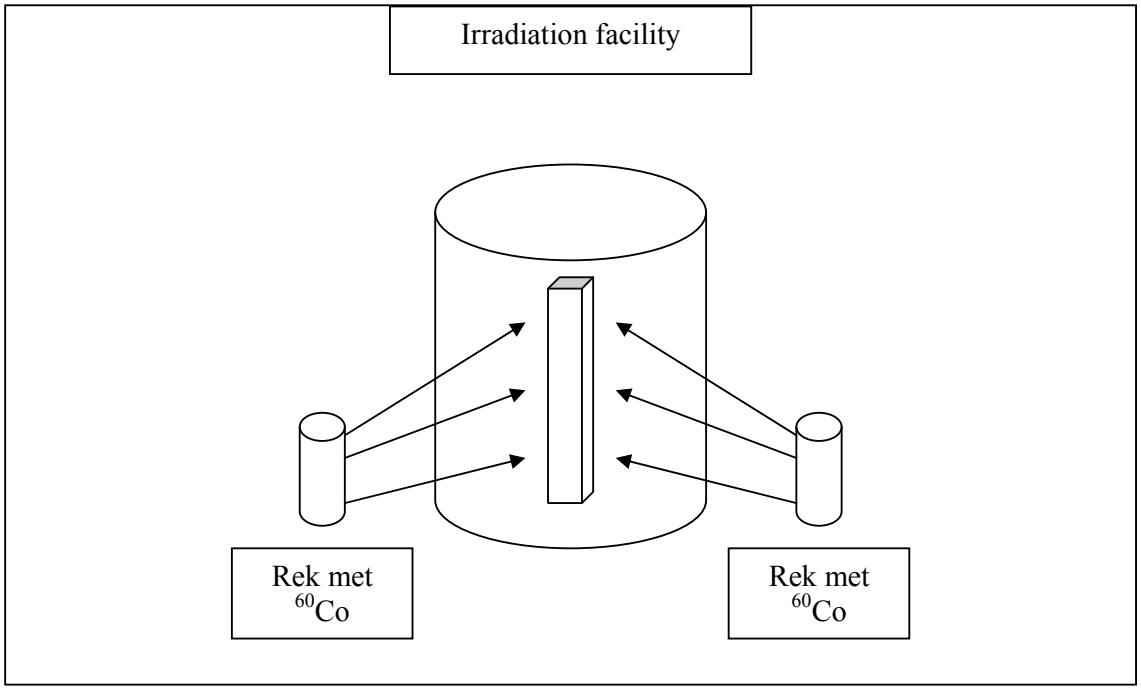
Vorig jaar had het KM KC een stageopdracht voor oudleerling Guy Diels om een portaalsite te maken om gammabestralingsexperimenten te beheren. De website moest een aantal functionaliteiten bieden aan de wetenschappers om gammabestralingsexperimenten beter te kunnen beheren. Dit resulteerde in de portaalsite EMS.
EMS is het Experiment Management System. Momenteel maakt de dienst Instrumentatie van het departement Reactor Veiligheid gebruik van de portaalsite om hun experimenten op de voet te kunnen volgen wanneer alles plaatsvindt. Andere wetenschappers kunnen op een eenvoudige wijze ook de planning van de projecten volgen. EMS zorgde voor een goed beheer van de experimenten.

Om EMS vlotter te kunnen begrijpen, leg ik eerst de werking en de types van een experiment uit.

Zoals al beschreven worden er op het SCK•CEN veel testen en onderzoeken verricht naar straling op verschillende materialen en de bijkomende gevolgen.
De gammabestralingsexperimenten gebeuren in het kader van onderzoek voor de ruimtevaart. Om deze experimenten uit te voeren, maakt het SCK•CEN gebruik van irradiation facilities. Deze benaderen het meest de elektronenrijke omgeving van de ruimte.
Een irradiation facility kan vergeleken worden met een cilindervormige container die omringd wordt met een rek waarin brandstof of ^{60}Co elementen voor het experiment zitten.
Het SCK•CEN bezit 5 gamma irradiation facilities waarvan er 2 werken met brandstof, 2 met ^{60}Co elementen en de laatste met neutron-gamma conversie. De 2 irradiation facilities die werken met ^{60}Co elementen bevinden zich onder water in een opslagkanaal van de BR2.
Al de experimenten worden verricht in opdracht van de dienst Instrumentatie maar ze worden uitgevoerd door de afdeling BR2.

Wanneer wetenschappers een materiaal willen onderzoeken, plaatst men dit in de irradiation facility. Door middel van de ^{60}Co elementen wordt het bestraald met γ-stralen. Tijdens de proef moeten er allerlei factoren gemeten en geregistreerd worden. Deze waarnemingen worden door de wetenschappers in een schrift geschreven. Achteraf gaan de wetenschappers controleren of hun voorspellingen overeenkomen met de waarnemingen.

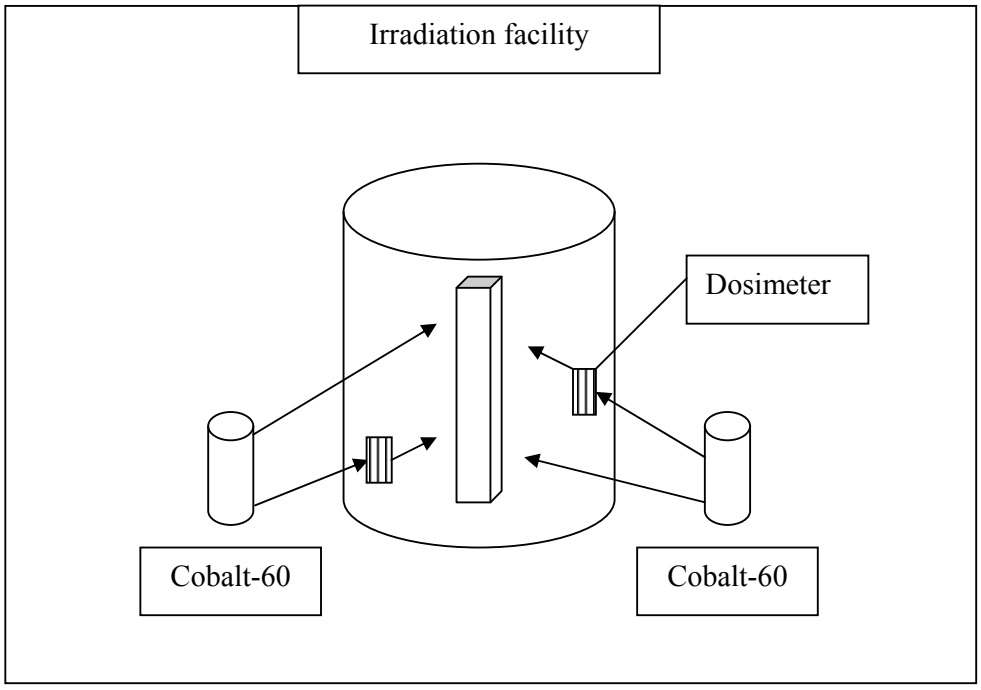
Aan de hand van de onderstaande figuur komt u tot een beter inzicht in de werking van gammabestralingsexperimenten.



Figuur 2.1 Een irradiation binnen een irradiation facility

Er zijn 3 types waaraan een experiment kan voldoen.

- Wanneer een materiaal bestraald wordt in een irradiation facility en verder niets meer, spreekt men over een irradiation. Ik heb reeds aangehaald dat een gammabestralingsexperiment het meest de elektronenrijke omgeving van de ruimte benadert. Een irradiation dient om na te gaan wat de reactie van dit materiaal in de ruimte zal zijn. Na de irradiations wordt er na gegaan of de verwachtingen overeenkomen met de resultaten.
- Wanneer een materiaal na verloop van tijd uit de irradiation facility gehaald wordt, spreekt men over een manipulation. Bestralingen lokken warmte uit. Om de temperatuur van het bestraalde materiaal te doen dalen, wordt het materiaal uit de irradiation facility gehaald.
- Het laatste type van een experiment noemt men dosimetrie. Dit wil zeggen dat men bij het materiaal dosimeters gaat plaatsen zodat deze sensoren de γ-stralen gaan absorberen. Een dosimeter kan je vergelijken met een amber- of geelkleurig stukje plastiek. Na het experiment zijn de dosimeters verkleurd door de straling. Vervolgens wordt de kleurverandering van de dosimeter gemeten met een spectrofotometer. Zo kan men de dosis via een vergelijking van een calibratiecurve berekenen. Dosimetries worden uitgevoerd om de veiligheidsgradaties van blootstelling aan bestraling te kunnen bepalen.



Figuur 2.2 Een dosimetrie binnen een irradiation facility

Nu u de werking en de afhandeling van experimenten doorgenomen hebt, zult u beter begrijpen wat mijn stageopdracht was. Aan de hand van de voorgaande informatie zult u de terminologie binnen de stageopdracht beter verstaan.

3 Stageopdracht

In dit hoofdstuk stel ik mijn stageopdracht voor.
Mijn stageopdracht werd een uitbreiding op het reeds bestaande project EMS.
EMS zorgde voor een goed beheer van gammabestralingsexperimenten maar met de waarnemingen van de experimenten werd weinig gedaan. Ze werden enkel in een schrift bijgehouden wat het niet makkelijk maakt om resultaten op te zoeken, te beheren en te interpreteren. Er was dringend nood aan een centrale opslagplaats voor de resultaten.
Uit de nood aan beheersing van de resultaten is mijn stageopdracht ontstaan.

In het eerste deel van mijn opdracht moest ik een logboek maken om wetenschappelijke waarnemingen bij te houden voor irradiations. In het tweede deel werd er een uitbreiding verwacht i.v.m. het opslaan van waarnemingen voor dosimeters.

3.1 Het logboek

Telkens wanneer er een experiment plaatsvindt, bestaat het experiment uit 3 types.
De dienst Instrumentatie bepaalt wanneer een experiment wordt gepland en van welk type dit moet zijn. Het is ook mogelijk dat er tijdens een experiment zowel een irradiation, manipulation als een dosimetry worden uitgevoerd.

Wanneer er een experiment wordt uitgevoerd, moet de afdeling BR2 de waarnemingen bijhouden. Afhankelijk van welk type experiment worden er andere parameters opgeslagen. Indien er tijdens een experiment een irradiation plaatsvindt, moeten de wetenschappers o.a. de temperatuur en de druk bijhouden. Het was mijn taak om voor de wetenschappers van de afdeling BR2 een functionaliteit te voorzien zodat zij hun waarnemingen op een eenvoudige manier kunnen wegschrijven in een elektronisch logboek. Voorlopig werden de waarnemingen nog in een schrift bijgehouden dus een gecentraliseerd beheer van de resultaten zou een grote stap vooruit zijn. Het maakt het makkelijker om resultaten op te zoeken en te vergelijken.

De dienst Instrumentatie moet deze resultaten kunnen bekijken en interpreteren. Om de gegevens te kunnen interpreteren, moest er een mogelijkheid zijn om een grafiek te genereren van de ingevulde factoren ten opzichte van de tijd. Verder moest de applicatie een export van de gegevens naar Microsoft Excel ondersteunen.

3.2 De dosimetrie

Wanneer er een dosimetrie gepland is voor een experiment, gaat de afdeling BR2 dosimeters bij de irradiation facility plaatsen. De wetenschappers gaan het materiaal bestralen met γ-stralen .
De mensen van de dienst Instrumentatie hebben vooraf bepaald op welke positie een dosimeter geplaatst moet worden.

Wanneer er zich een dosimetrie voordoet, is het belangrijk dat de wetenschappers van de afdeling BR2 de absorptie van een dosimeter kunen berekenen.
Deze absorptiewaarden worden door de mensen van de BR2 ingevoerd zodat de dienst Instrumentatie een overzicht krijgt van de waarnemingen. De absorptie van een dosimeter is belangrijk om de hoeveelheid Gray-straling te weten te komen. De wetenschappers halen na het experiment de dosismeter uit de irradiation facility. Vervolgens wordt er licht door de verkleurde dosimeter gestraald en via een spectrofotometer wordt de hoeveelheid absorptie gemeten. Door middel van de verkregen absorptiewaarde kan men de overeenkomstige gemiddelde dosis Gray berekenen. Dit gebeurt met de vergelijking van een calibratiecurve. Elke plaat waaruit een dosimeter is gemaakt, heeft zijn eigen calibratiecurve. Het is door de absorptiewaarde in de vergelijking van de curve in te vullen dat de dosis ontstaat. In de bijlage 1 en 2 vindt u een voorbeeld van een calibratiecurve.

Voorlopig is het enkel de bedoeling om de opgeslagen waarnemingen te tonen aan de dienst Instrumentatie. In de toekomst moet het misschien mogelijk gemaakt worden om de waarnemingen op een andere manier te bezichtigen. De gebruikers hadden hier nog weinig concrete eisen over zodat ze voorlopig alles in tabelvorm te zien krijgen.

3.3 Integratie in een portaalsite

De portaalsite van het SCK•CEN biedt de wetenschappers al een resem aan informatie. Daarom is het handig dat nauw aansluitende projecten geïntegreerd kunnen worden op de portaalsite. Vorig jaar ontwikkelde Guy Diels, een vroegere medestudent, een portaalsite om gammabestralingsexperimenten te beheren (DIELS, 2004). Zijn werk werd verder uitgebreid en afgewerkt door Kristof Coomans, een andere vroegere medestudent en nu werknemer op het SCK•CEN.
Aangezien mijn opdracht een uitbreiding is van dit project, is het vanzelfsprekend dat de module die ik moest maken hierin geïntegreerd moest worden.

Om de integratie van de module zo min mogelijk in de weg te staan, werd als basis voor de ontwikkeling van de module gekozen voor eZ publish. De mensen van het Knowledge Management hadden al enkele jaren goede ervaringen met dit systeem.

In één van de volgende hoofdstukken zal u meer komen te weten over het Content Management Systeem eZ publish, maar eerst worden de technologieën en toepassingen beschreven waarvan eZ publish gebruik maakt.

4 Overzicht van de gebruikte technologieën en toepassingen

Tijdens mijn stageperiode heb ik kennis gemaakt met eZ publish. eZ publish is een Content Management Systeem dat gebruik maakt van verschillende technologieën en toepassingen. Al deze technologieën zijn toepassingen van het open source principe.

Het open source principe betekent niet noodzakelijk dat al de software gratis is. Ook bij dit principe wordt er gebruik gemaakt van licenties. Een voorbeeld hiervan is de General Public License (GPL).
Wanneer er gesproken wordt over vrije software, bedoeld men vrije broncode en niet noodzakelijk software die gratis is. De GPL zorgt ervoor dat niemand u het recht kan ontzeggen om de broncode vrij te verkrijgen. U moet met software die onder de GPL valt altijd de broncode beschikbaar stellen voor gebruikers.
Verder mag u onder de GPL kopieën verspreiden van vrije software. U hebt ook het recht om wijzigingen aan de broncode te maken indien u dit vermeldt.
De enige manier waardoor u geld kunt verdienen aan de GPL is door uw services aan te bieden. Zo kunt u bijvoorbeeld geld vragen als u een website, gemaakt met open source software, moet onderhouden.

4.1 LAMP gebaseerd

eZ publish is LAMP gebaseerd. Dit wil zeggen dat het geschreven is voor Linux, de webserver Apache en het relationeel databasemangementsysteem MySQL met de scripting taal PHP. Al deze technologieën en toepassingen vallen onder de open source licentie, meer bepaald onder de GPL.

Omdat de meeste mensen maar weinig over deze technologieën vernomen hebben, stel ik ze even voor.

4.1.1 Apache

Al de informatie die in de volgende paragraaf beschreven staat, kunt u vinden op de website van Apache (Apache, 2005).

Apache is een krachtig, flexibel en robuust systeem dat door de meeste platforms ondersteund wordt. Onder het woord platform verstaan we het geheel van een besturingssysteem, de hardware en de taal die vanuit het besturingssysteem de hardware gaat aanspreken.

Het Apacheproject wordt beheerd door de Apachegroep. Dit zijn een groep van vrijwilligers van over de hele wereld die zich inzetten om de Apacheserver te verbeteren.

Doordat het een open source product is, kan iedereen de broncode bekijken en eventuele suggesties en verbeteringen aan het systeem maken.
Op die manier is de naam Apache tot stand gekomen. De populairste reden voor de naam kwam doordat het 'A PAtCHy server' was. De eerste versie van de webserver bestond uit de broncode met een resem aan patches er aan toegevoegd. Dankzij verschillende aanpassingen en het optimaliseren van het systeem is Apache momenteel één van de meest gebruikte open source webservers. Deze uitspraak wordt bevestigd door o.a. de website http://cities.lk.net/usawebstat.htm.

4.1.2 MySQL

Het volgende hoofdstuk is hoofdzakelijk gebaseerd op de documentatie van de website van MySQL .

MySQL is een relationeel databasemanagementsysteem (RDBMS). Het is een open source product met een vrij toegankelijke broncode.
Met MySQL is het mogelijk om grote hoeveelheden data te bewaren en te manipuleren.
Aan de hand van een scripttaal zoals bijvoorbeeld PHP kan de data op een eenvoudige manier opgeroepen en bewerkt worden vanuit een webpagina.

Hier kunt u enkele kenmerken van MySQL vinden:

- Integratie

Een groot pluspunt voor MySQL is dat het platform onafhankelijk is. U kunt MySQL zowel installeren op een server met besturingssysteem Mac OS X, Windows, Solaris, Linux en nog enkele andere.

- Schaalbaarheid

Met MySQL bestaat er de mogelijkheid om meerdere RDBMS'en met elkaar te verbinden. Wanneer deze situatie zich voordoet, zal er replicatie tussen de RDBMS'en ontstaan. Één van de databases zal de functie van master op zich nemen terwijl de anderen de rol van slave zal aannemen. Bij de master zullen alle aanpassingen aan de database uitgevoerd worden en opgeslagen in een logbestand. De slave zal door het repliceren gebruik maken van de opslagen logbestanden van de master om zijn database ook aan te passen.

- Prestatie

Naast het master-slave replicatie principe bestaat er ook het principe van clustering. Clustering wil zeggen dat meerdere MySQL servers parallel gaan draaien. De databases van de servers vormen binnen de cluster één fout-tolerante database. De cluster maakt gebruik van de MySQL servers met een cluster storage engine. Een groot voordeel van clustering is dat de hoeveelheid downtime van kritische applicatie tot bijna 0 minuten herleid wordt.

- Gebruiksvriendelijkheid

Volgens hun website is MySQL 'The world's most popular open source database'. Dit heeft het voornamelijk te danken aan zijn veelzijdigheid.
Door allerlei API's beschikbaar te stellen kunnen ontwikkelaars op een eenvoudige manier de database aanspreken. Zo is er bijvoorbeeld een API om vanuit PHP verschillende SQL-instructies te kunnen uitvoeren op de database. Dit maakt MySQL enorm populair bij ontwikkelaars van toepassingen.

Tot slot wil ik er u op wijzen dat MySQL tot de top behoort op gebied van databasemanagementsystemen. Dit is te merken aan de referenties die u op de website van MySQL kunt vinden. Grote organisaties zoals NASA, Cisco en Google maken allen gebruik van MySQL als databasemanagementsysteem.

4.1.3 PHP

PHP is de afkorting van PHP: Hypertext Preprocessor. Het is een veelgebruikte open source scripting taal. PHP wordt vaak gebruikt om webpagina's te ontwikkelen. Zo kan PHP makkelijk verwerkt worden tussen HTML.

Hier volgt een eenvoudig voorbeeld om het gebruik van PHP uit te leggen:

```
<html>
   <head>
        <title>Example</title>
   </head>
   <body>
        <?php
             print "Hello world!";
        ?>
   </body>
</html>
```

Figuur 4.1 Een PHP script

Het valt u misschien op dat dit een eenvoudige manier is om een HTML-pagina aan te passen. PHP heeft zijn eigen start- en stoptag waar de PHP-code tussen geplaatst kan worden. Er kan makkelijk in of uit PHP-mode gegaan worden.

Maar dit is niet meer de huidige trend om PHP te verwerken in een HTML-pagina.
Op deze manier is er geen onderverdeling tussen de logicalaag en de presentatielaag. Ze worden door mekaar gebruikt wat de flexibiliteit niet ten goede komt.
Het zou een interessantere manier van werken zijn als de logicalaag en de presentatielaag gescheiden zijn. Door dit onderscheid kunnen er op een eenvoudigere manier aanpassingen gemaakt worden. PHP-scripts kunnen in de logicalaag zorgen voor de verwerking van de data terwijl templates in de presentatielaag de output van de PHP-scripts op het scherm tonen. Een programma dat gebruikt kan worden om templates te genereren voor PHP-scripts is bijvoorbeeld Smarty.

Er zijn nog enkele andere scripting talen die gebruikt kunnen worden om HTML te genereren. Zo kan een ontwikkelaar gebruik maken van ASP of ASP.NET om een HTML-pagina aan te passen.

Verder biedt PHP vele uitgebreide mogelijkheden voor professionele programmeurs maar zelfs beginnelingen kunnen PHP-scripts maken. Het is namelijk een taal die zeer makkelijk aan te leren is. Dit is bijvoorbeeld door de uitgebreide documentatie die u op website kunt vinden en de gelijkaardige syntax aan Java en C++.

Om nu een webpagina van PHP-code te voorzien hebt u drie dingen nodig, namelijk een PHP parser, een browser en een webserver. De parser gaat de PHP-code in uw webpagina's controleren en omvormen tot HTML. De browser gaat de webpagina's tonen.
Tot slot moet u ook nog de webserver installeren om uw webpagina's 'online' te plaatsen.
De webserver moet een connectie hebben met de PHP parser. Dit is om de webserver te laten weten waar hij PHP-code moet laten omvormen om de pagina's juist te kunnen weergeven.

Er bestaan twee grote gebieden waar PHP gebruikt wordt.

- Server-side scripting

PHP is een server-side scrinting taal. Dit wil zeggen dat de PHP-code wordt omgevormd op de server en vervolgens de output naar de browser stuurt. Hier bevindt zich een groot verschil met bijvoorbeeld javascript. Javascript is een client-side scripting taal. Wanneer javascript in een webpagina gebruikt wordt, zal het omgevormd worden door de webbrowser en niet door de server.
Dit is een nadeel wanneer u een webpagina aanmaakt.
Webbrowser geven niet altijd hetzelfde resultaat na het uitvoeren van javascript. Op die manier bekomt men vaak verschillende resultaten van output.
Wanneer er gebruik gemaakt wordt van server-side scripting, zal er altijd een zelfde output naar de browser vertrekken wat vaker in een geslaagd resultaat oplevert.

- Command line scripting

U kunt PHP scripts ook uitvoeren zonder gebruik te maken van een webserver of een browser. Dit soort scripts worden uitgevoerd aan de hand van een CLI, een Command Line Interface. Ze zijn ideaal om taken uit te voeren die op regelmatige basis voorkomen. U kunt de scripts vergelijken met scripts geschreven in Pearl.

PHP kan op alle grote platforms gebruikt worden zoals Linux, Microsoft Windows en Mac OS X.
PHP biedt ook ondersteuning voor verschillende webservers. Apache, Microsoft IIS, Netscape and iPlanet servers. Vele andere webservers kunnen gebruik maken van PHP omdat PHP voor al deze servers een module heeft gemaakt. De module is een dll-bestand dat een aantal functionaliteiten bevat specifiek voor elke webserver. Aan de hand van de module kan PHP communiceren met de webserver.

Eén van de sterke punten van PHP is dat het vele databanken ondersteunt. Het is namelijk niet zo moeilijk om een webpagina te ontwikkelen die gebruik maakt van een databank. Hier vindt u een lijst met alle ondersteunde databanken.

Adabas D	InterBase	PostgreSQL
dBase	FrontBase	SQLite
Empress	mSQL	Solid
FilePro (read-only)	Direct MS-SQL	Sybase
Hyperwave	MySQL	Velocis
IBM DB2	ODBC	Unix dbm
Informix	Oracle (OCI7 and OCI8)	
Ingres	Ovrimos	

Er bestaat ook een DBX, Database aBstraction eXtension, om de gebruiker een database te laten kiezen die door deze extension ondersteund wordt. Zo ondersteunt PHP ook ODBC, de Open Database Connection standaard. Dit biedt de gebruiker de mogelijkheid om op een andere database te connecteren.

Nog een ander voordeel van PHP is dat het niet alleen HTML kan aanmaken. Naast HTML kan PHP ook nog output leveren in de vorm van XML, PDF, tekst, …

PHP biedt ook ondersteuning om met andere services te communiceren via protocollen zoals LDAP, IMAP, SNMP, NNTP, POP3 en HTTP.

Tot slot heeft PHP een handige tekstverwerking mogelijkheid om XML documenten te parsen. Hiervoor biedt PHP ondersteuning voor DOM en SAX. Er is ook de mogelijkheid XSLT extensions te gebruiken om XML documenten te transformeren.

5 eZ publish

Enkele jaren geleden werd op het SCK•CEN de bibliotheek omgevormd tot het Knowledge Management Knowledge Centre. Op dat moment moest er een keuze gemaakt worden welke software men ging gebruiken voor de ontwikkeling van de projecten.
Na een beraadslaging werd er gekozen voor het Content Management System eZ publish.

eZ publish is een open source Content Management System en ontwikkelingsframework met functionaliteiten voor webpagina's en intranets. Standaard biedt eZ publish al een hele resem functionaliteiten zoals workflowimplementatie, versiebeheer van documenten, ondersteuning voor het publiceren van artikels in meerdere talen, e-commerce functionaliteiten, … Het is een solide en flexibel systeem dat via een webinterface beheerd kan worden.

In dit hoofdstuk vindt u informatie die voornamelijk te vinden is op de website van eZ publish (EZ, 2005).

5.1 Content Management System

Een Content Management Systeem (CMS) of kennisbeheersysteem is een tool dat het mogelijk maakt om content (zoals tekst, grafieken, video, etc) te creëren, bewerken, beheren en tot slot publiseren. Het moet hierbij wel rekening houden met enkele gecentraliseerde beperkingen zoals regels, processen en workflows zodat er een samenhangende, gevalideerde website/portaalsite kan worden gemaakt. Een CMS kan zowel worden gebruikt door technische als non-technische gebruikers. (RUYSSEN, 2002)

Telkens als er een bestand wordt gecreëerd ongeacht welk type bestand, wordt dit opgeslagen als 'content', ruwe data.
Het is de taak van het Content Management System om elk contentelement te beheersen. Op deze wijze wordt alles automatisch volgens een gestructureerd patroon en op een flexibele manier opgeslagen. De contentmanager moet enkel zorgen dat de informatie op het scherm wordt getoond.

De kracht van een CMS steunt op het concept van 'single source publishing' waarbij dezelfde informatie automatisch gepubliceerd kan worden in verschillende formaten (ROBERTSON, 2003). Ook eZ publish gaat verder dan het publiceren van enkel webpagina's. Het biedt de functionaliteit om naast HTML ook andere bestandsformaten zoals PDF voort te brengen.

5.2 De interne structuur van eZ publish

Naar mijn mening is het systeem eZ publish 'eaZy' om te gebruiken dankzij de ruime documentatie. De belangrijkste reden hiervoor is waarschijnlijk omdat ik enkel de functionaliteiten van lage niveau's heb gebruikt. De complexiteit van deze niveau's is nog beperkt. Maar de verwerking van alle niveau's van het systeem is al minder eenvoudig.
eZ publish is een complex object georiënteerde applicatie geschreven in PHP. Het systeem wordt opgesplitst in 3 grote delen namelijk de OO-datalaag, de logicalaag en de templates presentatielaag.

5.2.1 De OO-datalaag

In de OO-datalaag van eZ publish bevindt zich de kernel en de libraries.
De kernel kan worden beschreven als de kern van het systeem. Het is vergelijkbaar met het hart van een mens. Het is een collectie van mechanismen die de functionaliteiten van lage niveau's afhandelt. Enkele voorbeelden van deze functionaliteiten zijn content afhandeling, afhandelen van workflows, versie controle, access controle, …
Het is de kernel die de database zal aanspreken wanneer er afhandeling van content moet gebeuren.

De libraries van eZ publish zijn de bouwstenen van het systeem. Deze bouwstenen zijn PHP klassen die door het systeem kunnen aangesproken worden. Het zijn herbruikbare klassen die onafhankelijk zijn van de kernel.
In eZ publish kunt u de libraries vinden onder het mapje 'lib'.

Een voorbeeld van deze voorgeprogrammeerde scripts is ezxml.php. Deze klasse is in staat om vanuit een XML-bestand een DOM object te creëren waardoor de XML kan gebruikt worden binnen eZ publish.
Een ander voorbeeld van een library klasse is ezfile.php. Deze klasse kan vanuit eZ publish een bestand aanmaken.
Zo zijn er nog verschillende voorgedefinieerde klassen, elk met hun eigen specifieke eigenschappen, die u probleemloos kunt aanspreken.

5.2.2 De logicalaag

In de logicalaag van eZ publish vindt u de modules terug.
Een module binnen eZ publish is een verzameling van functionaliteiten. Elke module heeft zijn eigen kenmerken en taken binnen eZ publish. Om een bepaalde taak te kunnen oproepen moet een module aangesproken worden. De module zorgt ook voor een interface van de functionaliteit naar een mechanisme van de kernel. Zo kan de specifieke taak van de module worden afgehandeld door de kernel.
In eZ publish kunt u de modules vinden onder het mapje 'kernel'. Daar vindt u alle interfaces naar de mechanismen van de kernel.

Een voorbeeld van een module is de content module. Binnen deze module bevinden zich allerlei functionaliteiten om objecten aan te passen. Er is bijvoorbeeld de taak removenode.php die zoals de naam al doet vermoeden, een node binnen eZ publish kan verwijderen. Een andere taak binnen de content module is view.php. Dit script zal er voor zorgen dat alle content objecten op het scherm getoond worden.
Een ander voorbeeld van een module is de user module. Binnen deze module bevinden zich allerlei taken die verband houden met het inloggen en uitloggen van een gebruiker. Binnen deze module is er bijvoorbeeld het script login.php die gaat controleren dat er een correcte login wordt verricht.

5.2.3 De templates presentatielaag

In het laatste deel van eZ publish vindt u de templates terug. Templates vormen een fundamenteel onderdeel om een website te tonen.
Een template is een overschrijfbaar XHTML-bestand dat beschrijft hoe een bepaald type content getoond moet worden. De templates vindt u in eZ publish terug onder het mapje ‘design’.

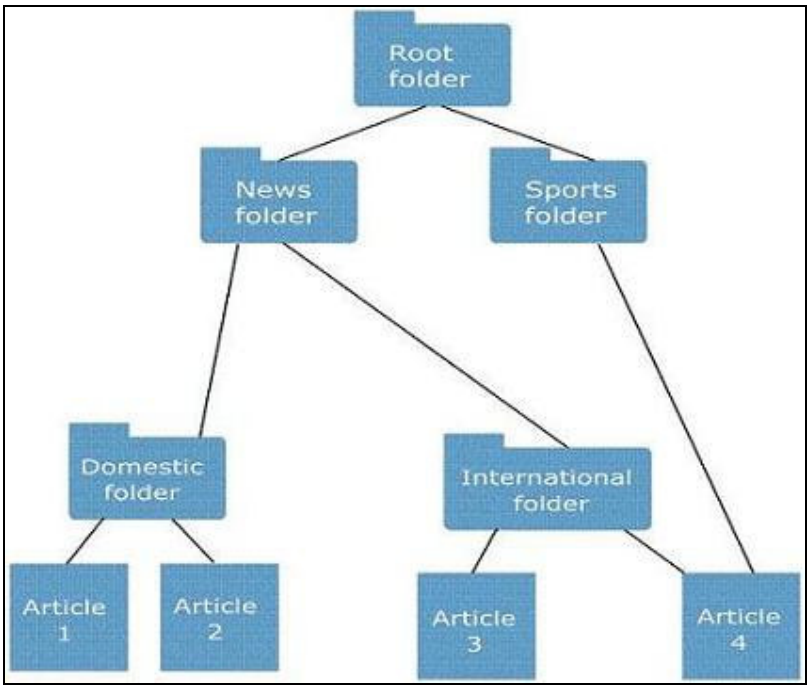
Om de content op een leesbare manier te tonen, moet de gebruiker templates aanmaken.
Maar om de web pagina grafisch mooier te maken, moet men gebruik maken van grafische ontwerpen.
Deze grafische ontwerpen zijn stylesheets en worden in het design gedeelte gemaakt. In een stylesheet of CSS worden kleuren, lettertypes, posities, tabellen, … aangemaakt om het geheel te verfraaien. De stylesheets worden vervolgens aan de template gekoppeld om het geheel af te werken.

5.3 Objecten, nodes en de content node tree

In de vorige paragraaf heb ik u verteld dat eZ publish een complex systeem is. Omdat het toch de moeite waard is om te weten hoe eZ publish alles verwerkt, ga ik deze complexiteit trachten uit de doeken te doen. Er wordt dieper ingegaan op de manier waarop eZ publish de ruwe data verwerkt en opslaat.

eZ publish is een object georiënteerde applicatie. Elk nieuw gemaakt object wordt geïnstancieerd als een object van de content klasse. Deze objecten moeten op een gestructureerde manier opgeslagen worden zodat de verwerking en behandeling ervan vlot kan verlopen. Om het probleem van het gestructureerd opslaan op te lossen, werd er een opslagsysteem bedacht door middel van nodes en een content node tree. In hoofdstuk 5.4 worden de voornaamste tabellen voor het opslaan van objecten uitlegd.

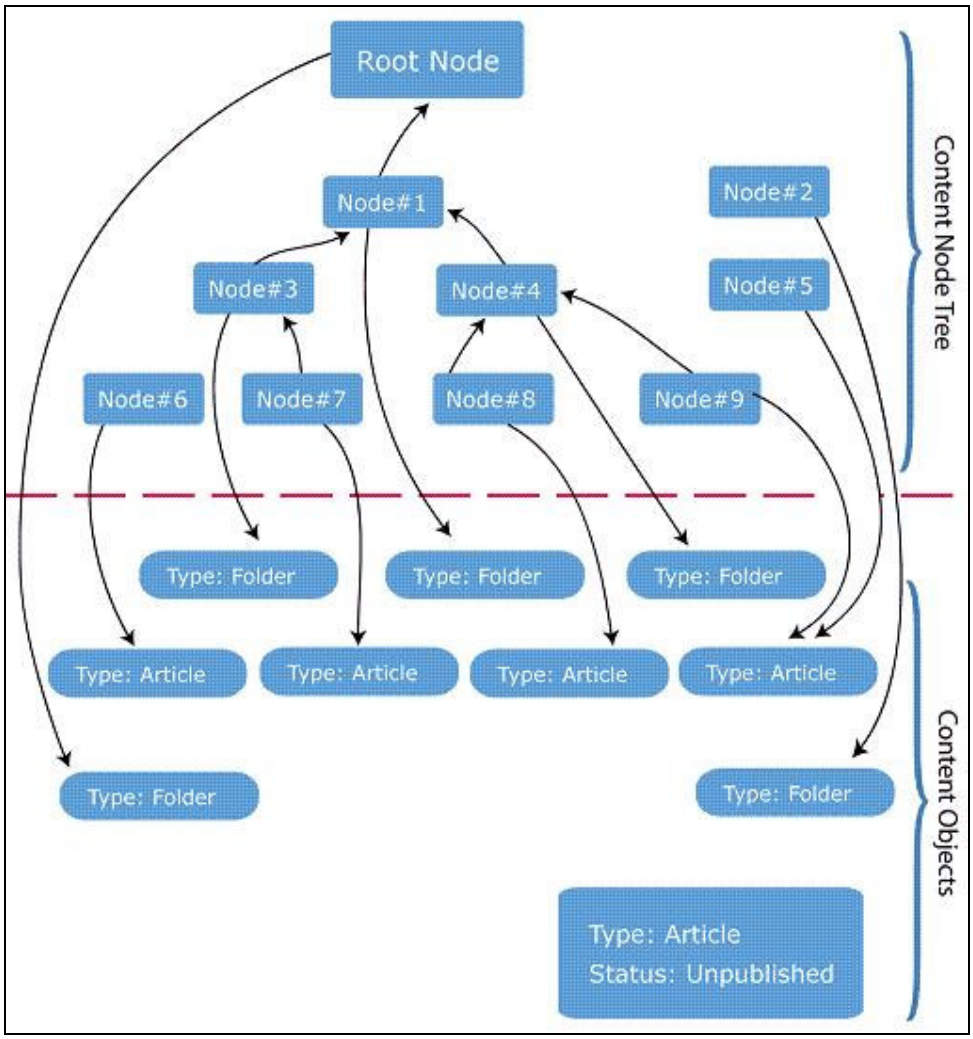
Een node is de verpakking van een content object. Aan de hand van al de nodes wordt er een content node tree opgebouwd. U kunt het vergelijken met de structuur van de verkenner in Windows. Elke content node tree bevat minimum 1 node, namelijk de root node. Het wordt hiërarchisch opgebouwd met alle nodes die er zijn.



Figuur 5.1 Voorbeeld van een content node tree in eZ publish

Door middel van de content node tree worden alle nodes beheerd. Maar hoe is de samenhang tussen nodes en objecten?
Elke node behalve de root node heeft 2 verwijzingen.
De eerste verwijzing gebeurt naar de parentnode. Deze verwijzing is belangrijk om de structuur en de opbouw tussen de nodes te maken. We bevinden ons hier in de logicalaag van eZ publish.
Door middel van de tweede verwijzing hangt het object vast aan de node. Nu wordt de overgang gemaakt naar de OO-datalaag van eZ publish. Het is zelfs mogelijk dat een object gekoppeld is tussen 2 nodes, maar 1 node verwijst slechts naar 1 object.
Op deze manier is de samenhang tussen de data en het opslaan van de data verzekerd zonder dat er zich problemen voordoen.

Op de volgende pagina kunt u figuur 5.2 zien die de relaties tussen de nodes en de objecten verduidelijkt..



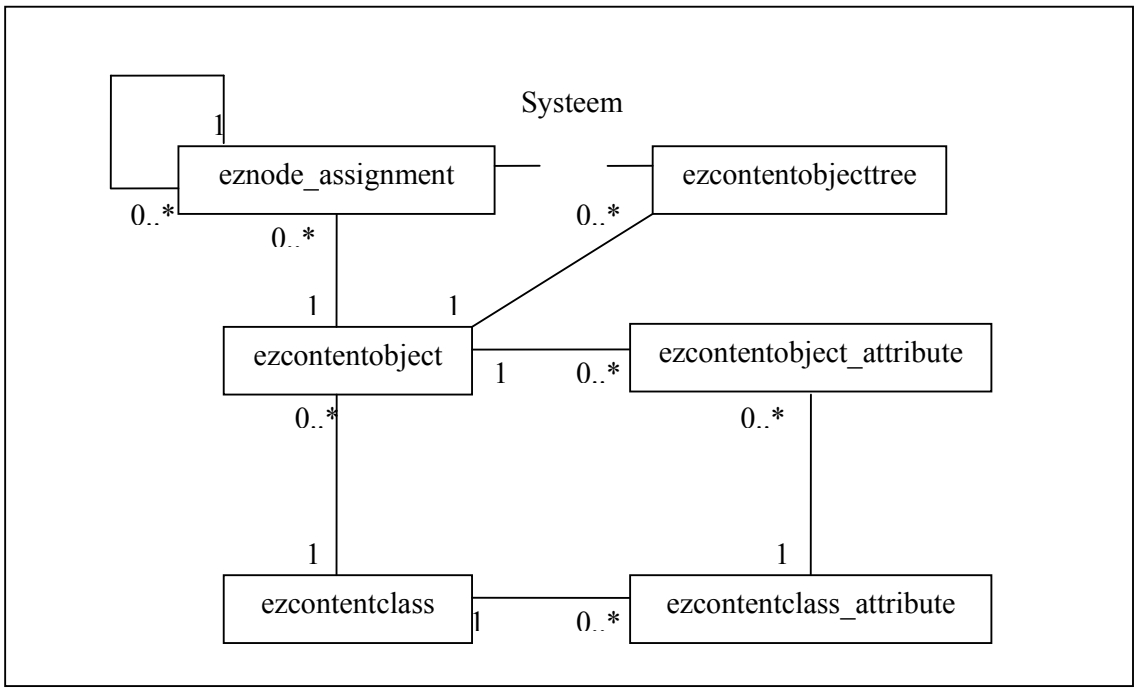
Figuur 5.2 Voorbeeld van een node structuur in eZ publish

5.4 Relationele diagram van eZ publish

In dit hoofdstuk vertel ik u hoe het relationele model van de database van de belangrijkste module in eZ publish werkt. Deze module is de content module.

Zoals al vermeld is, is eZ publish een object geörienteerde applicatie. Het systeem gaat al de content opslaan in objecten. Maar eZ publish maakt nog geen gebruik van een object geörienteerde database. Het gebruikt nog steeds een relationele database om de gegevens op te slaan. Daarom gebeurt het opslaan van gegevens via een complex netwerk van tabellen.
Ik tracht dit netwerk van relaties te verduidelijken voor de content module. De fanatici verwijs ik graag door naar de website van eZ publish waar ze een gedetailleerd schema kunnen zien van de hele structuur van de database.

U krijgt het relationele schema te zien van de tabellen van de content module.



Figuur 5.3 Het relationele model van de content module

De tabellen die centraal staan binnen de werking van de content module zijn *ezcontentobject*, *ezcontentobject_attribute*, *ezcontentclass* en *ezcontentclass_attribute*. Ik vertrek vanaf de tabel *ezcontentobject* om de verwerking van data te beschrijven.

Alle content wordt in eZ publish opgeslagen als een object, een contentobject. Een object kan meerdere attributen bevatten. Vandaar de 1 op ∞ relatie met *ezcontentobject_attribute*. Voor een content klasse trekt deze redenering zich door. Een klasse kan meerdere attributen hebben terwijl een attribuut telkens maar van 1 klasse is.

Een klasse definieert de structuur van een object door middel van de attributen. Daarom is een object een instantie van een klasse. De data zal zich dus niet in de klasse bevinden maar in het object. Omdat er meerdere objecten van een zelfde klasse kunnen zijn, ligt er een 1 op ∞ relatie tussen *ezcontentclass* en *ezcontentobject*. Ook hier trekt deze redenering zich weer door voor de attributen. Er kunnen meerdere objectattributen zijn van een klasse *attribuut*. Dit rechtvaardigt de relatie tussen de 2 tabellen.

Na het leggen van de vorige verbanden is de cirkel ‘rond’ voor wat de verhouding tussen klassen en objecten betreft. Maar er bevinden zich nog geen relaties tussen de nodes en de objecten. Daarvoor zorgen de tabellen *ezcontentobject_tree* en *eznode_assignment*.
Indien u hoofdstuk 5.3 gelezen hebt, dan weet u dat een object aan meerdere nodes kan geplaatst worden. Maar aan elke node kan maar 1 relatie naar een object zijn. Daarom ontstaat de relatie tussen een contentobject en de node assignment. Naast de tabel *eznode_assignment* is er ook de tabel *ezcontentobject_tree* die de relatie tussen objecten en nodes gaat bijhouden. Deze tabel wordt door het systeem gegenereerd en is gebaseerd op de tabel *eznode_assignment*.

U kunt ook zien dat er een recursieve koppeling is bij *eznode_assignment*. Die relatie toont dat een node een verwijzing kan hebben naar andere nodes.

Zoals u merkt worden er veel relaties tussen tabellen gelegd om informatie op te halen. U begrijpt dat het ophalen van informatie veel tijd in beslag kan nemen.
Omdat een snelle en eenvoudige export van de data noodzakelijk was voor mijn stageopdracht en omdat de overhead anders te groot werd, werd er dus besloten om zelf een tabel aan te maken. Ik moest geen gebruik maken van het bovenstaande principe om data in de juiste tabellen te plaatsen. Maar de manier waarop de data van mijn tabel gestockeerd wordt, verloopt volgens dezelfde werkwijze als dat eZ publish zijn content opslaat. Ik zal gebruik maken van de klasse *eZPersistentObject* om gegevens van de database te verkrijgen net zoals eZ publish dat doet. Deze klasse wordt u uitgelegd in hoofdstuk 8.

5.5 De ware kracht van eZ publish

Uit interesse naar de populariteit van eZ publish ben ik gestoten op een website waar de open source Content Management Systemen werden beoordeeld. Tot mijn verbazing trof ik eZ publish daar maar op de 21ste plaats in de ranglijst aan. (www.opensourcecms.com) Het bleek namelijk dat de hoger gequoteerde applicaties eenvoudiger en gebruiksvriendelijker waren.
Zij hebben dan wel het nadeel dat ze minder functionaliteiten hebben en dus ook minder mogelijkheden.
Hierin schuilt nu net de kracht van eZ publish.

Het heeft een flexibel ontwikkelingsframework dat het mogelijk maakt om zelf nieuwe functionaliteiten te ontwikkelen. Indien er een bepaalde functionaliteit niet aanwezig is in de standaard pakketten, dan kunt u deze zelf maken aan de hand van een nieuwe module.

Er werd ook de mogelijkheid voorzien om voor een klasse een nieuw datatype te maken.
Een klasse bestaat uit attributen. Elk attribuut heeft op zijn beurt een bepaald type. Het datatype zorgt voor de validatie van de ingevoerde gegevens en bewaart de gegevens in de database. U kunt bijvoorbeeld denken aan een klasse in Java die gebruik moet maken van datatypes zoals String, Integer of Boolean. Maar indien een bestaand datatype in eZ publish ontoereikend is, kan er een zelfgeschreven datatype gecreëerd worden.
Een voorbeeld van een bestaand datatype in eZ publish is XML TextField. Dit datatype maakt het mogelijk om een XML syntax te gebruiken in het attribuut.

Zelfs voor de templates zijn er uitbreidingen mogelijk gemaakt. Indien er in een template een functionaliteit ontbreekt, is er steeds de mogelijkheid om zelf een nieuwe template operator te programmeren. De template operator laat het template mechanisme toe om variabelen aan te passen of nieuw aan te maken.
Een voorbeeld van een bestaande template operator in eZ publish is count_chars. Deze template operator gaat het aantal letters tellen van een woord en dit tonen op het scherm.

Maar de grootste kracht van eZ publish is de variabele content structuur.
In tegenstelling tot andere Content Management Systemen maakt eZ publish geen gebruik van een voorgedefinieerd content model. Het laat de administrator de mogelijkheid om zijn eigen content structuur aan te maken. Op deze manier is het makkelijker om de aangepaste data te structureren, op te slaan, te verkrijgen en te presenteren. U moet het bijna onmogelijke doen om de data in een voorgedefinieerde structuur te plaatsen want nu kunt u de structuur aanpassen aan de data.
Niet elke gebruiker zal zijn eigen content structuur moeten maken. Daarom bevinden er in eZ publish al enkele ingebouwde structuren. Een voorbeeld van een ingebouwde structuur in eZ publish is de klasse Article waarin de gebruiker een nieuwsartikel kan opslaan.

Ik kon aan den lijve ondervinden wat dit krachtige systeem te bieden had. Zelf heb ik voor mijn stageopdracht een nieuwe module moeten creëren om nieuwe functies te voorzien om het logboek en de dosimetrie te beheren. Voorts heb ik nog een datatype *eZSchedule* en een template operator *drawExperimentSchedule* moeten aanpassen naar de nieuwe normen van mijn stageopdracht. In de volgende hoofstukken worden deze functionaliteiten verder uitgewerkt.

6 Waarom versiebeheer

Versiebeheer is de kunst van het beheren van aanpassingen aan bestanden. Deze tool werd vooral gebruikt door softwareontwikkelaars om hun aanpassingen aan bestanden bij te houden. Tegenwoordig makan niet alleen de softwareontwikkelaars gebruik van een versiebeheersysteem maar iedereen die het noodzakelijk acht om aanpassingen te beheren.
Versiebeheer biedt de gebruiker de zekerheid dat hij op elk moment kan terugkeren naar een vorige versie van zijn bestanden.
Een nauw aansluitend principe bij versiebeheer is releasebeheer. Wanneer de bestanden opgedeeld zijn in versies, kan er een beperking van een release worden bepaald. Er kan bijvoorbeeld worden bepaald dat enkel de bestanden van versie 2 gereleased mogen worden.
Op die manier kan de ontwikkelaar verder werken aan een nieuwe versie van zijn bestanden.

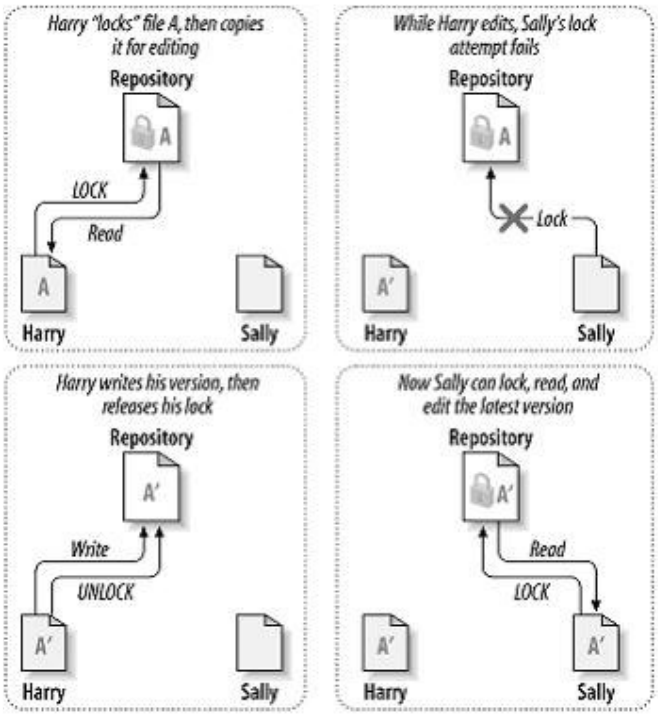
Op het SCK•CEN maakt men gebruik van Subversion als versiebeheersysteem. Subversion gaat voor elke gebruiker een repository aanmaken. Een repository is een gecentraliseerd systeem dat data kan opslaan. Het is vergelijkbaar met een file-share server. Subversion is een cliënt/server toepassing. Maar op de client zijde is het Subversion een command line programma. Daarom werd er TortoiseSVN ontwikkeld.
Dit programma kan bij Subversion geïnstalleerd worden zodat Subversion via een grafische interface benaderd kan worden.

Het volgende hoofdstuk is gebaseerd op informatie die te vinden is op de website van Subversion en TortoiseSVN.

6.1 Principe van een repository

Het probleem van een repository is meestal als volgt:
De personen Harry en Sally spreken eenzelfde bestand aan uit de repository. Ze bewerken beide dit bestand. Eerst slaat Harry zijn nieuw bestand op in de repository en vervolgens Sally. Het probleem hierbij is dat de aanpassingen van Harry nu verdwenen zijn omdat zijn bestand werd overschreven in de repository door het bestand van Sally.
Dit is een situatie die we willen vermijden.

Een oplossing voor dit probleem is het werken met een lock-modify-unlock mechanisme.
Als de repository dit mechanisme gebruikt dan kan er slechts één persoon aan een bestand. Zolang dat de persoon dit bestand aan het bewerken is, kan er niemand dit bestand openen. Het werd gelocked. Als het gewijzigde bestand bewaard wordt, dan wordt het ge-unlocked en kan vervolgens iedereen terug aan het bestand.



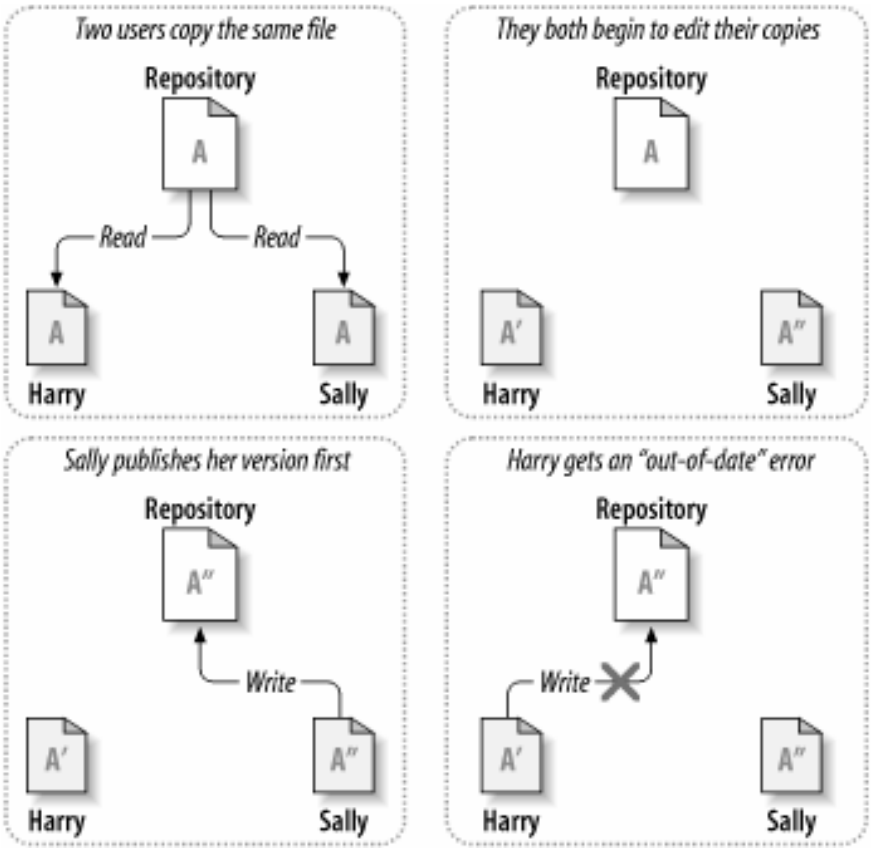
Figuur 6.1 Het lock-modify-unlock principe

Subversion maakt gebruik van nog een ander mechanisme om dit probleem te omzeilen. Het gaat gebruik maken van het copy-modify-merge mechanisme.

Wanneer een repository dit mechanisme gebruikt dan kunnen er meerdere personen een bestand lezen uit de repository. De gebruikers kunnen beide eenzelfde bestand bewerken zonder dat één van hen zijn informatie dreigt te verliezen. We bekijken dit principe weer met de personen Sally en Harry.

Sally en Harry kunnen beide een zelfde bestand aanspreken. Sally kan haar bewerkte bestand normaal bewaren maar Harry krijgt een out-of-date waarschuwing. Dit betekent dat het bestand dat Harry geopend heeft niet meer hetzelfde is dan het bestand waarover hij zijn nieuwe informatie wil wegschrijven. Vervolgens gaat Subversion voorstellen om de twee bestanden samen te voegen. Zo zullen de aanpassingen van Sally bewaard blijven en de aanpassingen van Harry worden toegevoegd aan het bestand.

Op de volgende pagina vindt u een schematische voorstelling van dit principe.



Figuur 6.2 Het copy-modify-merge principe

6.2 Subversion

Subversion is een versiebeheersysteem/releasebeheersysteem dat gebruik maakt van versie controle. De software is verkrijgbaar onder de open source vorm.

Hier kunt u enkele kenmerken van Subversion vinden:

- Versie controle

Subversion gaat niet enkel controleren of een bestand bestaat en of dat de inhoud gewijzigd is, maar het heeft ook de mogelijkheid om directories in zijn structuur op te nemen. Aan de hand van de controles worden er versies van de bestanden aangemaakt.

- Commit controle

Een handige controle ter bescherming van fouten is de commit controle. Er zal pas een commit van de gegevens worden uitgevoerd als alle gegevens gecommit kunnen worden. Wanneer er tijdens een commit van gegevens een fout ontstaat dan zullen ze niet worden toegevoegd.

- WebDAV/DeltaV protocol

Het WebDAV/DeltaV protocol is een uitbreiding voor het HTTP protocol. Gebruikers kunnen bestanden bewerken en beheren op webservers. Binnen Subversion wordt het WebDAV/DeltaV protocol gebruikt om de cliënt te vertellen welke bestanden er out of date zijn.

- Eenvoudig in gebruik

Er kan op een eenvoudige manier gewerkt worden met Subversion. Zo zal de toepassing duidelijk leesbare boodschappen tonen wanneer men iets wil committen. Ook wanneer het mis loopt tracht de toepassing zoveel mogelijk uitleg te verschaffen.

6.3 TortoiseSVN

TortoiseSVN is de cliënt toepassing van Subversion. Het is de grafische interface van Subversion op de client zijde. Met TortoiseSVN kunt u op eenvoudige wijze een bestand of directory toevoegen aan de repository. Verder kunt u er ook een checkout van de data van de repository mee doen.

Hier kunt u enkele kenmerken van TortoiseSVN vinden:

- Shell integratie

TortoiseSVN is integreerbaar in de Windows Shell. De werkbalk van TortoiseSVN wordt opgenomen bij de werkbalk van de Windows verkenner zodat u van de beide gebruik kunt maken.

- Iconen overzicht

Wanneer u een commit hebt gedaan van bestanden dan verschijnt er een icoontje bij de bestanden die u hebt toegevoegd aan de repository. Op die manier kunt u gemakkelijk de status van uw werkende kopie zien.

7 Conceptuele aanpak

In dit hoofdstuk wordt de aanpak beschreven die ik gebruikt hebt om mijn stageopdracht tot een goed einde te brengen. Mijn opdracht was een vervolg op een bestaand project namelijk EMS. Bij dit project werden de standaard tabellen van eZ publish gebruikt. Maar zoals reeds beschreven was dit project redelijk groot en zwaar en omdat er een eenvoudige export van de data beschikbaar moest zijn , werd er gekozen om zelf nieuwe tabellen te creëren. Dit werd gedaan om de overhead te beperken.

7.1 Analyse

Voordat ik aan mijn stageopdracht kon beginnen, moest er eerst een analyse van het probleem uitgevoerd worden.
Allereerst werd de stageopdracht i.v.m. het logboek geanalyseerd.

De uitkomst van de analyse resulteerde in de onderstaande tabel met attributen.

Logbook			
Veld	**Type**	**Verplicht**	**Key**
id	mediumint(9)	Ja	Primary Key
facility	varchar(40)	Ja	
date	date	Ja	
time	time	Ja	
t1	float(10,4)	Ja	
t2	float(10,4)	Ja	
t3	float(10,4)	Ja	
p1	float(10,4)	Ja	
p2	float(10,4)	Ja	
p3	float(10,4)	Ja	
f1	float(10,4)	Neen	
f2	float(10,4)	Neen	
freetext	tinytext	Neen	

Figuur 7.1 De tabel logbook

In hoofdstuk 3 wordt het logboek uitgelegd en daar kunt u zien dat er voor een irradiation facility een aantal parameters moet worden bijgehouden. Vandaar komen de attributen *facility*, t, p en f. In de attributen *t1*, *t2*, *t3* wordt de temperatuur van het bestraalde materiaal bijgehouden. De attributen *p1*, *p2*, *p3* gaat de druk bijhouden. Verder waren de attributen *f1* en *f2* vereist om de flow rate op te slaan.
Om de gegevens in hun context te plaatsen wordt er ook nog de datum en tijd opgeslagen.
Tot slot is er nog de mogelijkheid om eventuele opmerkingen toe te voegen.

Verder hadden de wetenschappers van de dienst Instrumentatie nog enkele functionele eisen zoals de mogelijkheid om een grafiek van de gegevens te bezichtigen. Daarnaast hadden ze ook graag een export van de gegevens naar Excel gehad.

Er werd ook nog een praktische eis gesteld. Een decimaal getal wordt in de database opgeslagen met een punt. Maar sommige wetenschappers vullen een decimaal getal in met een komma. Dit zou conflicten in de database geven. Vandaar moest ik voorzien dat de wetenschappers zowel een punt als komma voor een decimaal getal konden invoeren.

Na 4 weken werd het logboek afgewerkt. Vervolgens is er begonnen aan de analyse van de dosimetrie.

De uitkomst van deze analyse resulteerde in de onderstaande tabel met attributen.

Dosimeter			
Veld	**Type**	**Verplicht**	**Key**
id	mediumint(9)	Ja	Primary Key
batchnumber	varchar(20)	Ja	
serienumber	Int(11)	Ja	
z	float(8,2)	Ja	
r	float(8,2)	Neen	
angle	Int(3)	Neen	
a1	float(6,4)	Ja	
a2	float(10,4)	Ja	
a3	float(10,4)	Ja	
a4	float(10,4)	Ja	
dose	float(10,4)	Ja	
periodIdentifier	varchar(12)	Ja	
startperiod	int(12)	Ja	
endperiod	int(12)	Ja	

Figuur 7.2 De tabel dosimeter

In hoofdstuk 3 vindt u ook de stagebeschrijving van de dosimetrie. Er werd bepaald dat een dosimeter een aantal eigenschappen had die bijgehouden moeten worden. Deze eigenschappen zijn bijvoorbeeld het attribuut batchnumber en serienumber.
Daarnaast moet een dosimeter geplaatst worden in een irradiation facility. De plaatsing gebeurt door middel van de attributen z wat de hoogte in de container voorstelt, r wat de straal van de container voorstelt en tot slot angle om de hoek te bepalen binnen de container. Aan de hand van een combinatie van de 3 waarden bekomt de gebruiker 1 exacte positie in een irradiation facility.
Voorts moeten er ook waarnemingen worden opgeslagen in de tabel. Hiervoor zijn de attributen *a1* tot *a4*. Deze attributen gaan de hoeveelheid absorptie van een dosimeter bepalen. Verder wordt de hoeveelheid dosis ook opgeslagen. Hoewel het attribuut *dose* een berekende waarde is, moet het toch in de tabel worden opgenomen. De dosis wordt berekend via de gemiddelde absorptiewaarde die wordt ingevuld in een vergelijking. Maar omdat deze vergelijking kan veranderen, wordt de dosis ook opgeslagen.
Aanvankelijk werd er gedacht dat de dosimetrie ook aan EMS gekoppeld moest worden omdat de wetenschappers graag de totale dosimetrie tijd van een dosimeter hadden geweten. Er werd verondersteld dat de tijd uit EMS kon gehaald worden maar uiteindelijk bleek dat de wetenschappers zelf de periode invullen. Om dit op te vangen bevinden zich de attributen *startperiod* en *endperiod* in de tabel.
Het laatste attribuut en ook één van de belangrijkste is *periodIdentifier*. Dit attribuut gaat ervoor zorgen dat er een relatie is tussen een dosimetrie en een dosimeter.

7.2 Principe van een module

Zoals al vermeld werd er gekozen om een nieuwe tabel te creëren. Er moest bijgevolg een module gemaakt worden om deze tabel te kunnen aanspreken.

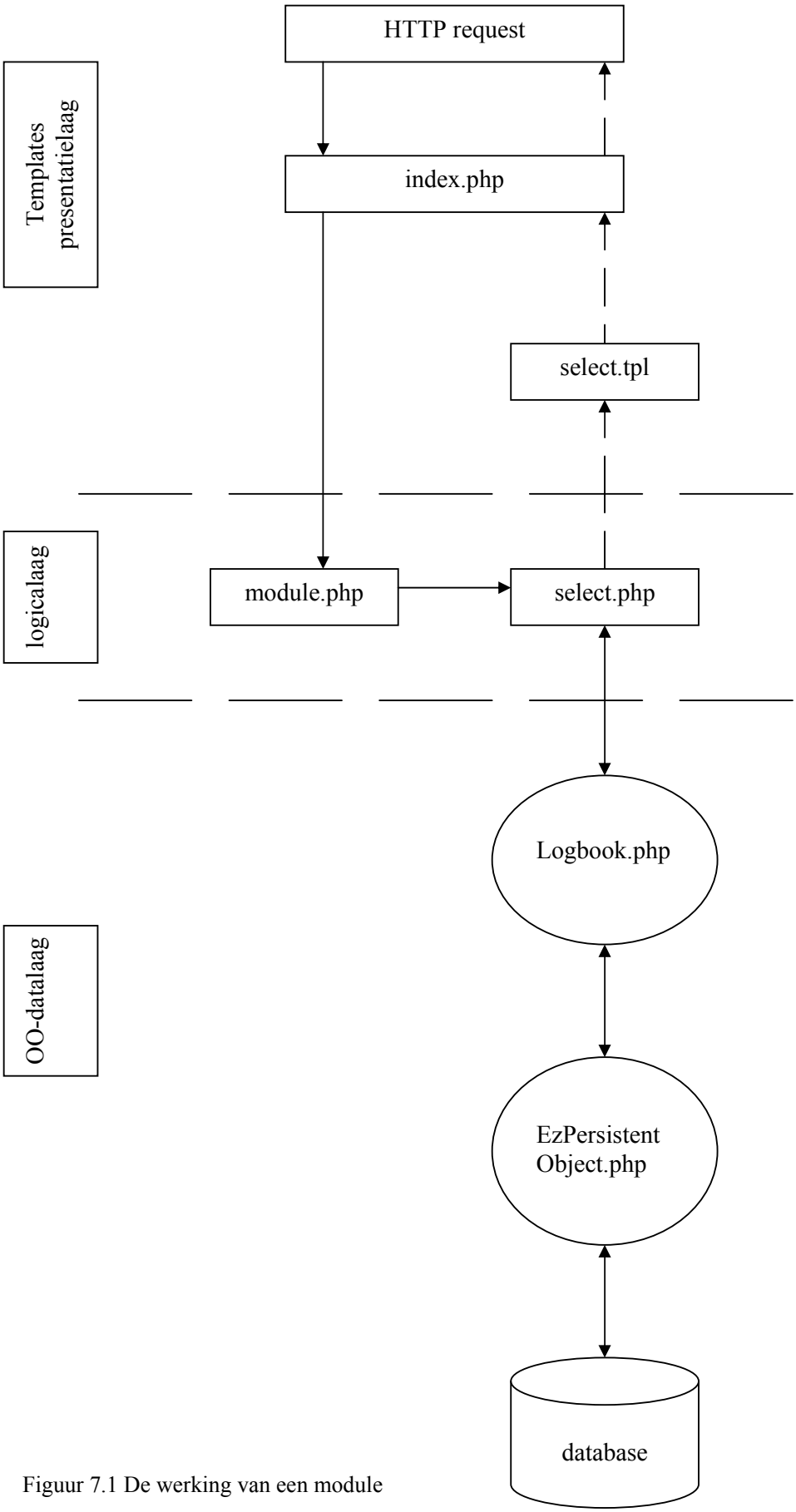
Het principe van een nieuwe module zal u worden voorgesteld aan de hand van een voorbeeld. U kunt de beschrijving van het principe meevolgen op de figuur 7.1.

Allereerst is er een gebruiker die naar de URL van de nieuwe module surft. De DNS van de gebruiker zal vertellen dat dit het IP adres is van bijvoorbeeld de server Terra van het SCK•CEN.

Vervolgens wordt de Apache webserver gedetecteerd die daarna zijn configuratiebestand zal aanspreken. Er wordt naar de sectie VirtualHost van het configuratiebestand gezocht en gecontroleerd of de URL open staat. Wanneer dit in orde is zal Apache naar de directory en het startbestand van de VirtualHost configuratie gaan. Bij een eZ publish installatie is dit startbestand index.php.

Index.php zal ervoor zorgen dat de pagelayout template geladen wordt. Verder gaat index.php op zoek naar het bestand module.php van de gevraagde module. Hierin bevinden zich alle views van de module. Met het woord view wordt bedoeld dat er een bestand moet zijn dat naar de templates gaat verwijzen wanneer een gebruiker een webpagina opvraagt. eZ publish gaat namelijk in module.php kijken of er een script is dat noemt zoals de opgevraagde web pagina.
Wanneer er een view bestaat van de opgevraagde pagina dan wordt eerst het PHP-bestand geladen. Wanneer er data uit de database gehaald moet worden, maakt men gebruik van voorgeprogrammeerde klassen. Om gebruik te maken van mijn tabel, moest ik eerst zelf een klasse aanmaken. Deze klasse *Logbook* implementeert de klasse *eZPersistentObject*. Op die manier erft de klasse *Logbook* over van *eZPersistentObject* zodat al de functies van *eZPersistentObject* ook beschikbaar zijn in de klasse *Logbook*. Deze klasse maakt het nu mogelijk om informatie uit een database te halen, te verwijderen en toe te voegen. Zoals u misschien merkt bevinden deze klassen zich in de OO-datalaag. In de logicalaag, binnen het PHP-bestand, wordt de verwerking van de parameters verder afgehandeld zodat het een bruikbaar geheel vormt voor de templates.

Vervolgens zal dat PHP-bestand de overgang naar de templates presentatielaag maken. Het bestand zal de overeenkomstige template laden. Wanneer de template door index.php geladen wordt, zal de gebruiker zijn gevraagde webpagina kunnen bezichtigen.



Figuur 7.1 De werking van een module

8 Klassen

8.1 De klasse Logbook

Om gebruik te maken van de tabel *logbook* moet er eerst een klasse aangemaakt worden. Deze klasse kan omdat ze van de klasse *eZPersistentObject* overerft, de tabel *logbook* aanspreken. Dit kunt u zien door het woord extends.

```
include_once( 'kernel/classes/ezpersistentobject.php' );

class Logbook extends eZPersistentObject
{
```

De belangrijkste functie binnen deze klasse is de functie *definition*. Het is namelijk aan de hand van deze gegevens dat de klasse *eZPersistentObject* SQL-instructies kan construeren. Deze SQL-instructies worden vervolgens naar de database gestuurd om de tabel *logbook* aan te passen of te ondervragen.

In de functie *definition* worden onder andere al de namen van de kolommen van de tabel opgesomd. Verder wordt er ook nog het type van de kolom opgegeven en of deze kolom verplicht is of niet. Al deze informatie wordt in de array fields gestopt.

U kunt zien dat er voor de naam *definition* een ampersand staat. Dit wil zeggen dat deze functie kan worden opgeroepen zonder dat de klasse geïnstancieerd moet worden. In paragraaf 2 vindt u een voorbeeld van een klasse die gebruik maakt van de statische functie *definition*.

```
function &definition()
{
  return array( 'fields' => array( 'facility' => array(
'name' => 'Facility',
'datatype' => 'string',
'required' => true ),
                                    'date' => array(
'name' => 'Date',
'datatype' => 'string',
'required' => true ),
```

In hoofdstuk 7 vindt u een lijst met al de attributen van de tabel *logbook*. Ook deze attributen werden in de functie *definition* opgenomen maar worden hier niet verder uitgewerkt.

Naast de velden bevat de functie *definition* ook nog array's waarmee de primary key van de tabel wordt aangeduid, welke de naam van de klasse is en wat de naam van de tabel is. Er kan ook nog de optionele parameter sort meegegeven worden. Deze waarde zal in de SQL-instructie als 'ORDER BY' komen te staan.

```
      'keys' => array( 'id' ),
      'class_name' => 'Logbook',
      'sort' => array( 'time' => 'asc' ),
      'name' => 'logbook' );
}
```

Verder staan er nog 2 functies in de klasse *Logbook*.
De eerste functie is *fetchData*. Deze functie gaat er voor zorgen dat er informatie uit de database gehaald kan worden door middel van de functie *fetchObjectList* van *eZPersistentObject*. De parameter die meegegeven wordt aan de functie *fetchData* wordt binnen de functie verwerkt. De waarden moeten op een correcte manier opgeslagen worden in een array zodat *fetchObjectList* een juiste where-clausule kan maken om de data op te halen.
U krijgt in de code te zien hoe de parameter *facility* wordt opgeslagen in een array zodat de nieuwe array $where kan worden meegegeven aan de functie *fetchObjectList*. Indien er geen parameters worden meegegeven zal *eZPersistentObject* alle rijen uit de database teruggeven.

Naast de parameters voor de where-clausule moet er zeker nog de definitie van de klasse meegegeven worden. De andere parameters om de sortering van de SQL-instructie te maken of een vast aantal rijen terug te verkrijgen, zijn optioneel.

```
function &fetchData( $params )
{
     $where = array();

     if( isset( $params['Facility'] ) && ! is_null(
$params['Facility'] ) )
     {
         $facility = array( 'facility' => $params['Facility'] );
         $where = array_merge( $facility, $where );
     }

     if( isset( $params['Offset'] ) && ! is_null( $params['Offset']
) && isset( $params['Length'] ) && ! is_null( $params['Length'] ) )
     {
         $limit = array( 'offset' => $params['Offset'], 'length' =>
$params['Length'] );
     }
     else
     {
         $limit = null;
     }

     $sorts = array( 'date' => 'asc', 'time' => 'asc' );

     return eZPersistentObject::fetchObjectList(
Logbook::definition(), null, $where, $sorts, $limit );
}
```

Hieronder vindt u het resultaat van de functie *fetchData* wanneer er op id = 100 werd gezocht.

```
array(1) {
  ["result"]=>
  array(1) {
    [0]=>
    object(logbook)(14) {
      ["PersistentDataDirty"]=>
      bool(false)
      ["Facility"]=>
      string(8) "brigitte"
      ["Date"]=>
      string(10) "2005-04-12"
      ["Time"]=>
      string(8) "06:00:00"
      ["T1"]=>
      string(7) "25.0000"
      ["T2"]=>
      string(7) "24.6000"
      ["T3"]=>
      string(7) "24.3000"
      ["P1"]=>
      string(7) "50.0000"
      ["P2"]=>
      string(7) "52.0000"
      ["P3"]=>
      string(7) "51.0000"
      ["F1"]=>
      string(6) "0.0000"
      ["F2"]=>
      string(6) "0.0000"
      ["Freetext"]=>
      string(0) ""
      ["ID"]=>
      string(3) "100"
    }
  }
}
```

De andere functie binnen de klasse *Logbook* is *removeData*. Deze functie maakt gebruik van de functie *removeObject* van *eZPersistentObject* om een rij uit de database te wissen.
Net zoals bij de functie *fetchObjectList* moet er de definitie van de klasse meegegeven worden. Daarnaast kan er ook een parameter meegegeven worden die als where-clausule van de SQL-instructie gaat dienen. Deze parameter wordt opgebouwd aan de hand van de verkregen parameter van de functie *removeData*.

```
function removeData( $params )
{
    $conditions = array();
    if( isset( $params['Id'] ) and ! is_null( $params['Id'] ) )
{
    $id = array( 'id' => $params['Id'] );
    $conditions = array_merge( $id, $conditions );
}
    return eZPersistentObject::removeObject( Logbook::definition(),
$conditions );
}
}
```

8.2 De klasse Dosimeter

Nadat het logboek gemaakt was, kon ik beginnen aan de tabel *dosimeter*. Eerst werd de klasse *Dosimeter* aangemaakt. Net zoals de vorige klasse *Logbook* erft ook *Dosimeter* over van *eZPersistentObject*. Voorts krijgt u ook de contructor van de klasse te zien. U kunt opmerken dat er een rij als parameter moet worden meegegeven zodanig dat *eZPersistentObject* via die rij gegevens kan ophalen.

```
include_once( 'kernel/classes/ezpersistentobject.php' );

class Dosimeter extends eZPersistentObject
{

function Dosimeter( $row = array() )
{
    $this->eZPersistentObject( $row );
}
```

Bij elke klasse die van *eZPersistentObject* overerft is de functie *definition* de belangrijkste. Er worden al de attributen van de tabel *dosimeter* beschreven, zodat de klasse *eZPersistentObject* een SQL-instructie kan maken om gegevens uit deze tabel op te vragen.

Hier worden enkele attributen zoals *id, batchnumber* en *serienumber* in een array geplaatst. Achter elk attribuut worden ook de eigenschappen van dat attribuut in een array opgeslagen. Zo kunt u zien dat het attribuut *id* van het type integer is en verplicht is in te vullen.

```
function &definition()
{
  return array( 'fields' => array( 'id' => array( 'name' => 'Id',
                                                  'datatype' =>
'integer',
                                                  'required' => true
),
                                   'batchnumber' => array( 'name' =>
'Batchnumber',

'datatype' => 'string',
'required' => true ),
                                   'serienumber' => array( 'name' =>
'Serienumber',
'datatype' => 'integer',
'required' => true )
```

De overige attributen kunt u aantreffen in hoofdstuk 7. Ook deze attributen werden in de functie *definition* gedeclareerd maar worden verder niet meer uitgelegd.

Om de functie *definition* te vervolledigen moeten er nog enkele instellingen gekend zijn. De klasse *eZPersistentObject* moet weten uit welke tabel het informatie kan halen en welke de primary key is van de tabel. Daarnaast wordt ook nog de naam van de klasse meegedeeld en de sorteringswijze van de verkregen rijen data. Deze informatie wordt gekend door de code op de volgende pagina.

```
        'keys' => array( 'id' ),
        'increment_key' => 'id',
        'class_name' => 'Dosimeter',
        'sort' => array( 'id' => 'asc' ),
        'name' => 'dosimeter' );
    }
```

Om informatie uit de database te halen, werd de functie *fetchData* aangemaakt. Er wordt een parameter meegegeven om de data te filteren. De parameter kan zowel het attribuut *id* als *periodIdentifier* bevatten om een selectie van de data te maken. Wanneer één van deze attributen wordt opgegeven als parameter wordt de waarde in de array $where geplaatst.
Vervolgens wordt de functie *fetchObjectList* van *eZPersistentObject* opgeroepen met de definitie en de variabele $where als parameters om de data uit de database te halen.

```
function &fetchData( $params )
{
    $id = '';
    $periodIdentifier = '';

    $where = array();

    if( isset( $params['Id'] ) && ! is_null( $params['Id'] ) )
    {
        $id = array( 'id' => $params['Id'] );
        $where = array_merge( $id, $where );
    }

    if( isset( $params['PeriodIdentifier'] ) && ! is_null(
$params['PeriodIdentifier'] ) )
    {
        $periodIdentifierArray = array( 'periodidentifier' =>
$params['PeriodIdentifier'] );
        $where = array_merge( $periodIdentifierArray, $where );
    }

    return eZPersistentObject::fetchObjectList(
Dosimeter::definition(), null, $where );
}
```

Het resultaat van *fetchData* naar id 10 geeft de volgende output weer.

```
array(1) {
  ["result"]=>
  array(1) {
    [0]=>
    object(dosimeter)(17) {
      ["PersistentDataDirty"]=>
      bool(false)
      ["Id"]=>
      string(2) "10"
      ["Batchnumber"]=>
      string(1) "Z"
      ["Serienumber"]=>
      string(8) "14796852"
      ["Type"]=>
      NULL
      ["Z"]=>
      string(5) "15.60"
      ["R"]=>
      string(4) "0.00"
      ["Angle"]=>
      string(1) "0"
      ["Width"]=>
      string(6) "4.0000"
      ["A1"]=>
      string(6) "2.3000"
      ["A2"]=>
      string(6) "2.4500"
      ["A3"]=>
      string(6) "2.0000"
      ["A4"]=>
      string(6) "2.3600"
      ["Dose"]=>
      string(8) "314.1152"
      ["PeriodIdentifier"]=>
      string(6) "6537P1"
      ["StartPeriod"]=>
      string(10) "1114603200"
      ["EndPeriod"]=>
      string(10) "1114664400"
    }
  }
```

Omdat de dienst Instrumentatie de id van een dosimeter moet doorgeven aan de afdeling BR2, moet er ook een functie zijn die een nieuwe id gaat creëren.
De functie *getNewID* gaat gebruik maken van de functie *newObjectOrder* van de klasse *eZPersistentObject*. Deze functie gaat de maximum waarde van een bepaald veld opvragen, verhogen met 1 en dan terugsturen. Om ook hier de SQL-instructie te kunnen aanmaken, moet de definitie als parameter worden meegegeven. De tweede parameter is het attribuut waarvan de maximumwaarde moet gezocht worden. De laatste parameter is een conditie die kan worden meegegeven.

Wanneer de functie *getNewID* wordt opgeroepen, zal altijd de SQL-instructie ‘SELECT MAX(id) FROM dosimeter’ worden uitgevoerd.

```
function getNewID()
{
    return eZPersistentObject::newObjectOrder(
Dosimeter::definition(), 'id', null );
}
```

Er werd nog een derde tabel *functionparameters* en dus ook een klasse gecreëerd. Maar de werking van deze klasse is identiek aan de werking van degene die al besproken zijn. Daarom wordt deze klasse niet besproken.

9 Module EMS DATA

Hoewel de stageopdracht uit 2 delen bestond, werden zowel de functionaliteiten voor het logboek en de dosimetrie gebundeld in 1 module. De stageopdracht vormde een aanvulling voor EMS. Er moest een module gemaakt worden die de data van EMS ging beheren. Logischerwijze werd de module EMS DATA genoemd.

Allereerst ben ik begonnen aan het maken van het logboek omdat dit qua analyse de eenvoudigste opdracht was. Na ongeveer een maand werk ben ik overgeschakeld naar het dosimetriebeheer.

9.1 Logboek

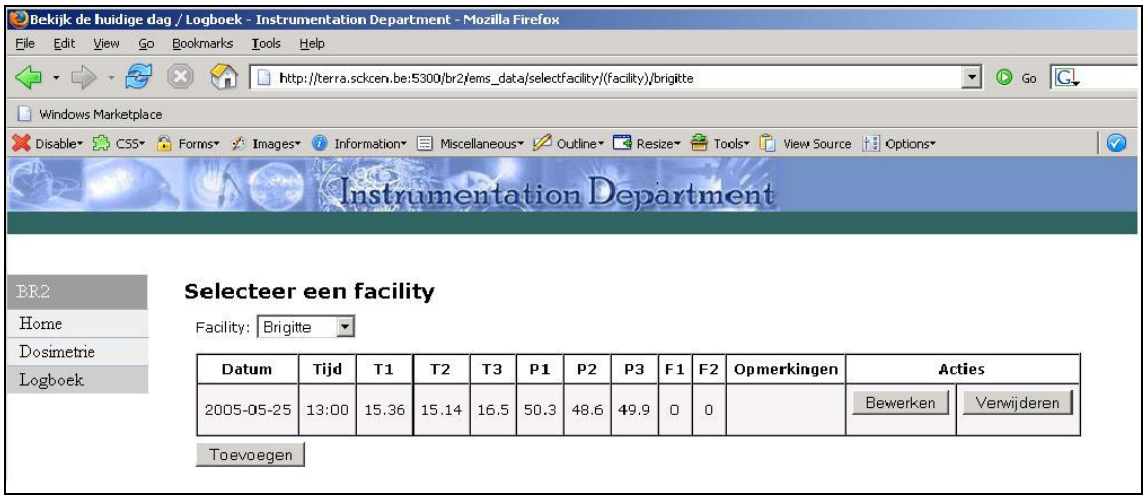
De werking van het logboek zit als volgt in mekaar:
De dienst Instrumentatie zal op EMS een experiment plannen. Maar de uitvoering wordt niet door de dienst Instrumentatie gedaan maar door de afdeling BR2. Het zal op de afgesproken datum het experiment starten. Wanneer er zich een irradiation voordoet, noteren wetenschappers de parameters die ze van enkele sensoren kunnen aflezen.
Normaal wordt er om de 6 uur een waarneming genomen. Na verloop van tijd worden de waarnemingen in EMS ingebracht via de interface *selectfacility* en *editentry*.
Wanneer een experiment aan de gang is, kan de dienst Instrumentation op elk ogenblik de waarnemingen bekijken via een klik op de grafiek van de planning.
Er wordt van de waarnemingen een grafiek gemaakt zodat de wetenschappers een snel oordeel kunnen vellen. Indien ze de resultaten wat nader willen bekijken kunnen ze een export van de gegevens naar Excel doen. Deze laatste handelingen gebeuren allemaal op de interface *viewlogbook*.

Allereerst ben ik van start gegaan met het maken van de interface *selectfacility* waar de wetenschappers van de afdeling BR2 eerst een facility kunnen selecteren. Vervolgens krijgen ze de resultaten van de afgelopen 24 uur te zien.

9.1.1 View selectfacility

Het uitgangspunt van de interface *selectfacility* was enerzijds om de wetenschappers van de afdeling BR2 een waarneming te laten toevoegen en anderzijds om hen een overzicht van de waarnemingen van de afgelopen 24 uur te tonen. De gebruiker moet eerst een selectie maken en vervolgens wordt hij aan de hand van een druk op een knop naar een andere pagina gestuurd. Daar zal hij nieuwe metingen kunnen registreren voor die irradiation facility.

De interface in de browser ziet eruit als volgt:



Figuur 9.1 De interface van de view selectfacility

De onderstaande code vindt u in module.php. Het bepaalt de naam van de view, namelijk *selectfacility* en de naam van welk script er geladen moet worden.
Tot slot ziet u nog de parameters ‘single_post_actions’ en ‘post_action_parameters’. De parameter single_post_actions gaat de knoppen van de template associëren met een waarde. Zo kunt u zien dat de knop ‘btnRemove’ de waarde remove krijgt. Wanneer er op een knop wordt geklikt, zal er in de parameter post_action_parameters naar de waarde remove gezocht worden. Vervolgens wordt de parameter id opgevuld met de waarden van het tekstvak ‘txtId’. Op deze manier worden de parameters van een template bekend in een PHP-script.

U treft in de viewlist ook nog de parameter *functions* aan. Deze parameter wordt gelinkt aan de functionlist. De functionlist is noodzakelijk om rechten op de module te geven.
Er is voor zowel de afdeling BR2 als de dienst Instrumentatie een nieuwe rol gecreëerd binnen EMS. Aan deze rollen wordt dan de module en de rechten ‘edit’ of ‘read’ toegekend. Vervolgens moet er nog bepaald worden welke gebruikers tot welke rol behoren. Het instellen van de rechten kan in de administrator interface gedaan worden.

Omdat dit voor alle interfaces ongeveer hetzelfde is, wordt het voor de rest van het eindwerk niet meer besproken.

```
$ViewList['selectfacility'] = array(
        'functions' => array( 'edit' ),
        'script' => 'selectfacility.php',
        'single_post_actions' => array( 'btnAdd' => 'add', 'btnEdit'
=> 'edit', 'btnRemove' => 'remove' ),
        'post_action_parameters' => array( 'add' => array(),

'remove' => array( 'id' => 'txtId' ) )
    );

$FunctionList['edit'] = array();
$FunctionList['read'] = array();
```

Van de achterliggende template code belicht ik enkel de meest interessante stukken.

```
<form action={"content/action"|ezurl} method="post" id="frmFirst">
<div>
    <input type="hidden" name="DestinationURL"
value="ems_data/selectfacility" />
    {let facilities=fetch( 'content', 'list', hash( parent_node_id,
6448, sort_by, array( array( "name", true() ) ) ) )}
    <div class="textfield_space">Facility:
    <select name="(facility)" onchange="submit('frmFirst')">
        <option value="none"></option>
        {section var=fac loop=$facilities}
            {section show=eq(
$fac.item.object.data_map.identifier.content, $facility )}
                <option
value="{$fac.item.object.data_map.identifier.content}"
selected="selected">{$fac.item.name}</option>
            {section-else}
                <option
value="{$fac.item.object.data_map.identifier.content}">{$fac.item.na
me}</option>
            {/section}
        {/section}
    </select>
    </div>
</div>
</form>
```

In de bovenstaande code wordt de dropdownlijst gecreëerd.
De dropdownlijst is toch anders dan een normale dropdownlijst. Hier wordt gebruik gemaakt van een techniek genaamd HTTP Post to URL converter.
Deze techniek maakt het mogelijk om een waarde in de template te selecteren en die waarde als parameter in de URL te plaatsen.
In figuur 9.1 kunt u een voorbeeld van deze techniek zien. U kunt zien dat de parameter ‘facility’ tussen haken in de URL verschijnt samen met zijn waarde *Brigitte* in dit geval.

Verder is er nog een veiligheidscontrole in de pagina ingebouwd. Wanneer een gebruiker een rij wil verwijderen, krijgt hij eerst een messagebox om een bevestiging te geven. Op deze manier kunnen er geen gegevens per ongeluk verwijderd worden.

Het achterliggende script ziet er als volgt uit:

```
<?php
    include_once( 'kernel/common/template.php' );
    ext_class( 'ems_data', 'logbook' );
    include_once( 'lib/ezlocale/classes/ezdate.php' );
    include_once( 'lib/ezlocale/classes/eztime.php' );
    include_once( 'lib/ezlocale/classes/ezlocale.php' );
    include_once(
'extension/ems_data/modules/ems_data/customdatetimeformat.php' );
```

Bovenstaande regels zijn nodig om een aantal klasse en methodes te importeren waarop deze functie steunt. De meeste klassen zijn klassen van eZ publish zelf maar er zijn ook enkele zelfgedefinieerde klassen bij zoals *CustomDateTime* en *Logbook*.

```
$tpl =& templateInit();

$Module =& $Params['Module'];
$button = $Module->currentAction();

if( $button == 'remove' )
{
    if( $Module->hasActionParameter( 'id' ) )
    {
        $params = array( 'Id' => $Module->actionParameter( 'id' ) );
        Logbook::removeData( $params );
    }
}
```

Als eerste wordt er het templateobject $tpl geïnitialiseerd.
Vervolgens ziet u hoe de parameters van de module in de variabele $Module geladen worden. Daarna wordt er de parameter $button geladen met de knop waarop de gebruiker geklikt heeft. Deze parameter is terug te vinden in het script module.php onder de benaming single_post_actions.

Als $button de waarde van de remove-knop heeft, wordt de klasse *Logbook* aangesproken om de rij te verwijderen.

```
$parameter =  $Params[ 'UserParameters' ];

if( isset( $parameter[ 'facility' ] ) or ! is_null( $parameter[
'facility' ] ) )
{
    $tpl->setVariable( 'facility', $parameter[ 'facility' ] );
}
```

In de bovenstaande code vindt u de verwerking van de parameter die door middel van de HTTP Post to URL converter wordt doorgegeven.

Verder worden er in de tabel alle gegevens getoond van de voorbije 24 uur. Dit gebeurt door de volgende code.

```
$today = new eZDate();
$yesterday = new eZDate();
$yesterday->adjustDate( 0, -1 );

$time = new eZTime();

$params = array( 'Facility' => $facility, 'StartDate' =>
CustomDateTime::writeDate( $yesterday ), 'EndDate' =>
CustomDateTime::writeDate( $yesterday ), 'StartTime' =>
CustomDateTime::writeTime( $time ) );
$datayesterday =& Logbook::fetchData( $params );

$params = array( 'Facility' => $facility, 'EndDate' =>
CustomDateTime::writeDate( $today ) );
$datatoday =& Logbook::fetchData( $params );

$table = array_merge_recursive( $datayesterday, $datatoday );
```

```
$tpl->setVariable( 'table', $table );
```

U kunt bijvoorbeeld zien dat de functie *writeDate* wordt aangesproken. Deze functie bevindt zich in de zelf gemaakte klasse *CustomDateTime*. Deze klasse werd aangemaakt omdat er binnen eZ publish geen klasse aanwezig was die de datum en tijd op een correcte wijze kon weergeven.

Voorts kunt u zien dat de database 2 keer wordt aangesproken om de gegevens op te halen. Dit gebeurt door middel van het aanroepen van de klasse *Logbook* en de functie *fetchData*. De database werd 2 keer aangesproken om de gegevens van de afgelopen 24 uur op te halen. Er kon geen SQL-instructie gemaakt worden die zowel de gegevens van gisteren na het huidige uur en de gegevens van vandaag kon ophalen. Wanneer het momenteel 2 uur is, moet ik al de gegevens van gisteren ophalen later dan 2 uur en de gegevens van vandaag. Indien deze selectie in een SQL wordt geplaatst, krijg ik enkel de gegevens van gisteren en vandaag na 2 uur.
De SQL zag er dan als volgt uit:

SELECT * FROM logbook WHERE date BETWEEN ‘yesterday’ AND ‘today’ AND time > 2;

Daarom worden eerst de gegevens van gisteren opgehaald en vervolgens die van vandaag. Tot slot worden de array’s samengevoegd om de totale gegevens te verkrijgen.

Tot slot wordt de template geladen en kunnen de geplaatste parameters getoond worden.

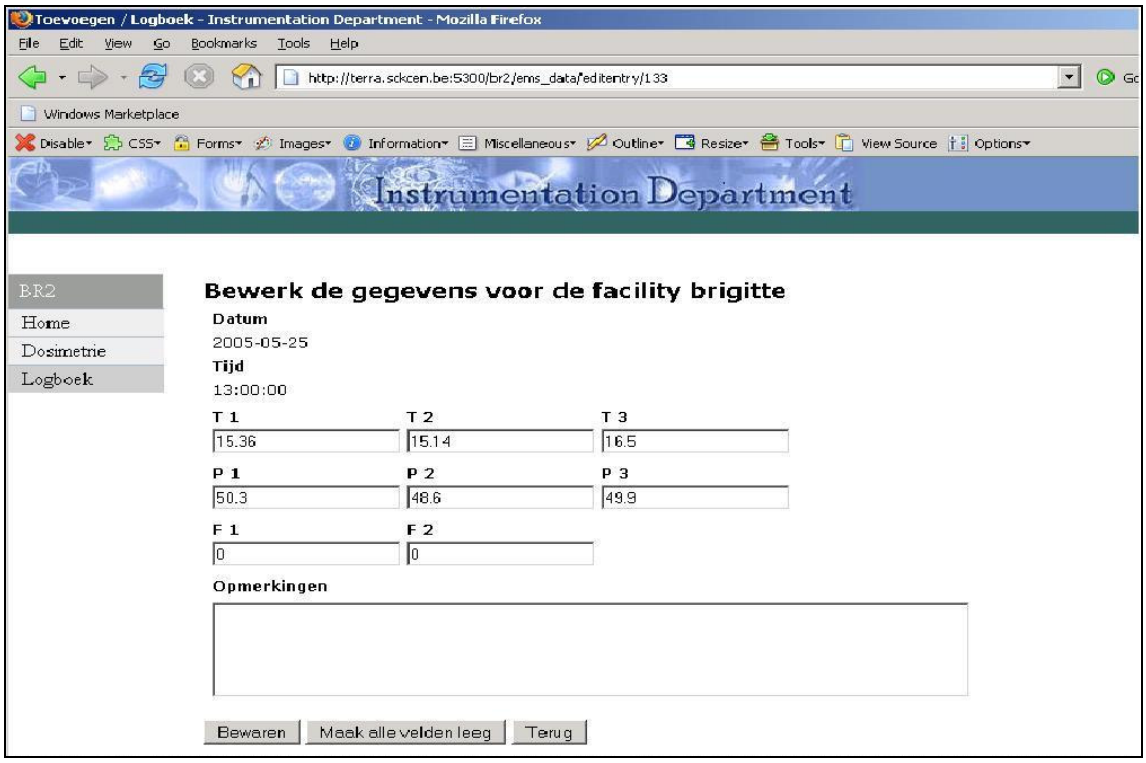
```
$Result = array();
$Result['content'] = $tpl->fetch(
'design:ems_data/selectfacility.tpl' );
$Result['path'] = array( array( 'url' => false, 'text' => 'Logboek'
),
                         array( 'url' => false, 'text' => 'Bekijk de
huidige dag' ) );
?>
```

Nadat ik deze interface had afgewerkt, kon ik beginnen aan de interface waar de wetenschappers hun waarnemingen konden in wegschreven.

9.1.2 View editentry

Na het klikken op knop ‘toevoegen’ of ‘bewerken’ van de vorige interface komt de wetenschappers aan in de view *editentry*. Hier kan hij een nieuwe waarneming toevoegen of een bestaande wijzigen.

De interface in de browser kunt u zien op de volgende pagina:



Figuur 9.2 De interface van de view editentry

Net zoals in de vorige interface ga ik ook hier enkel de meest interessante stukken aankaarten. De moeilijkheid aan deze interface was om een onderscheid te maken wanneer er een rij bewerkt wordt of wanneer er een nieuwe rij toegevoegd wordt. Voorts zit er ook nog een uitgebreide foutcontrole in zodat de gebruiker op een leesbare manier weet hoe hij zijn fout kan oplossen.

In de onderstaande code vindt u een afhandeling van de foutboodschap wanneer er een foutieve invoer van een temperatuur was.

```
{section show=and( $t1valid, $t2valid, $t3valid )|not}
    {let text="U moet een getal invullen voor: "}
    {let counter=0 fields=array( $t1valid, $t2valid, $t3valid )
messages=array( "temperatuur 1", "temperatuur 2", "temperatuur 3" )}
    {section var=field loop=$fields}
        {section show=$field|not}
            {section show=eq( $counter, 0 )}
                 {set text=concat( $text,  $messages[$field.index]
)}
            {section-else}
                {set text=concat( $text, ", ",
$messages[$field.index )}
            {/section}
            {set counter=$counter|inc}
        {/section}
    {/section}
    {/let}
    <span class="false_input">{$text}</span>
```

```
    {/let}
{/section}
```

In de template wordt er nagegaan of de parameter goed is ingevuld. Dit gebeurt aan de hand van de verkregen variabele, bijvoorbeeld $t1valid. Wanneer de parameter niet correct is ingevuld, wordt hij aan de lijst toegevoegd als foutieve ingave. Op het einde wordt de lijst met foutieve ingaven afgedrukt.

Figuur 9.3 De foutboodschap van editentry

Verder wordt in de onderstaande scripts de verwerking van deze template uit de doeken gedaan.

Eerst wordt er een templateobject geïnitialiseerd. Daarnaast worden er ook enkele imports uitgevoerd om enkele zelfgeschreven klassen of klassen van eZ publish te laden. De klassen bevatten functies die noodzakelijk zijn om de view *editentry* correct te laten functioneren.

```
<?php
include_once( 'kernel/common/template.php' );
ext_class( 'ems_data', 'logbook' );
include_once(
'extension/ems_data/modules/ems_data/controlfields.php' );

$tpl =& templateInit();

$Module =& $Params['Module'];
$button = $Module->currentAction();
```

Zoals ik al vermeld heb, was er een moeilijkheid om na te gaan of er een rij bewerkt of toegevoegd wordt. Deze afhandeling wordt in het script gedaan.
Er wordt nagegaan of de verkregen parameter een id of een facility is. Wanneer het script een id verkrijgt, wordt er een rij bewerkt. In het andere geval gaat de gebruiker een nieuwe rij toevoegen.

```
$facility = $Params['Parameters'][0];
$entry = new Logbook();

if( is_numeric( $facility ) )
{
    $tpl->setVariable( 'edit', 'true' );
    $tpl->setVariable( 'id', $facility );

    $params = array( 'Id' => $facility );
    $data =& Logbook::fetchData( $params);

    if( count( $data ) != 0 )
    {
        $entry = $data[0];
        $tpl->setVariable( 'selection', $data[0]->attribute(
'facility' ) );
```

```
    }
    else
    {
        $entry = array();
        $tpl->setVariable( 'recordexists', false );
    }
}
```

U kunt in het bovenstaande script opmerken dat eerst de parameter uit de URL wordt opgehaald. Daarnaast wordt er ook de variabele $entry gedeclareerd als een object van de klasse *Logbook*. Wanneer we te maken hebben met een id, wordt er aan de template via een de variabele *edit* verteld dat de gegevens bewerkt worden. Vervolgens wordt de gekeken of er een rij in de database bestaat voor de id. U kunt dit zien doordat de klasse *Logbook* wordt aangesproken met de functie *fetchData*. Wanneer er zich een rij in de database bevindt, wordt de geselecteerde facility naar de template verstuurd. U ziet ook dat de variabele $entry de waarde van de geselecteerde gegevens krijgt. De variabele wordt onderaan het script naar de template gestuurd.

Als de gebruiker een nieuwe rij wil toevoegen, wordt het onderstaande script uitgevoerd.

```
if( is_numeric( $facility ) )
{
    …
}
else
{
    $tpl->setVariable( 'selection', $facility );
    $entry->setAttribute( 'facility', $facility );
    $date = date('Y-m-d');
    $time = date( 'H' ) . ':00';
    $entry->setAttribute( 'date', $date );
    $entry->setAttribute( 'time', $time );
}
```

Als er een nieuwe rij wordt toegevoegd moeten er alleen enkele standaardwaarden in de variabele $entry geplaatst. $entry is een object van de klasse *Logbook* dat op het einde van het script naar de templates wordt gestuurd.

Als de gebruiker op de knop ‘Bewaren’ klikt, wordt er nagegaan of de parameters voldoen aan de vereisten.
Wanneer alle controles gebeurd zijn en er deden zich geen onregelmatigheden voor dan heeft de parameter $valid de waarde true. Hier zit nog een laatste controle in het systeem. Een gebruiker mag geen 2 rijen met dezelfde datum en uur kunnen ingeven. Daarom wordt er, indien de gebruiker een rij wil toevoegen, nog eens gecontroleerd of de rij al bestaat. Er wordt alvorens nog wel eerst gecontroleerd of een rij gewijzigd of nieuw is. Dit gebeurt door middel van de functie *is_numeric* van de parameters in de URL.

```
if ( $valid )
{
    if( is_numeric( $facility ) )
    {
        $tpl->setVariable( 'id', $entry->attribute( 'id' ) );
        $entry->store();
```

```
        $url = 'ems_data/selectfacility/(facility)/' . $entry-
>attribute( 'facility' );
        $Module->redirectTo( $url );
    }
    else
    {
        $params = array( 'Facility' => $facility, 'StartDate' =>
$date, 'EndDate' => $date, 'StartTime' => $time, 'EndTime' => $time
);
        $record =& Logbook::fetchData($params);

        if ( count( $record ) != 0 )
        {
            $tpl->setVariable ( 'recordexists', 'true' );
        }
        else
        {
            $entry->store();
            $url = 'ems_data/selectfacility/(facility)/' .
$facility;
            $Module->redirectTo( $url );
        }
    }
}
```

Indien de gebruiker op de knop ‘Maak alle velden leeg’ klikt, wordt de standaard actie reset van de tag ‘form’ uitgevoerd. Bij het klikken op de knop ‘terug’ gebeurt er een redirect naar de interface *selectfacility*. Dit kunt u zien in het onderstaande script.

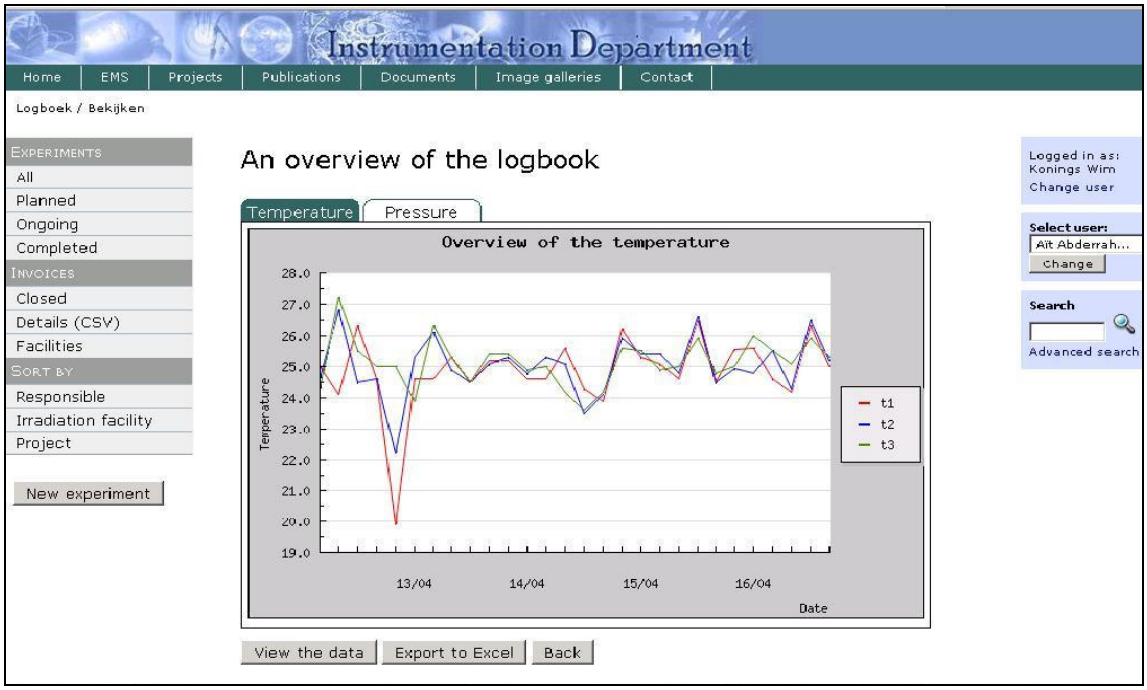
```
switch( $button )
{
    …
    case 'back':
    {
        if( is_numeric( $facility ) )
        {
            $params = array( 'Id' => $facility );

            $data =& Logbook::fetchData( $params );
            $entry = $data[0];

            $facility = $entry->attribute( 'facility' );
        }
        $url = 'ems_data/selectfacility/(facility)/' . $facility;
        $Module->redirectTo( $url );
    }
    break;
}
```

Nu de wetenschappers van de afdeling BR2 hun waarnemingen kunnen invullen, kon er aan de interface voor de dienst Instrumentatie begonnen worden. Zij hadden een interface nodig waar ze de waarnemingen konden bezichtigen.

9.1.3 View viewlogbook

De volgende interface die behandeld wordt, is aangemaakt voor de dienst Instrumentatie. Via de portaalsite van EMS kunnen de wetenschappers nieuwe experimenten registreren. Vervolgens maakt EMS een grafiek zodat de wetenschappers op een eenvoudige manier de periodes kunnen aflezen. Wanneer de wetenschappers op een irradiation period klikken, krijgen ze een overzicht van de waarnemingen in grafiekvorm.

De interface in de browser ziet er als volgt uit:



Figuur 9.4 De interface van de view viewlogbook

Om het voor de gebruiker wat aangenamer te maken om de waarnemingen te lezen werden er tabbladen aangemaakt. De tabbladen kunnen door middel van een script verwisseld worden. De techniek van het verwisselen van tabbladen komt eigenlijk neer op het verbergen en het tonen van code die zich in een div-tag bevindt. Het onderstaande script maakt dit allemaal duidelijk.

```
<div id="tabmenu">
    <div id="tab1"  class="outer"><a href="javascript:void 0;"
onclick="show('page1','tab1');hide('page2', 'tab2');hide('page3',
'tab3');"><span>Temperature</span></a></div>
    <div id="tab2" class="outer"><a href="javascript:void 0;"
onclick="show('page2','tab2');hide('page1', 'tab1');hide('page3',
'tab3');"><span>Pressure</span></a></div>
    {section show=$showflowrate}
        <div id="tab3" class="outer"><a href="javascript:void 0;"
onclick="show('page3','tab3');hide('page1', 'tab1');hide('page2',
'tab2');"><span>Flow rate</span></a></div>
    {/section}
```

```
</div>

<div id="page1" class="tabpage">
{let graph=line_plot( $graph_data_temperature )}
     <div class="scrollable"><img src="{$graph.file}" alt="graph"
/></div>
{/let}
</div>

<div id="page2" class="tabpage">
{let graph=line_plot( $graph_data_pressure )}
    <div class="scrollable"><img src="{$graph.file}" alt="graph"
/></div>
{/let}
</div>

{section show=$showflowrate}
    <div id="page3" class="tabpage">
    {let graph=line_plot( $graph_data_flowrate )}
        <div class="scrollable"><img src="{$graph.file}" alt="graph"
/></div>
    {/let}
    </div>
{/section}
```

De bovenstaande template code toont hoe de tabbladen worden geplaatst. Er zal slechts 1 tabblad zichtbaar gemaakt worden.
In de eerste section worden de namen van de tabbladen getoond. Indien er geen gegevens over de flow rate beschikbaar zijn, moet dit tabblad niet getoond worden. Vervolgens worden de tabbladen gevuld met de grafiek. Dit gebeurt door middel van de template operator *line_plot* die in één van de volgende hoofdstukken aan bod komt. U hebt waarschijnlijk al wel gemerkt dat de div-tag tot een klasse behoort. Deze klasse wordt in een CSS gedefinieerd zodat de lay-out van een tabblad tevoorschijn komt.

Deze template bevat nog 3 knoppen. De eerste knop toont de gebruiker al de gegevens in tabelvorm. De tweede knop staat de gebruiker een export naar excel toe. Met de derde knop gaat de gebruiker terug naar het gekozen experiment.

Figuur 9.5 De knoppen van de view viewlogbook

```
{let facility=$logbook[0].facility startdate=$logbook[0].date
enddate=$logbook[$logbook|count|dec].date url=concat(
'ems_data/viewlogbook/', $periodIdentifier, '/', $facility, '/',
$startdate, '/', $enddate )}
    <form method="post" action={$url|ezurl}>
        <div class="viewdata">
            <div class="button_div"><input type="submit" value="View
the data" name="btnView" /></div>
            <div class="button_div"><input type="hidden"
value="test" name="test" /><input type="submit" value="Export to
Excel" name="btnExport" /></div>
            <div class="button_div"><input type="submit"
value="Back" name="btnBackExp" /></div>
            <div class="clear_div"></div>
        </div>
    </form>
{/let}
```

Wanneer de gebruiker op de knop ‘View the data’ klikt, dan krijgt hij een interface met de waarden van de grafiek in tabelvorm te zien. Bij het klikken op de knop ‘Back’ wordt er terug gegaan naar het experiment. U kunt de werking zien in het onderstaande script.

```
switch( $Module->currentAction() )
{
    case 'back':
    {
        $position = strpos( $periodIdentifier, 'P' );
        $experiment_id = substr( $periodIdentifier, 0, $position );
        $experiment_object =& eZContentObject::fetch( $experiment_id
);
        $node =& $experiment_object->attribute('main_node');
        $url = $node->attribute( 'url_alias' );

        $Module->redirectTo( '/' . $url );
    }
    break;
    …
}
```

Uit de URL wordt de parameter $periodIdentifier gehaald. Deze parameter bestaat uit de *object_id* van een experiment, gevolgd door de letter ‘P’ en vervolgens een volgnummer.
Nu wordt eerst de *object_id* van het experiment uit de periodIdentifier gehaald.
Vervolgens wordt uit de database het contentObject gehaald van deze *object_id*. Dit gebeurt door middel van de functie *fetch* van de klasse *eZContentObject*. Om nu de juiste url_alias te verkrijgen van het gekozen experiment, wordt eerst het attribuut ‘main_node’ uit het contentObject gehaald. Deze main_node is een object van de klasse *contentObjectTreeNode* waaruit vervolgens de url_alias kan worden gehaald.

Bij het klikken op de knop ‘Export to Excel’ wordt de onderstaande code uitgevoerd.

```
$params = array( 'StartDate' => $startdate, 'EndDate' => $enddate,
'Facility' => $facility );
$data =& Logbook::fetchData( $params );

switch($Module->currentAction() )
{
    …
    case 'export':
    {
        $filename = 'logbook_' . $facility . '_' . $startdate . '_'
. $enddate . '.csv';

        $tpl->setVariable( 'logbooks', $data );

        $excelFile =& $tpl->fetch(
'design:ems_data/viewlogbook_excel.tpl' );
        $contentType = 'text/csv';

        eZHttpTool::headerVariable( 'Pragma',  'public' );
        eZHttpTool::headerVariable( 'Expires', '0' );
        eZHttpTool::headerVariable( 'Cache-Control', 'private' );

        eZHttpTool::headerVariable( 'Content-type', $contentType );
        eZHttpTool::headerVariable( 'Content-Disposition',
'attachment; filename=' . $filename );

        print( $excelFile );

        eZExecution::cleanExit();
    }
    break;
}
```

Deze code gaat eerst de bestandsnaam van het te maken bestand aanmaken. Vervolgens wordt al de data uit de database naar de template gestuurd om de lay-out van de data te bepalen.

Het resultaat van de template wordt opgeslagen in een variabele die vervolgens de inhoud van het Excel bestand wordt.
Voordat het bestand getoond kan worden, moeten er eerst enkele aanpassingen gebeuren aan de header van het bestand. Zo kunt u zien dat de headervariabele *Pragma* op public wordt geplaatst. De variabele *Cache-Control* werd op private geplaatst wat wil zeggen dat het document in de browser moet gecached worden.
Voorts kan er nog ingesteld worden wanneer het bestand vervalt dankzij de Expires parameter. Er wordt ook bepaald van welk type het bestand moet zijn, namelijk text/csv. Tot slot wordt de variabele *Content-Disposition* op attachement geplaatst waardoor er een ‘Opslaan als’ dialoogbox tevoorschijn kan komen.
Op deze manier kan het bestand door meerdere browsers ondersteund worden.
Normaal wordt de template getoond maar de laatste regel van het script gaat dit tegenhouden. Deze regel zorgt ervoor dat eZ publish geen verdere output van gegevens moet aanmaken, en enkel de dialoogbox toont.

De onderstaande code is van de template ‘viewlogbook_excel’. De output hiervan wordt doorgegeven aan de variabele $excelFile waarop het nieuwe bestand is gebaseerd. In de template wordt gebruik gemaakt van de template operator *customround*. Via deze operator kan een getal worden afgerond tot op de opgegeven waarde. Daarnaast wordt er ook gebruik gemaakt van de template operator *csvescape*. Met deze operator worden alle quotes van de waarde van het attribuut ‘freetext’ vervangen door dubbele quotes. Als een waarde een quote bevat en er werd geen escape gedaan, dan kon Excel deze waarde niet volledig tonen.
Indien u een voorbeeld wilt bekijken van de output van de template ‘viewlogbook_excel’, verwijs ik u door naar bijlage 3.

```
Logbook

Facility:;{$logbooks[0][facility]}
Start date:;{$logbooks[0][date]}
End date:;{$logbooks[$logbooks|count|dec][date]}


Facility;Date;Time;T1;T2;T3;P1;P2;P3;F1;F2;Remarks

{section var=logbook loop=$logbooks}
{$logbook.facility};{$logbook.date};{$logbook.time|extract_left(5)};
{customround( $logbook.t1, 2)};{customround( $logbook.t2,
2)};{customround( $logbook.t3, 2)};{customround( $logbook.p1,
2)};{customround( $logbook.p2, 2)};{customround( $logbook.p3,
2)};{section show=ne( $logbook.f1, 0 )}{customround( $logbook.f1,
2)};{customround( $logbook.f2, 2)};{section-
else};;{/section}{csvescape( $logbook.freetext|wash )}

{/section}
```

Hieronder vindt u het script dat zorgt voor de verwerking van de rest van de template. Er wordt een variabele geïnitialiseerd met de gegevens voor de facility. Deze variabele wordt vervolgens naar de template operator *line_plot* gestuurd. In hoofdstuk 11 vindt u hiervan de code.

```
$params = array( 'StartDate' => $startdate, 'EndDate' => $startdate,
'Facility' => $facility );

$data =& Logbook::fetchData( $params );

if( count( $data ) != 0 )
{
    $tpl->setVariable( 'exists', true );
    $tpl->setVariable( 'logbook', $data );

    $color = array( 'red', 'blue', 'chartreuse4' );

    $limit = 6;
    $offset = count( $data ) % $limit;
    $requesteddata =& $data;
    foreach( $requesteddata as $result )
    {
        $t1data[]=$result->attribute( 't1' );
        $t2data[]=$result->attribute( 't2' );
```

```
        $t3data[]=$result->attribute( 't3' );
        $p1data[]=$result->attribute( 'p1' );
        $p2data[]=$result->attribute( 'p2' );
        $p3data[]=$result->attribute( 'p3' );
        $f1data[]=$result->attribute( 'f1' );
        $f2data[]=$result->attribute( 'f2' );
        $datedata[]=$result->attribute( 'date' );
        $timedata[]=$result->attribute( 'time' );
    }

    $t1 = array( 'name' => 't1', 'data' => $t1data, 'color' =>
$color[0] );
    $t2 = array( 'name' => 't2', 'data' => $t2data, 'color' =>
$color[1] );
    $t3 = array( 'name' => 't3', 'data' => $t3data, 'color' =>
$color[2] );

    $p1 = array( 'name' => 'p1', 'data' => $p1data, 'color' =>
$color[0] );
    $p2 = array( 'name' => 'p2', 'data' => $p2data, 'color' =>
$color[1] );
    $p3 = array( 'name' => 'p3', 'data' => $p3data, 'color' =>
$color[2] );

    $f1 = array( 'name' => 'f1', 'data' => $f1data, 'color' =>
$color[0] );
    $f2 = array( 'name' => 'f2', 'data' => $f2data, 'color' =>
$color[1] );

    $date = array( 'name' => 'date', 'data' => $formatdate );

    $graphdatatemp = array( 'title' => 'Overview of the
temperature', 'x' => 'Date', 'y' => 'Temperature', 'data_x' =>
array( $date ), 'data_y' => array( $t1, $t2, $t3 ), 'offset'
=>$offset, 'limit' => $limit );

     $tpl->setVariable( 'graph_data_temperature', $graphdatatemp );
}
```

Eerst en vooral wordt er gecontroleerd of er wel gegevens voor de facility bestaan tijdens de 2 periodes. Dit gebeurt door middel van de functie *fetchData* van de klasse *Logbook*.
Wanneer er zich gegevens in de database bevinden, worden er enkele eigenschappen van de grafiek bepaald. Zo kunt u zien dat de grafiek voor T1 rood is, voor T2 blauw en voor T3 chartreuse4.
Vervolgens worden de variabele *limit* en *offset* gedeclareerd. De variabele *limit* bepaalt om de hoeveel ticks op de X-as er een benaming van de waarde moet komen. In dit geval zal er om de 6 ticks een datum staan.
De variabele *offset* bepaalt de beginpositie van een benaming van de waarde. In dit geval wordt de offset berekend aan de hand van de rest van de gehele deling tussen het totaal aantal waarden en de limit.
Daarna worden de gegevens in een array geplaatst.

In het onderstaande stuk code vindt u de output van 3 rijen die gegenereerd worden door de foreach loop.

```
object(logbook)(14) { ["PersistentDataDirty"]=> bool(false)
["Facility"]=> string(8) "brigitte" ["Date"]=> &string(10) "2005-04-
12" ["Time"]=> &string(8) "06:00:00" ["T1"]=> &string(7) "25.0000"
["T2"]=> &string(7) "24.6000" ["T3"]=> &string(7) "24.3000" ["P1"]=>
&string(7) "50.0000" ["P2"]=> &string(7) "52.0000" ["P3"]=>
&string(7) "51.0000" ["F1"]=> &string(6) "0.0000" ["F2"]=>
&string(6) "0.0000" ["Freetext"]=> string(0) "" ["ID"]=> string(3)
"100" }

object(logbook)(14) { ["PersistentDataDirty"]=> bool(false)
["Facility"]=> string(8) "brigitte" ["Date"]=> &string(10) "2005-04-
12" ["Time"]=> &string(8) "10:00:00" ["T1"]=> &string(7) "24.1000"
["T2"]=> &string(7) "26.8000" ["T3"]=> &string(7) "27.2000" ["P1"]=>
&string(7) "53.6000" ["P2"]=> &string(7) "47.6000" ["P3"]=>
&string(7) "52.4000" ["F1"]=> &string(6) "0.0000" ["F2"]=>
&string(6) "0.0000" ["Freetext"]=> string(0) "" ["ID"]=> string(3)
"102" }

object(logbook)(14) { ["PersistentDataDirty"]=> bool(false)
["Facility"]=> string(8) "brigitte" ["Date"]=> &string(10) "2005-04-
12" ["Time"]=> &string(8) "14:00:00" ["T1"]=> &string(7) "26.3000"
["T2"]=> &string(7) "24.5000" ["T3"]=> &string(7) "25.5000" ["P1"]=>
&string(7) "55.3000" ["P2"]=> &string(7) "51.3000" ["P3"]=>
&string(7) "54.6000" ["F1"]=> &string(6) "0.0000" ["F2"]=>
&string(6) "0.0000" ["Freetext"]=> string(0) "" ["ID"]=> string(3)
"103" }
```

Tot slot worden alle parameters en waarden in 1 parameter ($graphdatatemp) geplaatst die vervolgens naar de template wordt gestuurd. Het is belangrijk om de waarden met de keys in een array te plaatsen. Zo kan de template operator al de gegevens correct uit de array halen en omzetten naar een grafiek.
De array moet de volgende keys bevatten:

- title: bevat de titel van de grafiek
- x: bevat de titel van de X-as
- y: bevat de titel van de Y-as
- data_x: bevat de gegevens van de X-as
- data_y: bevat de gegevens van de Y-as
- offset: bevat de startpositie van de benaming van de waarden
- limit: bevat het aantal ticks per benaming

Toen deze interface klaar was, werd me gevraagd om nog een laatste interface te maken voor het logboek. In deze interface *viewdata* worden al de waarnemingen die op de grafiek te zien zijn in een tabel gegoten. De wetenschappers verkrijgen zo een gedetailleerd overzicht van al de waarnemingen voor die periode. Er werd ook een handige filter aangemaakt om de gegevens per dag te kunnen bezichtigen. Na enkele uren was ook deze interface af en was het logboek klaar.

Vanaf nu was het tijd voor het beheren van een dosimetrie.

9.2 Dosimetrie

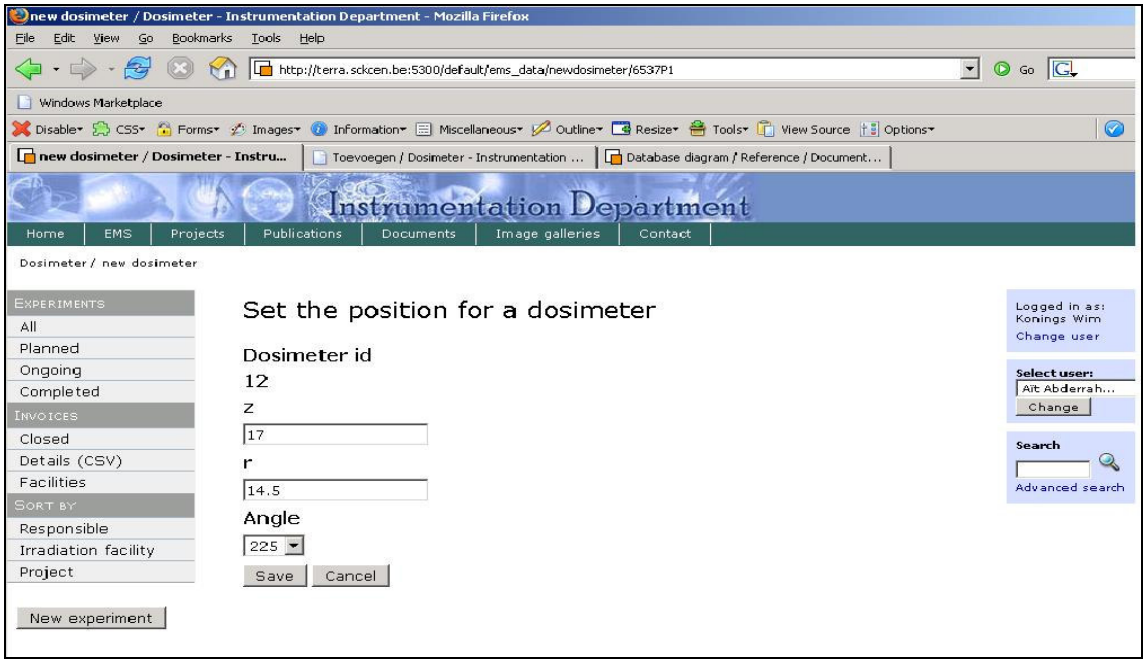
Om een degelijke werking van het dosimetriebeheer te garanderen, gebeurt dit proces in 2 fases. In de eerste fase moet de dienst Instrumentatie bepalen tijdens welke dosimetrieperioden er dosimeters geplaatst moeten worden. Voor elke dosimeter wordt een plaats binnen de irradiation facility bepaald door de wetenschappers.
Aan de hand van deze gegevens kan de afdeling BR2 aan de slag gaan. In de 2de fase gaan de wetenschappers van de afdeling BR2 de dosimeters plaatsen op de vooropgestelde coördinaten. Na het uitvoeren van het experiment worden de waarnemingen ingevuld per dosimeter. Vanaf dit moment krijgen de wetenschappers van de dienst Instrumentatie per dosimetrieperiode een overzicht van de resultaten.

9.2.1 View newdosimeter

Ik ben eerst begonnen met het maken van de interface *placedosimeter*. De wetenschappers van de dienst Instrumentatie bereiken deze interface nadat ze in EMS in de grafiek op een dosimetrie hebben geklikt. De interface biedt de wetenschappers een overzicht van dosimeters die er tijdens de dosimetrie aanwezig zijn. Voor elke dosimeter worden ook zijn waarnemingen getoond indien die al ingevuld zijn. Na het klikken op de knop 'add new dosimeter' komt de wetenschapper in de interface *newdosimeter* terecht.

In deze interface gaan de wetenschappers bepalen op welke positie een dosimeter geplaatst moet worden. De module genereert een nieuwe id voor een dosimeter die de wetenschappers vervolgens moeten opschrijven. Met deze id kan de afdeling BR2 de resultaten van de dosimeter invullen.

De interface in de browser ziet er als volgt uit:



Figuur 9.6 De interface van de view newdosimeter

In deze interface moet de gebruiker enkel de posities van de dosimeter ingeven en vervolgens op ‘Save’ klikken. Ik zal u laten zien hoe de verwerking van de interface tot stand komt.

```
<?php
    include_once( 'kernel/common/template.php' );
    ext_class('ems_data', 'dosimeter' );

    $tpl =& templateInit();
    $urlparams = $Params['Parameters'];
    $periodIdentifier = $urlparams[0];

    $valid=true;
    $tpl->setVariable( 'zvalid', true);
    $tpl->setVariable( 'rvalid', true );
    $tpl->setVariable( 'noPeriodIdentifier', false );

    $Module =& $Params['Module'];
    $button = $Module->currentAction();

    if( ! isset( $periodIdentifier ) && $periodIdentifier == '' )
    {
        $tpl->setVariable( 'noPeriodIdentifier', true );
    }
```

In het script worden eerst enkele parameters ingelezen. U kunt in het bovenstaande script zien dat de variabele $periodIdentifier uit de URL wordt gehaald. De periodIdentifier is de eerste key in de array van de parameters. U krijgt nu de output van de variabele $urlparams te zien. Dit principe om variabelen uit een URL te halen komt nog enkele malen aan bod.

```
array(1) {
  [0]=>
  string(6) "6536P4"
}
```

Verder wordt de variabele $button geladen met de waarde van de knop waarop geklikt is. Naast het inlezen van enkele parameters worden er ook nog enkele parameters geïnitialiseerd om de foutcontrole van de template te bevorderen. Op die manier kan er een gepaste foutboodschap getoond worden bij de verkeerd ingevulde parameter. Na het initialiseren van de variabelen wordt de onderstaande code verder uitgevoerd.

```
$newID = DosimeterFunctionCollection::getNewID();

$dosimeter = new Dosimeter();
$dosimeter->setAttribute( 'id', $newID );
$dosimeter->setAttribute( 'r', 0 );
$dosimeter->setAttribute( 'angle', 0 );
$dosimeter->setAttribute( 'periodidentifier', $periodIdentifier );
```

Na het laden van de variabelen wordt er een object van de klasse *Dosimeter* geïnitialiseerd. In dit object worden attributen geladen met standaard waarden. Na overleg met de wetenschappers van de dienst Instrumentatie werd er geconcludeerd dat het attribuut ‘angle’ en ‘r’ standaard op 0 mogen worden geplaatst.
Daarnaast wordt ook het attribuut *id* opgevuld met de nieuwe id van een dosimeter.

Wanneer de gebruiker klikt op de knop ‘Save’ worden alle parameters uit de template gehaald zodat de verwerking kan starten.

```
switch( $button )
{
    case 'save':
    {
        $z = $Module->actionParameter( 'z' );
        $z = str_replace( ',', '.', $z );
        $r = $Module->actionParameter( 'r' );
        $r = str_replace( ',', '.', $r );
        $angle = $Module->actionParameter( 'angle' );

        if( ! is_numeric( $z ) )
        {
            $valid=false;
            $tpl->setVariable ( 'zvalid', false);
        }

        if( ! is_numeric( $r ) )
        {
            $valid=false;
            $tpl->setVariable ( 'rvalid', false);
        }

        if( $valid )
        {
            $dosimeter->setAttribute( 'z', $z );
            $dosimeter->setAttribute( 'r', $r );
            $dosimeter->setAttribute( 'angle', $angle );
            $dosimeter->store();
            $url = 'ems_data/placedosimeter/' . $periodIdentifier;
            $Module->redirectTo( $url );
        }
    }
    break;
    …
```

Het eerste wat u ziet gebeuren in de tak ‘save’ is het inladen van de variabelen $z en $r met de waarden van de parameters. Vervolgens ziet u dat de functie *str_replace* een bewerking gaat doen met de variabelen. In de analyse van de dosimetrie stond beschreven dat de wetenschappers zowel een komma als een punt mogen schrijven voor een decimale waarde. Maar MySQL ondersteunt geen komma’s in decimale waarden. Vandaar dat elke komma vervangen wordt door een punt.

De enige controles die er in deze view moeten gebeuren is het controleren op numerieke waarden van de variabele $z en $r. Als dit het geval is, is alles geldig en kan de verdere afhandeling van het bewaren gebeuren.
Om het object in de database op te slaan wordt de functie *store* van de klasse *eZPersistentObject* gebruikt. Nadien zal er een redirect gebeuren naar de overzichtspagina van de dosimeters. Dit gebeurt eveneens wanneer de gebruiker op de knop ‘Cancel’ klikt.

Ter volledigheid laat ik u ook de code achter de knop ‘Cancel’ zien.

```
switch( $button )
{
    …
    case 'cancel':
    {
        $url = 'ems_data/placedosimeter/' . $periodIdentifier;
        $Module->redirectTo( $url );
    }
    break;
}
```

Dit waren al de interfaces die er nodig zijn om een dosimetrie te laten starten. Maar voor de verdere verwerking van het experiment had de dienst Instrumentatie nog 1 interface nodig, namelijk *createfunction*. Ik heb u al eerder verteld dat de dosis van een dosimeter wordt berekend aan de hand van de vergelijking van een calibratiecurve. Het zijn de wetenschappers van de dienst Instrumentatie die bepalen welk type dosimeter er geplaatst moet worden. Zij weten wat de vergelijking is van de calibratiecurve. Om te bepalen hoeveel dosis een dosimeter heeft opgevangen, moeten de wetenschappers dus een interface krijgen waarin ze de vergelijking kunnen noteren.

9.2.2 View createfunction

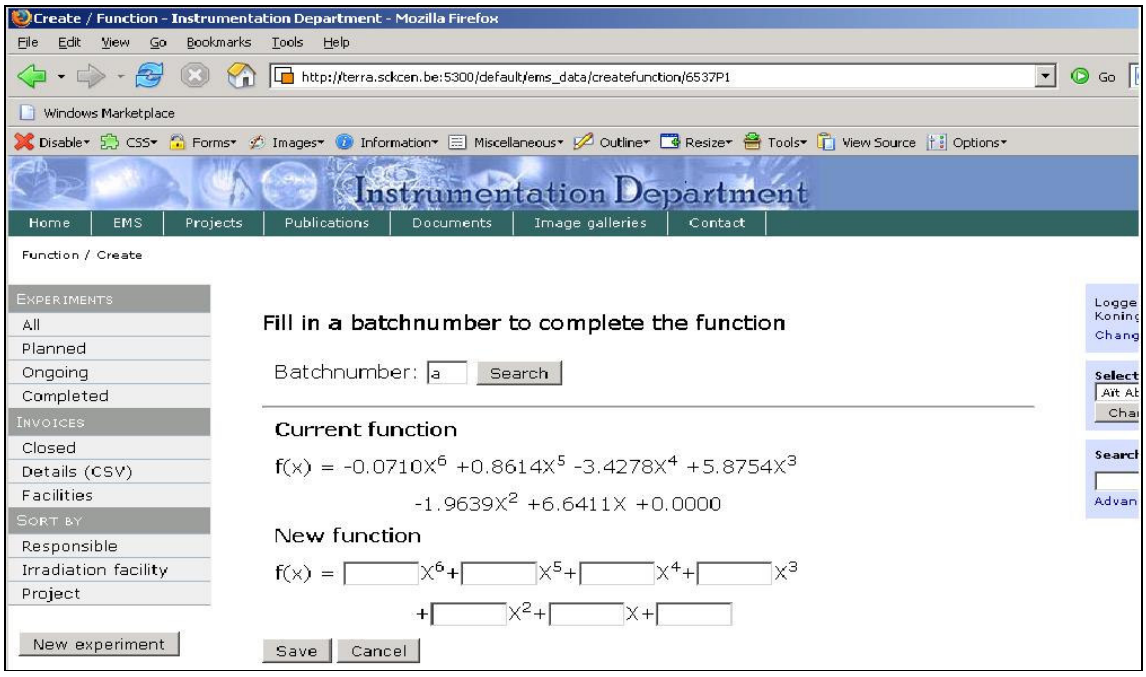
Het uiteindelijke doel van de dienst Instrumentation voor een dosimetrie is het te weten komen hoeveel dosis er door een dosimeter is geabsorbeerd. Na een experiment wordt er eerst berekend hoeveel licht er geabsorbeerd is om vervolgens de overschakeling naar dosis te maken. De berekening van de hoeveelheid dosis gebeurt door middel van een calibratiecurve waarvan de vergelijking gekend is. Eerst werd er gedacht om de vergelijking in een bestand op de server te beheren. Maar nadat bleek dat het type vergelijking altijd hetzelfde bleef, werd besloten om de variabelen van de vergelijking in een tabel op te slaan. Via een interface kunnen de wetenschappers de waarden van de vergelijking makkelijk aanpassen.

Om goed te begrijpen hoe deze interface werkt, is het belangrijk te weten dat de vergelijking gebaseerd is op het materiaal waaruit een dosimeter is gemaakt. Elke dosimeter heeft een batchnummer die verwijst naar een plaat waaruit de dosimeters zijn gemaakt. Een batchnummer gaat van A tot Z, in totaal zijn er dus 26 vergelijkingen.
Na verloop van tijd zijn alle batchnummers opgebruikt waardoor er een bestaand nummer wordt herbruikt. Wanneer bijvoorbeeld batchnummer A overschreven wordt, moet ook de vergelijking voor dit batchnummer veranderen. Elke plaat waaruit dosimeters gemaakt worden, heeft zijn gevoeligheid en dus ook zijn vergelijking. In de bijlage1 en 2 kunt u een voorbeeld zien van een calibratiecurve met zijn vergelijking.

Net omdat een batchnummer na verloop van tijd van vergelijking kan veranderen, wordt het berekende attribuut dosis ook opgeslagen in de database.

In de interface is er eerst een knop voorzien om te zoeken naar een batchnummer. Vervolgens kan de gebruiker nieuwe waarden invullen voor een nog niet bestaande vergelijking. Of hij kan een bestaande vergelijking overschrijven met de nieuwe waarden.

De interface in de browser ziet er als volgt uit:



Figuur 9.7 De interface van de view createfunction

Eerst wil ik uw aandacht vestigen op de verwerking van de knop ‘Search’.

Net zoals in alle scripts wordt er ook in dit script enkele klassen geïmplementeerd, o.a. de parameters uit de URL gehaald en enkele parameters voor de verwerking van de template geïnitialiseerd. Omdat deze code meestal dezelfde is en omdat ik ze al regelmatig besproken heb, ga ik ze niet meer bespreken. Zo kan ik de meest relevante stukken van de verwerking van het PHP-script aan u tonen.

```
$Module =& $Params['Module'];
$button = $Module->currentAction();

$batch = $Module->actionParameter( 'batch' ) ;
$tpl->setVariable( 'batch', $batch );

$entry = new FunctionParameters();

$params['Batchnumber'] = strtoupper( $batch );
$data =& FunctionParameters::fetchData( $params );

if( count( $data ) > 0 )
{
    $tpl->setVariable( 'data', $data[0] );
}

switch( $button )
{
    case 'search':
    {
```

```
        if( ! isset( $batch ) || $batch == '' || is_numeric( $batch
) )
        {
            $tpl->setVariable( 'batchvalid', 'false' );
        }
        else
        {
            $tpl->setVariable( 'batchvalid', 'true' );
        }
    }
    break;
    …
}
```

In het bovenstaande script gebeurt de verwerking van het zoeken naar het batchnummer.

Voordat de template de gezochte batchnummer kan tonen, gebeuren er nog enkele stappen in de verwerking.
Eerst wordt de waarde van de parameter *batch* opgehaald uit de template.
Indien het batchnummer een alfanumerieke waarde heeft, wordt er in de database gezocht naar een rij met deze batchnummer. Dit gebeurt door middel van de functie *fetchData* van de klasse *FunctionParameter*. Vervolgens worden de resultaten van de fetch doorgesluisd naar de template.

Indien de gebruiker op de knop 'Search' geklikt heeft, wordt er een controle op het batchnummer uitgevoerd. De variabele $batch moet bestaan en mag geen numerieke waarde zijn. Indien dit wel het geval is, wordt de variabele $batchvalid geïnitialiseerd met de waarde *false* en vervolgens naar de template gestuurd. Zo kan er een gepaste foutboodschap getoond worden.

Figuur 9.8 De foutboodschap van createfunction

Nadat de gebruiker naar een batchnummer heeft gezocht, kan hij nieuwe waarden invullen voor de vergelijking. Indien de gebruiker een bestaande vergelijking wil wijzigen is de code van de knop 'Save' als volgt.

```
<input type="submit" value="Save" name="btnSave" onclick="return
save(false)" />
```

De knop zal eerst een javascript triggeren om een bevestiging te vragen voor het overschrijven van de functie. De functie zal de gebruiker de vraag stellen 'Are you sure you want to overwrite the existing file?' via een dialoogbox. Afhankelijk van de keuze van de gebruiker zal er een submit van het formulier gebeuren of niet.

Wanneer de gebruiker een functie wil opslaan, wordt er net zoals bij het opzoeken van een batchnummer eerst gecontroleerd of het batchnummer alfanumeriek is. Vervolgens wordt elke parameter afzonderlijk gecontroleerd aan de hand van een zelfgemaakte functie *controlParameter*. In de onderstaande code vindt u het inlezen van de waarde van parameter *p7*. Vervolgens gebeurt de controle van de variabele.

```
$p7 = $Module->actionParameter( 'p7' );
controlParameter( 'p7', &$p7, &$tpl, &$entry, &$valid );
```

De functie *controlParameter* gaat eerst alle komma's van de waarde $p7 vervangen in punten. Een getal met een komma in kan niet worden opgeslagen in de database, maar er moest wel ondersteuning zijn indien de gebruiker een komma invulde.
Vervolgens wordt gecontroleerd of de waarde numeriek is. Afhankelijk van het resultaat van de controle worden de variabele $p7valid en $valid aangepast en doorgegeven aan de template. U vindt de variabele $p7valid niet terug in deze code maar hij werd bovenaan het script geïnitialiseerd met de waarde true.

U merkt dat er voor de variabelen een ampersand staat. Dit wil zeggen dat de variabele by reference wordt doorgegeven aan de functie. Op die manier wordt de nieuwe variabele slechts een verwijzing naar de originele variabele.

```
if( $valid )
{
    if( count( $data ) == 0 )
    {
        $entry->setAttribute( 'batchnumber', strtoupper( $batch ) );
    }
    else
    {
        $entry->setAttribute( 'id', $data[0]->attribute('id') );
        $entry->setAttribute( 'batchnumber', strtoupper( $batch ) );
    }
    $entry->store();
    $url = 'ems_data/placedosimeter/' . $periodIdentifier;
    $Module->redirectTo( $url );
}
```

Nadat alle parameters correct werden ingevuld komt PHP in de bovenstaande code terecht.
In het begin van het script werd de rij uit de database gehaald van het gezochte batchnummer. Om te weten of het om een nieuwe of een bestaande vergelijking gaat, wordt het aantal rijen uit de database geteld. Als de waarde 0 is, hebben we te maken met een nieuwe vergelijking anders met een bestaande vergelijking. Enkel het batchnummer moet ingevuld worden indien het over een nieuwe vergelijking gaat. Indien een bestaande vergelijking wordt overschreven moet de id van de rij ook worden ingevuld in het object.

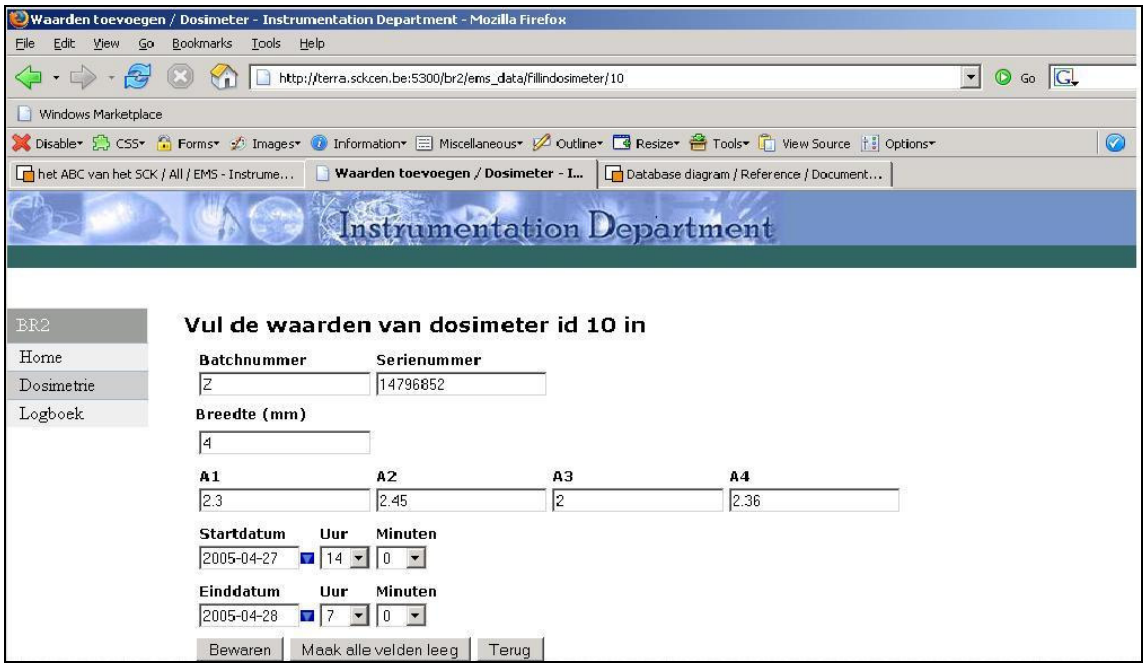
Na het opslaan van de gegevens gebeurt er een redirect naar de view *placedosimeter*.

Nu zijn al de interfaces over de dosimetrie voor de dienst Instrumentatie af. Er dienen enkel nog 2 interfaces aangemaakt te worden zodat de afdeling BR2 de waarnemingen kan opslaan. De eerste interface die ik toen heb aangemaakt was selectdosimeter. Ik heb al vermeld dat de wetenschappers van de dienst Instrumentatie bij een nieuwe dosimeter de id opschreven zodat de afdeling BR2 deze nummer kan gebruiken. Het is dit nummer dat ingevuld moet worden door de wetenschappers om een waarneming van een dosimeter op te slaan in de interface *selectdosimeter*. Nadat ze een id hebben ingevuld, gaan de wetenschappers verder naar de laatste interface namelijk *fillindosimeter*.

9.2.3 View fillindosimeter

In de laatste interface kunnen de wetenschappers van de afdeling BR2 hun waarnemingen voor een dosimeter invullen.

De interface in de browser ziet er als volgt uit:



Figuur 9.9 De interface van de view fillindosimeter

In deze interface kan de wetenschapper zijn waarnemingen over een dosimeter ingeven en vervolgens opslaan door op ‘Bewaren klikken. Ik zal u laten zien hoe de verwerking van de interface tot stand komt.

Net zoals in al de andere scripts worden er ook nu weer enkele klassen geïmplementeerd. Daarnaast wordt ook het templateobject geïnitialiseerd om o.a. variabelen naar de template te sturen. Zo kunt u zien dat er nog andere variabelen worden geïnitialiseerd om de verwerking van de template te bevorderen.

```
<?php
    include_once( 'kernel/common/template.php' );
    ext_class( 'ems_data', 'dosimeter' );
    ext_class( 'ems_data', 'functionparameters' );
    include_once(
'extension/ems_data/modules/ems_data/controlfields.php' );
    include_once( 'lib/ezfile/classes/ezfile.php' );

    $tpl =& templateInit();

    $tpl->setVariable ( 'batchvalid', true );
    $tpl->setVariable ( 'serievalid', true );
    $tpl->setVariable ( 'widthvalid', true );
```

```
    $tpl->setVariable ( 'a1valid', true);
    $tpl->setVariable ( 'a2valid', true );
    $tpl->setVariable ( 'a3valid', true);
    $tpl->setVariable ( 'a4valid', true );
    $tpl->setVariable ( 'startperiodvalid', true);
    $tpl->setVariable ( 'endperiodvalid', true );
    $tpl->setVariable ( 'checkdatevalid', true );
    $tpl->setVariable( 'recordexists', true );
    $tpl->setVariable( 'makefile', false );
```

Nadat deze variabelen geïnitialiseerd zijn, kan de verdere verwerking van de template gebeuren. Eerst moet er worden gecontroleerd of de id van de dosimeter correct wordt doorgegeven, of hij numeriek is en, of er een rij in de database bestaat voor die id.

```
if( isset( $Params['Parameters'][0] ) && ! is_null(
$Params['Parameters'][0] ) )
{
    $id = $Params['Parameters'][0];
}

if( is_numeric( $id ) && ! is_null( $id ) )
{
    $params = array( 'Id' => $id );
    $data =& Dosimeter::fetchData( $params );
}
else
{
    $data = array( 'result' => array() );
}

if( count( $data ) != 0)
{
    $entry = $data[0];
    $tpl->setVariable( 'logbookentry', $data[0] );
}
else
{
     $tpl->setVariable( 'recordexists', false );
}
```

In de eerste if-structuur wordt er gecontroleerd of er een parameter in de URL zit. Indien dit waar is, wordt de variabele $id opgevuld met de id van de dosimeter. Vervolgens wordt er gecontroleerd of de id numeriek is. Indien deze bewering waar is, wordt er gezocht in de database naar een rij met deze id. Tot slot wordt er gecontroleerd of de rij bestaat en dan wordt dit object doorgestuurd naar de template.

Vervolgens wordt de onderstaande code uitgevoerd nadat de gebruiker op de knop ‘Bewaren’ heeft geklikt. In deze code wordt er enkel aandacht besteed aan de verwerking van de startperiode.

```
switch( $button)
{
    case 'save':
    {
    $startdate = $Module->actionParameter( 'startdate' );
    $starthour = $Module->actionParameter( 'starthour' );
    $startminute = $Module->actionParameter( 'startminute' );

    if( $startdate == '' || ! is_numeric( $starthour ) || !
is_numeric( $startminute ) )
    {
        $tpl->setVariable( 'startperiodvalid', false );
        $valid = false;
    }
    else
    {
        $startperiod = mktime( $starthour, $startminute, 0, substr(
$startdate, 5, 2 ), substr( $startdate, 8, 2 ), substr( $startdate,0
, 4 ) );
        $entry->setAttribute( 'startperiod', $startperiod );
    }
```

Nadat de variabelen $startdate, $starthour en $startminute werden ingeladen, wordt er gecontroleerd of deze waarden niet numeriek zijn. Indien dit het geval is, is de startperiode niet correct. Er zal vervolgens een aangepaste foutboodschap getoond worden.

Figuur 9.10 De foutboodschap van fillindosimeter

Indien de variabelen wel correct zijn, wordt er voor de startperiode een unix timestamp gemaakt van de variabelen. Deze timestamp wordt vervolgens in het object $entry geplaatst.

Nadat alle parameters correct werden ingevuld en in het object $entry werden geplaatst, wordt de dosis berekend van de dosimeter.
De dosis wordt berekend aan de hand van de vergelijking van een calibratiecurve. Hier moet de waarde van de absorptie ingevuld worden om de overeenkomstige dosis te bekomen.
In het onderliggende script wordt eerst de gemiddelde absorptiewaarde berekend. Omdat de vergelijking van de curve telkens van hetzelfde type is, kon de vergelijking hard gecodeerd worden in PHP. In bijlage 1 en 2 vindt u een voorbeeld van een vergelijking van een calibratiecurve.
Voordat de dosis berekend wordt, wordt er eerst nog gecontroleerd of er al een vergelijking bestaat voor het batchnummer. Nadien worden enkel de bijkomstige parameters geladen voor de vergelijking om de dosis te berekenen.
De vergelijking is van het volgende type: $aX^6 + bX^5 + cX^4 + dX^3 + eX^2 + fX + g$.

```
if( $valid )
{
    $average = ( $a1 + $a2 + $a3 + $a4 ) / 4;

    $params['Batchnumber'] = $batch;
    $data =& FunctionParameters::fetchData( $params );

     if( count( $data ) == 0 )
    {
        $tpl->setVariable( 'makefile', true );
    }
    else
    {
        $p1 = $data[0]->attribute( 'p1' );
        $p2 = $data[0]->attribute( 'p2' );
        $p3 = $data[0]->attribute( 'p3' );
        $p4 = $data[0]->attribute( 'p4' );
        $p5 = $data[0]->attribute( 'p5' );
        $p6 = $data[0]->attribute( 'p6' );
        $p7 = $data[0]->attribute( 'p7' );
        $dose = ($p1 * pow( $average, 6 ) + $p2 * pow( $average, 5 )
+ $p3 * pow( $average, 4 ) + $p4 * pow( $average, 3 ) + $p5 * pow(
$average, 2 ) + $p6 * pow( $average, 1 ) + $p7);

        $entry->setAttribute( 'dose', $dose );
        $entry->store();

        $url = 'ems_data/selectdosimeter';
        $Module->redirectTo( $url );
    }
}
```

Wanneer de gebruiker op de knop ‘Terug’ klikt, wordt onderstaande code uitgevoerd. Er zal enkel een redirect gedaan worden naar de view *selectdosimeter*.

```
switch( $button )
{
    …
    case 'back':
    {
        $url = 'ems_data/selectdosimeter';
        $Module->redirectTo( $url );
        break;
    }
}
```

Nu deze view klaar was, waren ook de functionaliteiten van het dosimetriebeheer klaar. Nadat de gebruikers van de afdeling BR2 hun waarnemingen hebben ingevuld, kunnen de mensen van de dienst Instrumentatie gebruik maken van het overzicht in de interface *placedosimeter*. Via dit overzicht kunnen zij de waarnemingen van een dosimeter beoordelen.

In de volgende hoofdstukken kunt de uitleg vinden over het aangepaste datatype ‘eZSchedule’ en de template operators ‘drawExperimentSchedule’ en ‘line_plot’.

10 Datatype

Binnen EMS was er een datatype om de periodes van experimenten te beheren. Het datatype *eZSchedule* was echter niet compleet voor mijn stageopdracht. Ik heb een extra attribuut toegevoegd aan het datatype om een periode van een experiment uniek te maken.

10.1 eZSchedule

Het datatype *eZSchedule* werd aangemaakt om de ganse planning van een experiment op te slaan. Het heeft de mogelijkheid om de start- en einddatum van een experiment bij te houden maar ook om een type experiment met de bijbehorende start- en eindperiodes op te slaan. In functie van de stageopdracht moesten de periodes van een dosimetrie uniek en gekend zijn zodat er dosimeters aan een bepaalde dosimetrie gekoppeld konden worden.

De klasse van het datatype eZScheduleType maakt gebruik van 3 hulpklassen. In deze klassen heb ik slechts enkele kleine aanpassingen moeten doen zodat het unieke nummer toegevoegd wordt aan het datatype.

Om de werking van het datatype te begrijpen zal ik eerst uitleggen hoe de onderlinge relaties tussen de klassen zijn.
In eerste instantie is er de klasse van het datatype, namelijk *eZScheduleType*. Hierin bevinden zich allerlei functies om bijvoorbeeld de input te valideren, op te vangen of op te slaan. In de functie *fetchObjectAttributeHTTPInput* wordt de input opgevangen. Deze functie maakt gebruik van de eerste hulpklasse namelijk *eZSchedule*.
De functie *addPeriod* wordt opgeroepen om een nieuw object van de klasse *Period* aan het scheduleobject toe te voegen. Hier komt de tweede hulpklasse aan bod, namelijk *SCKPeriod*. Elk type bestraling heeft zijn periode en dit wordt bijgehouden aan de hand van een object van de klasse *SCKPeriod*.
Maar logischerwijze kan een periode niet zonder een datum en tijd. De derde hulpklasse *SCKDateTime* zal de datum en tijd van een periode bijhouden.
Samenvattend is er een datetimeobject dat een attribuut is van de klasse *SCKPeriod*. Verder is er een period_object dat een object is van de klasse *SCKPeriod* en een attribuut van de klasse *eZSchedule*. Tot slot is er een scheduleobject dat een object is van de klasse *eZSchedule* en dat wordt toegevoegd aan het contentObjectAttribute. Dit object zal uiteindelijk worden opgeslagen in de database.

Vervolgens worden mijn specifieke aanpassingen uitgelegd.

In de klasse van het datatype heb ik enkel in de functie *fetchObjectAttributeHTTPInput* een nieuwe parameter moeten toevoegen. Deze functie wordt opgeroepen als het datatype de variabelen van de webpagina wil opvangen. De nieuwe parameter, *periodIdentifier*, bestaat uit de *object_id* van het aangemaakte experiment gevolgd met een P en een volgnummer. De webpagina kreeg al een contentattribute object mee waaruit ik de *object_id* van het experiment kon uithalen. Voorts bevatte de webpagina ook al een id van een periode zodat ik in deze klasse weinig werk had.

```
$experiment_object_id = $contentObjectAttribute->attribute(
'contentobject_id' );

foreach( $periodIDArray as $key => $periodID)
{
     $schedule->addPeriod( new SckPeriod( $experiment_object_id,
$periodIDArray[$key], $periodStartDate, $periodEndDate,
$periodTypeArray[$key], $periodDescriptionArray[$key] ) );
}
```

In het bovenstaande script worden, van de parameters $experiment_object_id en $periodIDArray, een object van de klasse *SckPeriod* gemaakt. Dit object wordt vervolgens aan een functie van de hulpklasse *eZSchedule* meegegeven. Op deze manier wordt het object in een variabele gestopt van de klasse *eZSchedule*. In een latere fase wordt de variabele in de database gestockeerd.

U vraagt zich misschien af hoe deze grote hoeveelheid aan informatie in een database wordt opgeslagen. Het is niet nodig om evenveel veldjes aan te maken zodat alle gegevens apart in de database terecht komt.
eZ publish biedt u de mogelijkheid om complexe zaken, zoals onder andere dit datatype, als XML output in de database te stockeren.
Binnen de klasse *eZSchedule* bestaat er een functie *XMLString* die van een object een XML gaat maken. In deze functie moest ik enkele nieuwe regels code toevoegen om de periodIdentifier als een attribuut van een periode op te slaan.
In de onderstaande code vindt u de aanpassingen aan de functie *XMLString*.

```
$periodNode->set_attribute( 'identifier', $period->attribute(
'identifier' ) );
```

Om de werking in omgekeerde richting in goede banen te leiden moest ik in de functie *decodeXML* ook enkele aanpassingen doen. Deze functie zorgt er voor dat een XML-bestand kan worden omgevormd tot een object.

```
foreach( $periodNodes as $periodNode )
{
   $periodIdentifier = $periodNode->attributeValue( 'identifier' );
   $position = strpos( $periodIdentifier, 'P' );

   $experiment_object_id = substr( $periodIdentifier,0,$position );
   $period_object_id = substr( $periodIdentifier, $position+1,
count( $periodIdentifier ) );

   $periodStartNode =& $periodNode->elementByName( 'start' );
   $periodEndNode =& $periodNode->elementByName( 'end' );

   $period = new SckPeriod( $experiment_object_id,
$period_object_id, SckDateTime::decodeDOMNode( $periodStartNode ),
SckDateTime::decodeDOMNode( $periodEndNode ), $periodNode-
>elementTextContentByName( 'type' ), $periodNode-
>elementTextContentByName( 'description' ) );

   $this->addPeriod( $period );
}
```

In het bovenstaande script worden al de periodes overlopen die zich in het XML-formaat bevinden. Het attribuut *periodIdentifier* wordt daarna uit de XML gehaald. Vervolgens wordt er nagegaan op welke positie van de periodIdentifier de letter ‘P’ zit. Al de waarden voor deze plaats vormen de *experiment_object_id*. Al de waarden na deze plaats vormen de *period_object_id*. Tot slot worden alle gegevens in een object van de klasse *Period* geplaatst die vervolgens in een object van de klasse *Schedule* komt.

Nu alles in gereedheid is om de waarden op te vangen moest ik er alleen nog voor zorgen dat de period_id bij elke nieuwe periode verhoogd werd. Dit gebeurt aan de hand van de onderstaande template code.

```
{section show=ne( $Period.identifier, '' )}
    {let periodID=-1}
    {section var=id loop=$Period.identifier|count_chars}
        {section show=eq( $Period.identifier|extract( $id.index, 1
), 'P' )}
            {set periodID=$Period.identifier|extract( $id.index|inc,
$Period.identifier|count_chars )}
         {/section}
     {/section}
         <input type="checkbox"
name="{$attribute_base}_schedule_period_remove_{$attribute.id}[]"
value="{$Period.index}" />
         <input type="hidden"
name="{$attribute_base}_schedule_period_id_{$attribute.id}[]"
value="{$periodID}" />
     {/let}
{section-else}
    {let maximum=-1}
    {section var=id loop=$periods}
        {section show=ne( $id.identifier, '' )}
            {let periodID=-1}
                {section var=per loop=$id.identifier|count_chars}
                    {section show=eq( $id.identifier|extract(
$per.index, 1 ), 'P' )}
                        {set periodID=$id.identifier|extract(
$per.index|inc, $id.identifier|count_chars )}
                    {/section}
                {/section}
            {section show=$periodID|gt( $maximum ) }
                {set maximum=$periodID}
            {/section}
            {/let}
        {/section}
    {/section}
        <input type="checkbox"
name="{$attribute_base}_schedule_period_remove_{$attribute.id}[]"
value="{$Period.index}" />
        <input type="hidden"
name="{$attribute_base}_schedule_period_id_{$attribute.id}[]"
value="{$maximum|inc}" />
    {/let}
{/section}
```

In deze code wordt er eerst gekeken of er sprake is van een lege periodIdentifier. Indien dit niet het geval is, wordt de identifier uit de periodIdentifier gehaald. Vervolgens wordt deze variabele in een hidden tekstvak geplaatst.
Indien de periodIdentifier wel leeg is, moet de identifier de hoogste bestaande waarde krijgen. Indien er nog geen periodIdentifier bestaat, wordt de identifier automatisch op 0 geplaatst. In het andere geval worden al de bestaande identifiers gecontroleerd op de hoogste waarde.
Bij de hoogste waarde wordt nog 1 bijgeteld zodat er een nieuwe identifier ontstaat. Tot slot wordt de nieuwe hoogste waarde in een hidden tekstveld geplaatst.

Op deze manier wordt er telkens een nieuwe waarde gegenereerd voor een periodIdentifier. Voorheen was het zo dat wanneer een periode verwijderd werd, de identifiers automatisch opschoven. Op die manier zouden het aantal dosimeters van een bepaalde periode niet meer kloppen. Dit is nu echter wel gegarandeerd.

In de bijlage 4 vindt u een tekstbestand met de gegevens van het datatype *eZSchedule*. U kunt nagaan hoe eZ publish de gegevens opslaat als XML formaat.

11 Template operator

Zoals ik al vermeld heb genereert EMS een grafiek om periodes van experimenten te beheren. Binnen mijn stageopdracht moest men van een irradiation period verschillende waarnemingen kunnen opslaan. Het is voor de gebruiker handig om op de balk van een irradiation period in de grafiek te klikken en vervolgens al de waarnemingen te zien. Om dit te bekomen heb ik de bestaande template operator 'drawExperimentSchedule' aangepast.

Verder heb ik een tweede template operator aangemaakt om een grafiek van allerlei gegevens te genereren. Met behulp van de grafiek kunnen wetenschappers op een snelle en eenvoudige manier onregelmatigheden waarnemen.

De grafieken werden gegenereerd met behulp van de klassebibliotheek *jpGraph*. Deze klasse heeft de mogelijkheid om zowel lijngrafieken, staafgrafieken, barcodegrafieken, … aan te maken. Door deze klasse te importeren in het script kon ik van al de functies gebruik maken.

11.1 drawExperimentSchedule

Voor mijn aanpassingen werd de grafiek gegenereerd aan de hand van een standaard functie *CreateSimple*. Maar op de balken die de grafiek vervolgens genereerde konden geen clickable maps geplaatst worden. Er zat dus niets anders op dan de data uiteen te halen en er afzonderlijke balken van te maken. Op deze manier was er wel de mogelijkheid om clickable maps op de balken te plaatsen zodat de gebruiker hier op kon klikken.

Het onderstaande script toont aan wat er moest veranderen.

```
$index = $data[0][0];
$label = $data[0][2];
$start = $data[0][3];
$end = $data[0][4];
$facility = $data[0][8];

$graphbar = new GanttBar( $index, $label, $start, $end );
$graphbar->SetPattern( GANTT_SOLID, 'black' );
$graphbar->title->SetFont(FF_FONT1,FS_BOLD);
$graphbar->SetLabelLeftMargin(20);
$graph->Add( $graphbar );
```

In de bovenstaande code wordt eerst al de overkoepelende informatie van een experiment verkregen. Zo kunt u zien dat o.a. de naam van het experiment, de start- en eindtijd en de facility uit de array van parameters wordt gehaald. Daarna wordt er van al de gegevens een eerste balk gemaakt voor op de grafiek. Deze balk omvat de totale periode van het experiment omvat.

Verder moeten de afzonderlijke periodes toegevoegd worden aan de grafiek. Wanneer het een periode van een irradiation of dosimetry was, moest er een image map over de balk geplaatst worden.

```
foreach( $data as $value )
{
    $index = $value[0];
    $label = $value[2];
    $start = $value[3];
    $end = $value[4];
    $periodIdentifier = $value[6];

    $graphbar = new GanttBar( $index, $label, $start, $end );
    $graphbar->SetPattern( BAND_RDIAG, 'yellow' );
    $graphbar->SetFillColor( 'red' );

    if( trim( strtolower( $label ) ) == 'irradiations' && $start <=
date( 'Y-m-d' ) )
    {
        $url = 'http://' . eZSys::hostname() .
'/default/ems_data/viewlogbook/';
        $graphbar->SetCSIMTarget( $url . $facility . '/' . $start .
'/' . $end );
        $graphbar->SetCSIMAlt( 'View the logbook for this period' );
        $graphbar->title->SetCSIMTarget( $url . $facility . '/' .
$start . '/' . $end  );
        $graphbar->title->SetCSIMAlt( 'View the logbook for this
period' );
    }
    if( trim( strtolower( $label ) ) == 'dosimetries' && $start <=
date( 'Y-m-d' ) )
    {
        $url = 'http://' . eZSys::hostname() .
'/default/ems_data/placedosimeter/';
        $graphbar->SetCSIMTarget( $url . $periodIdentifier );
        $graphbar->SetCSIMAlt( 'Set the position of a dosimeter for
this period' );
        $graphbar->title->SetCSIMTarget( $url );
        $graphbar->title->SetCSIMAlt( 'Set the position of a
dosimeter for this period' );
    }
    $graph->Add( $graphbar );
    unset( $graphbar );
}
```

Het resultaat van de bovenstaande code is een rode balk per periode die wordt toegevoegd aan de grafiek. Wanneer het type bestraling een irradiation of dosimetrie is, wordt er met de functie *SetCSIMTarget* een image map aan de balk toegevoegd. Als parameter wordt de URL meegegeven waarnaar de gebruiker gaat als hij op de balk klikt.

Op de volgende pagina vindt u een voorbeeld van output van de waarde waar overlopen wordt. Zo kunt u de start- en eindperiode terug vinden in de array net zoals het type en de periodIdentifier.

```
array(7) {
  [0]=>
  int(1)
  [1]=>
  int(0)
  [2]=>
  string(17) "  Irradiations   "
  [3]=>
  string(10) "2005-04-12"
  [4]=>
  string(10) "2005-04-16"
  [5]=>
  string(0) ""
  [6]=>
  string(6) "6537P0"
}
…
```

Er wordt in het script ook gebruik gemaakt van de klasse *eZSys*. Deze klasse maakt het mogelijk om de URL van de website te achterhalen. Zo wordt de URL dynamisch gegenereerd.

Tot slot moesten er nog enkele aanpassingen gebeuren aan de tags die gegenereerd worden met de grafiek.

```
$graph->Stroke( $this->CacheDir . eZSys::fileSeparator() . $this-
>FileName );

$areas =& $graph->GetCSIMareas();

$filename = 'jpgraph' . $this->FileName;
$areas = str_replace( '>', ' />', $areas );

$imagemap = "<map name=\"$filename\" id=\"$filename\">\n";
$imagemap .= $areas;
$imagemap .= "</map>";

return array
(
    'imageMap' => $imagemap,
    'areas' => $areas,
    'fileURL' => 'http://' . eZSys::hostname() . '/' .
eZSys::cacheDirectory() . '/' . $this->FileName,
    'fileName' => $filename
);
```

In de eerste regel van het bovenstaande script ziet u de functie *Stroke* die de grafiek zal tekenen. Het tekenen van de grafiek is eigenlijk het maken van een figuur van de grafiek. Daarom worden er parameters zoals de naam van het bestand en de URL doorgegeven aan de template zodat de template eigenlijk alleen maar een figuur laadt.

Verder worden er nog de area-tags aangepast en krijgt de tag ‘map’ nog een attribuut ‘id’ bij. Deze aanpassingen zijn noodzakelijk om de webpagina te laten voldoen aan de standaard XHTML Strict. Dit is een strengere en ‘schonere’ versie van HTML. XHTML maakt gebruikt van een DTD om aan te duiden welke tags en welke attributen toegestaan zijn. Wanneer een pagina gebruik maakt van XHTML zullen webbrowsers vaker eenzelfde resultaat laten zien.

Tot slot wordt er een return van een array gedaan met daarin de waarden van allerlei variabelen. Door een return te doen van deze waarden zullen de keys van de array gebruikt kunnen worden binnen de template.

Nu u weet hoe de template operator is aangemaakt, moet u enkel nog weten hoe u hem moet gebruiken. Dit kunt u zien in de volgende code.

```
{let graphParam=drawExperimentSchedule($node) }
    <img src="{$graphParam.fileURL}" alt="schedule"
usemap="#{$graphParam.fileName}"/>
    {$graphParam.imageMap}
{/let}
```

Zoals reeds vermeld wordt de grafiek opgeslagen als een figuur. Wanneer vervolgens de template operator aangeroepen wordt, en opgeslagen in een variabele, kan deze variabele gebruikt worden om naar het pad van de figuur te verwijzen. Door de variabele in de source van een figuur te plaatsen, wordt de figuur getoond.

11.2 line_plot

Naast het aanpassen van een bestaande template operator moest ik ook een template operator creëren om een lijngrafiek te genereren.
In hoofdstuk 9 werd de view *viewlogbook* besproken waar er verschillende parameters werden geprepareerd om ze vervolgens aan de template operator mee te geven. Deze parameters worden hier terug uit de array gehaald om ze te gebruiken voor het maken van de grafiek.

```
case 'line_plot':
{
include_once('extension/ezpowerlib/jpgraph/src/jpgraph_line.php');

$graphdata = $namedParameters['graph_data'];

$ticks = count($graphdata['data_x'][0]['data']);
$div = 7 * $graphdata['limit'];
$weeks = ( $ticks - $ticks % $div ) / $div ;
$width = ( $weeks + 1 ) * 570;

$graph = new Graph($width, 350, 'auto');
```

Er wordt in de bovenstaande code een nieuwe grafiek gecreëerd met een breedte die dynamisch is. De grafiek krijgt een breedte zodat gegevens van een volledige week zichtbaar zijn. Wanneer gegevens van een langere periode getoond worden, zal er automatisch een scrollbar tevoorschijn komen.

De scrollbar is te danken aan de onderstaande code van de stylesheet.

```
div.scrollable {
    width: 100%;
    overflow: auto;
    overflow-x: scroll;
    font-size: 0;
}
```

De eigenschap *overflow* wordt op auto geplaatst waardoor inhoud die groter is dan de div-tag een scrollbar verkrijgt. Onder de eigenschap *overflow* vindt u de eigenschap *overflow-x* die de waarde scroll heeft. Normaal was enkel de eigenschap overflow nodig om scrollbars te verkrijgen maar dit is niet zo in Internet Explorer. Daarom wordt de eigenschap *overflow-x* ook nog toegevoegd aan de stylesheet zodat ook Internet Explorer een scrollbar toont.

Verder worden de eigenschappen van de grafiek aangepast zodat de gebruiker een duidelijk beeld heeft van de gegevens.

```
$graph->title->set( $graphdata['title'] );

$graph->img->SetMargin(60,80,40,60);
$graph->img->SetAntiAliasing();

$graph->xaxis->SetTickLabels($date);
$graph->xaxis->SetLabelMargin(20);
$graph->xaxis->SetTitleMargin(30);
$graph->xaxis->title->Set( $graphdata['x'] );

$graph->yaxis->title->Set( $graphdata['y'] );
$graph->yaxis->SetTitleMargin(40);

$graph->legend->Pos(0.0035,0.5, 'right', 'center');

$graph->SetTickDensity(TICKD_NORMAL);
```

Tot slot worden de gegevens uit de parameter gehaald. De gegevens worden vervolgens toegevoegd aan de lijngrafiek.
Het is belangrijk om de gegevens de juiste benaming te geven. Wanneer de gegevens op een andere wijze worden opgeslagen als in hoofdstuk 8 vermeld is, dan wordt er geen grafiek gegenereerd.
Op deze manier ziet u goed de samenhang tussen de templates en de scripts.

```
foreach($graphdata['data_y'] as $row)
{
    $lineplot = new Lineplot( $row['data'] );
    $lineplot->SetColor( $row['color'] );
    $lineplot->SetLegend( $row['name'] );
    $graph->Add( $lineplot );
    unset( $lineplot );
 }
```

In de onderstaande code krijgt u een overzicht van de gegevens die zich in de foreach-structuur bevinden in de variabele $row.

```
array(3) {
  ["name"]=>
  string(2) "t1"
  ["data"]=>
  array(28) {
    [0]=>
    string(7) "25.0000"
    [1]=>
    string(7) "24.1000"
    [2]=>
    string(7) "26.3000"
    …
}
  ["color"]=>
  string(3) "red"
  …
}
```

Net zoals bij de vorige template operator moeten er enkele waarden doorgegeven worden aan de template. Omdat de grafiek als een figuur wordt opgeslagen zijn de parameters ‘filename’ en ‘file’ belangrijk om de grafiek te tonen op het scherm.

```
$graph->Stroke($this->CacheDir . $this->FileName);

$ret = array();
$ret['file'] = "http://". eZSys::hostname() . '/'
.eZSys::cacheDirectory()."/jpgraph/".$this->FileName;
$ret['filename'] = $this->FileName;
$ret['imagemap'] = $graph->GetHTMLImageMap($this->FileName);

$operatorValue = $ret;
```

Om deze template operator te kunnen gebruiken, gaat u op dezelfde manier tewerk als de template operator *drawExperimentSchedule*. U moet eerst een variabele declareren met de template operator en de gegevens. Vervolgens moet u ook nu het pad tot de figuur in de source van de img-tag plaatsen om de figuur te kunnen waarnemen.

```
{let graph=line_plot( $graph_data_temperature )}
  <div class="scrollable"><img src="{$graph.file}" alt="graph" /></div>
{/let}
```

Besluit

Het systeem voor logboek-en dosimetriebeheer was op tijd klaar. Het voldeed aan al de ‘voorlopige’ functionele en niet-functionele eisen van de wetenschappers. Maar er is een verschil tussen de manier waarop de wetenschappers het systeem wensen en de manier hoe het systeem is. Het is dus nog af te wachten wat het oordeel van de wetenschappers is over de huidige verwezenlijking.

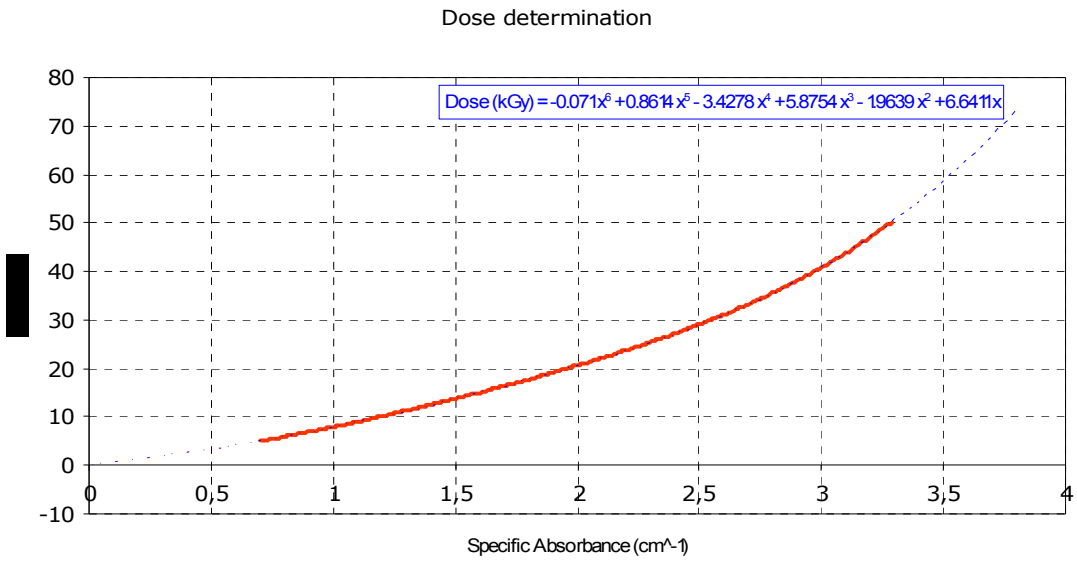
Concreet werd er de mogelijkheid voorzien voor de wetenschappers om waarnemingen in te vullen. Dit werd zowel gedaan voor het logboekbeheer als voor het dosimetriebeheer. Maar de mogelijkheden om de resultaten te interpreteren zijn talrijker voor het logboek dan voor de dosimetrie. Voor de dosimetrie kunnen de resultaten enkel in tabelvorm bezichtigd worden terwijl voor het logboek de resultaten in grafiekvorm tevoorschijn komen. In de toekomst eisen de wetenschappers misschien een eenvoudigere vorm van resultaten te vergelijken dan de huidige tabelvorm.

Momenteel bevinden mijn verrichtingen zich nog op de testserver. Nadat de wetenschappers het licht op groen hebben gezet, wordt het systeem waarschijnlijk op de portaalsite geplaatst. Het kan worden geïntegreerd in de portaalsite van het Experiment Management System.

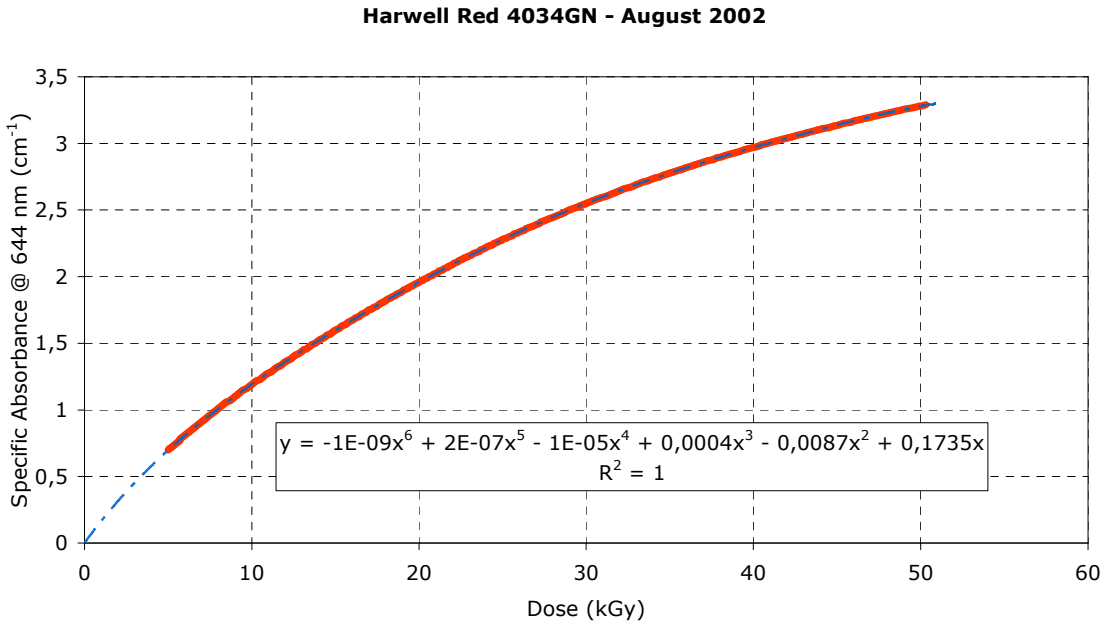
Gedurende mijn stage heb ik veel bijgeleerd over object georiënteerd programmeren en PHP. Maar er zijn nog enkele factoren die me zijn opgevallen. Bijvoorbeeld aan de laadsnelheid van een webpagina wordt op school geen aandacht aan besteed maar het is van essentieel belang in een bedrijf. Daarom vind ik deze stageperiode van 3 maanden zeer nuttig. Je leert een heleboel over zowel de schoolse als niet-schoolse zaken.

Bijlagen

Bijlage 1: de calibratiecurve van absorptie naar dosis



Bijlage 2: de calibratiecurve van dosis naar absorptie



Logbook

Facility: brigitte

Start date: 12/04/2005

End date: 14/04/2005

Facility	Date	Time	T1	T2	T3	P1	P2	P3	F1	F2	Remarks
brigitte	12/04/2005	6:00	25	24.6	24.3	50	52	51			
brigitte	12/04/2005	10:00	24.1	26.8	27.2	53.6	47.6	52.4			
brigitte	12/04/2005	14:00	26.3	24.5	25.5	55.3	51.3	54.6			
brigitte	12/04/2005	18:00	24.6	24.6	25	50.3	50.4	50.2			
brigitte	12/04/2005	22:00	19.9	22.2	25	50	52.3	48.5			
brigitte	13/04/2005	2:00	24.6	25.3	23.9	49.5	49.6	49.8			
brigitte	13/04/2005	6:00	24.6	26.1	26.3	48.9	49.7	51.2			
brigitte	13/04/2005	10:00	25.3	24.9	25.3	54	50	54			
brigitte	13/04/2005	14:00	24.5	24.5	24.5	50.5	50.5	50.5			
brigitte	13/04/2005	18:00	25.2	25.1	25.4	51.2	52	51.8			
brigitte	13/04/2005	22:00	25.2	25.3	25.4	51.2	52.4	49.9			
brigitte	14/04/2005	2:00	24.6	24.8	24.9	50.6	50.4	50.5			
brigitte	14/04/2005	6:00	24.6	25.3	25	49.8	49.9	50.5			
brigitte	14/04/2005	10:00	25.6	25.1	24.2	50.6	51.5	53.1			
brigitte	14/04/2005	14:00	24.3	23.5	23.6	50.6	51.7	49.6			
brigitte	14/04/2005	18:00	23.9	24.1	24.2	50.3	50.4	50.2			
brigitte	14/04/2005	22:00	26.2	25.9	25.6	51.5	51.4	51.4			

Bijlage 4: een voorbeeld van het opslaan van het datatype ezschedule als XML-formaat in de database

```
<?xml version="1.0" encoding="UTF-8"?>
<ezschedule>
  <periods>
    <period identifier="6537P0">
      <start>
        <year>2005</year>
        <month>04</month>
        <day>12</day>
        <hour>2</hour>
        <minute>0</minute>
        <second>0</second>
      </start>
      <end>
        <year>2005</year>
        <month>04</month>
        <day>16</day>
        <hour>18</hour>
        <minute>0</minute>
        <second>0</second>
      </end>
      <type>Irradiation</type>
      <description></description>
    </period>
    <period identifier="6537P1">
      <start>
        <year>2005</year>
        <month>04</month>
        <day>26</day>
        <hour>8</hour>
        <minute>0</minute>
        <second>0</second>
      </start>
      <end>
        <year>2005</year>
        <month>05</month>
        <day>03</day>
        <hour>4</hour>
        <minute>0</minute>
        <second>0</second>
      </end>
      <type>Dosimetry</type>
      <description></description>
    </period>
    <period identifier="6537P2">
      <start>
        <year>2005</year>
        <month>05</month>
        <day>11</day>
        <hour>18</hour>
        <minute>0</minute>
        <second>0</second>
      </start>
      <end>
        <year>2005</year>
        <month>05</month>
        <day>13</day>
```

```
      <hour>10</hour>
      <minute>0</minute>
      <second>0</second>
    </end>
    <type>Manipulation</type>
    <description></description>
  </period>
  <period identifier="6537P3">
    <start>
      <year>2005</year>
      <month>05</month>
      <day>23</day>
      <hour>6</hour>
      <minute>0</minute>
      <second>0</second>
    </start>
    <end>
      <year>2005</year>
      <month>06</month>
      <day>08</day>
      <hour>22</hour>
      <minute>0</minute>
      <second>0</second>
    </end>
    <type>Irradiation</type>
    <description></description>
  </period>
  <period identifier="6537P4">
    <start>
      <year>2005</year>
      <month>06</month>
      <day>14</day>
      <hour>3</hour>
      <minute>0</minute>
      <second>0</second>
    </start>
    <end>
      <year>2005</year>
      <month>06</month>
      <day>22</day>
      <hour>0</hour>
      <minute>0</minute>
      <second>0</second>
    </end>
    <type>Dosimetry</type>
    <description></description>
  </period>
</periods>
<start>
  <year>2005</year>
  <month>04</month>
  <day>04</day>
  <hour></hour>
  <minute></minute>
  <second></second>
</start>
<end>
  <year>2005</year>
  <month>06</month>
  <day>30</day>
  <hour></hour>
```

```
    <minute></minute>
    <second></second>
  </end>
</ezschedule>
```

Bijlage 5: de eisen van de dienst Instrumentatie i.v.m. dosimetrie

Red Perspex http://www.harwell-dosimeters.co.uk/dosimeter_faq.html

Experiment Name
Budget
Irradiation facility
Begin dosimetry
End dosimetry
Total time
Experiment status

From EMS

Planned
Ongoing
Completed
Closed

	r	Angle	z*
1	0	0	
2	0	0	
3	0	0	
4	0	0	
5	0	0	
6	0	0	

z is measured from the bottom of the irradiation facility

* Mandatory field

Add New dosimeter

Link to reference technical drawing for facility

Additional information

Interface for BR2

Begin dosimetry
End dosimetry
Total time
dosimetry

	Batch Nbr	Serial nber	Width (mm)	A1	A2	A3	A4	Average	Dose (Gy)	Dose rate (Gy/h)
1										
2										
3										
4										
5										
6										

Computed from Batch Number through calibration curve & total dosimetry time

Export to MS Excel

Literatuurlijst

GOVAERTS, Paul, 2002. 1952 – 2002. Brochure, Mol, SCK•CEN, 57 p.

GOVAERTS, Paul, 1999. Klaar voor de 21ste eeuw, Mol, SCK•CEN, 32 p.

COUWBERGHS, Tom, 2003. Ontwikkeling van een gecentraliseerd bibliografisch referentiesysteem met integratie in een portaalsite. Niet-gepubliseerd eindwerk, Geel, Katholieke Hogeschool Kempen Departement Handelswetenschappen en Bedrijfskunde, 88 p.

COOMANS, Kristof, 2004. Ontwikkeling van een centraal bibliografisch referentiesysteem met eZ publish. Niet-gepubliseerd eindwerk, Geel, Katholieke Hogeschool Kempen Departement Handelswetenschappen en Bedrijfskunde, 78 p.

DIELS, Guy, 2004. Beheer van gammabestralingsexperimenten. Niet-gepubliseerd eindwerk, Geel, Katholieke Hogeschool Kempen Departement Handelswetenschappen en Bedrijfskunde, 57 p.

BAKKEN, S., AULBACH, A., SCHMID, E., WINSTEAD, J., WILSON, L. T., ZMIEVSKI,A., AHTO, J., 2004. PHP Handleiding. http://www.php.net/manual/nl/ (4 april 2005).

OVERVIEW, 2005. Overview of the MySQL Database Management System. http://dev.mysql.com/doc/mysql/en/what-is.html/ (4 april 2005).

APACHE, 2005. Apache HTTP Server Version 2.0 Documentation. http://httpd.apache.org/docs-2.0/ (8 april 2004).

EZ, 2005. eZ publish Documentation. http://www.ez.no/ez_publish/documentation/ (8 april 2005).

ROBERTSON, James, 3 juni 2003. So, what is a content management system? http://www.steptwo.com.au/paper/kmc_what/ (10 mei 2005).

CMS, 2005. CMS Ratings. http://www.opensourcecms.com/index.php?option=content&task=view&id=388&Itemid=143 (16 mei 2005).

RUYSSEN, Marie-Laure, 6 augustus 2002. Definition of Content vs. Documents. http://portal-projects.sckcen.be/article/articleview/54/1/49/ (23 mei 2005).

FERNANDEZ FERNANDEZ, A., 2 mei 2005. SCK•CEN gamma irradiation facilities for radiation testing in high-vacuum conditions., http://www.sckcen.be/people/affernandez/pdf/BB_radecs2003_3.pdf/ (23 mei 2005).

HOW, 2005. How to use the GPL or LGPL. http://www.gnu.org/licenses/gpl-howto.html/ (23 mei 2005).

WEB, 2005. Web Servers Statistics. http://cities.lk.net/usawebstat.htm/ (23 mei 2005).

WHAT, 2005. What is NEMO? http://www.micro-optics.org/ (24 mei 2005).

COLLINS-SUSSMAN, B., FITZPATRICK, B.W., PILATO, C.M., 2005. Version Control with Subversion. http://svnbook.red-bean.com/en/1.1/svn-book.pdf/ (27 mei 2005).

FERNANDEZ FERNANDEZ, A., 2003. Irradiation facilities at SCK•CEN for radiation tolerance assessment of space materials. http://www.sckcen.be/people/affernandez/pdf/AFF_ESA03.pdf/ (1 juni 2005).