REM Automatisering

HET ONTWERPEN EN BOUWEN
BIJ REM AUTOMATISERING
VAN EEN APPLICATIE
TER ONDERSTEUNING VAN
DATABASE BEHEERTAKEN

Auteur: M. de Haij,

EINDVERSLAG 9 JANUARI 2003

REM Automatisering

Haagse Hogeschool

Naam: Martin de Haij
Studentnummer: 97002056

Bedrijf: REM Automatisering
Afdeling: Software Ontwikkeling
Begeleider: Paul de Boer

School: Haagse Hogeschool
Sector: Informatica
Opleiding: Informatievoorziening en Informatietechnologie (IVIT)
Studiepad: Ontwikkeling van Software en Technische Infrastructuur (OSTI)
Examinatoren:
Rianne Bechet-Tjoonk
Alwine Lousberg-Orbons

Onderwerp:	Het ontwerpen en bouwen bij REM automatisering van een applicatie ter ondersteuning van database beheertaken
Afstudeerperiode:	mei 2003- oktober 2003, oktober 2003 – februari 2003

Voorwoord

De student wil graag alle overige betrokkenen bij het afstuderen van harte bedanken, waaronder

- Paul de Boer, de bedrijfsmentor
- Herbert Beemstra en Annelies Snelders, de college programmeurs
- de overige collega's die zorgden voor gezellige pauzes
- de examinatoren Rianne Bechtet-Tjoonk en Alwine Lousberg-Orbons
- Adrie Schipper, die het mogelijk maakte om een deeltijdstudie in Delft te beginnen.

De student heeft een heel leuke afstudeerperiode gehad bij REM Automatisering en tussen de bedrijven door ook een van de fijnste zomervakanties van de afgelopen jaren.
Tijdens het afstuderen heeft de student nog wel enkele vreemde dingen meegemaakt. Zo is hij een avond te lang blijven werken en per abuis opgesloten. Toen hij opstond ging opeens loeihard het alarm af! Verder is er een steekvlam van 15 cm aan de achterkant uit zijn monitor gekomen. Iets meer naar rechts en hij had een plastic poster geraakt....

Inhoudsopgave

1 Inleiding

Dit is het afstudeerverslag van Martin de Haij. In dit verslag heb ik zoveel mogelijk chronologisch de werkzaamheden tijdens de afstudeerperiode verteld. De eerste hoofdstukken behandelen de inwerkperiode en zijn voor de lezer dan ook zeer aan te raden.

De hoofdstukken Business Study, Functional Model Iteration en Design and Build Iteration gaan over het tot stand komen van het ontwerp en het programma. Aan het eind van de Design and Build Iteration wordt voor het eerst het programma getoond met uitleg bij de schermen.

Het hoofdstuk implementatie is niet in verleden tijd geschreven maar richt een blik op de toekomst. Het verslag eindigt tenslotte met een evaluatie van de processen en de producten.

2 Organisatie

2.1 beschrijving

REM Automatisering is opgericht in 1985 en is gevestigd in Amsterdam. Het bedrijf heeft 15 medewerkers in dienst, waaronder 3 systeemontwikkelaars. Veel medewerkers van REM werken in deeltijd. Het belangrijkste produkt is het urenverantwoordingsysteem Big Ben. REM is een van de drie grote spelers op de Nederlandse markt voor urenverantwoordingsystemen en REM verwacht dat deze markt dit jaar flink zal groeien.

Big Ben gebruikt DbExpress('DBX') om Oracle, PostgreSql of MS-SQLServer databases op uniforme wijze te benaderen. In oktober 2003 staat de oplevering van Big Ben versie 2.4.0 gepland. Big Ben bestaat momenteel uit 4 modules: Invoer, Management, Rapporten en Beheer.

Bij REM wordt gewerkt met de programmeertaal Delphi. Voor deze taal is een zeer goede ontwikkelomgeving beschikbaar waarin het uiterst eenvoudig is om informatie over klassen of procedures op te vragen. Delphi ondersteunt objectgeoriënteerde softwareontwikkeling.

Het is de gewoonte bij het bedrijf om een zeer kort systeemontwerp traject te volgen, wat resulteert in minimale documentatie van de bestaande systemen. De programmeurs zijn echter al langere tijd in dienst en weten erg veel van Big Ben. De opgeleverde documentatie bestaat meestal uit een stroomschema (voor ingewikkelde procedures) en een gebruikershandleiding. De stroomschema's zijn zelden langer dan één pagina maar de gebruikershandleidingen daarentegen kunnen meer dan 40 pagina's bevatten en zijn zeer volledig.

REM hanteert vele code conventies wat de overdraagbaarheid van de code zeker ten goede komt. De naamgeving van de variabelen is op uniforme wijze geregeld, maakt gebruik van prefixen en lijkt een beetje op de Hongaarse notatie zoals gebruikt door Microsoft.

2.2 plaats van de afstudeerder in de organisatie

De student werkt op de afdeling Softwareontwikkeling. Elke week is er een software vergadering waar de student ook aan deelneemt. Tijdens deze vergaderingen krijgt de student van zijn collega's nuttige feedback met betrekking tot de lay-out en het beslissingsalgoritme. Bij het systeemontwerp traject is de student geheel op zichzelf aangewezen. Er is binnen het bedrijf geen ervaring met UML aanwezig.

Als de student vragen heeft over de database architectuur of het programmeren in Delphi kan hij daarvoor meestal wel bij zijn collega's terecht. Om de zoveel weken bespreekt de student zijn code met een collega en ook hieruit krijgt de student zeer nuttige feedback. Deze feedback heeft betrekking op de systeemarchitectuur en de naleving van de code conventies.

3 Plan van Aanpak

3.1 inleiding

In dit hoofdstuk is het volledige Plan van Aanpak opgenomen, gebaseerd op de definitieve opdrachtsomschrijving van 1 juli.

3.2 kader

De produktie- of testdatabase van Big Ben gebruikers krijgt na verloop van tijd een grote omvang. Klanten vinden dit bezwaarlijk en zij willen grote hoeveelheden gegevens in één keer kunnen verwijderen. Momenteel nemen zij contact op met REM en het bedrijf stuurt vervolgens een script dat tot een bepaalde datum alle gegevens verwijdert

3.3 probleemstelling

De huidige aanpak heeft als nadeel dat:

- klanten voor het verwijderen van gegevens tot een bepaalde datum elk jaar opnieuw om een script moeten vragen;
- klanten run-time geen invloed uit kunnen oefenen op de te verwijderen gegevens;
- klanten run-time geen feedback krijgen;
- REM voor elke klant scripts moet schrijven en deze moet beheren;

3.4 doelstelling

Het doel van de afstudeeropdracht is het ontwikkelen van een nieuwe Big Ben module ('Big Ben Systeem') voor applicatiebeheerders. Deze module zal waarschijnlijk slechts eenmaal per jaar gebruikt worden en maakt het mogelijk om grote hoeveelheden gegevens te verwijderen.

De systeemmodule moet de volgende functionaliteiten bevatten:

- verwijderen van gegevens tot een bepaalde datum;
- verwijderen van gegevens die niet gebruikt worden.;
- verwijderen van gegevens per tabel;

Er bestaat ook een wens tot het reorganiseren van gegevens. Deze wens zal pas in een latere update verwezenlijkt worden en valt buiten het kader van de opdracht. Bij het maken van de user-interface dient de student echter wel rekening te houden met de reorganiseeropties.

3.5 uitgangssituatie

3.5.1 aanwezige documentatie

Van de Big Ben modules waren enkele stroomschema's aanwezig die een procedure verduidelijken. Deze stroomschema's zijn een voorbeeld van de functionele programmeercultuur die nog grotendeels binnen REM Automatisering leeft (Delphi vereist geen OO-stijl). De toegevoegde waarde van de stroomschema's was klein.

3.5.2 aanwezige programmacode

De code is redelijk becommentarieerd al bevat deze ook veel onzinnig commentaar als:

```
Tabel.Open; // Open de tabel
```

In vergelijking met het vorige stagebedrijf van de student worden de code conventies bij REM Automatisering zeer trouw nageleefd en dat merk je. Van variabelen in andermans code is vaak al direct duidelijk welk type ze hebben en dit ervaart de student als een groot pluspunt. Verder heeft de student nog nooit gemerkt dat andere Big Ben modules vastliepen en dit is echt iets om trots op te zijn.

Nu de positieve punten van de aanwezige programmacode genoemd zijn zal de student beginnen met het noemen van enkele negatieve punten:

- Het grote nadeel van de code is de hechte koppeling tussen de verschillende units. Dit maakt het zeer moeilijk om een losse unit op te nemen in de Big Ben Systeemmodule. Ook wordt information hiding nauwelijks toegepast en zijn er enkele voorbeelden van ad-hoc ontwerp. Vooral de koppeling tussen de units en de datamodule is veel te hecht. Zo vereist de AboutForm unit bijvoorbeeld dat een bepaalde Tabel in de Datamodule geopend is. Dit betekent dat deze Tabel nu in de Big Ben modules altijd open staan want de gebruiker zou wel eens op Help→About kunnen klikken. Deze eisen staan niet genoteerd in de header van de units en moeten uit de code gehaald worden.

- Ook de afhandeling van Exceptions is niet altijd even netjes. De aanwezige programmacode verbergt foutmeldingen soms voor de gebruiker en probeert ondertussen gewoon verder te gaan alsof er niets aan de hand is.
- Als laatste punt van kritiek wil de student het ontcijferen van het versleutelde password noemen. Dit lukt namelijk alleen als de student de controle tegen overflow van de Delphi compiler uitschakelt. Zonder overflow werkt de desbetreffende code niet(!).

Na de enorme omvang van de Big Ben database heeft het gebrek aan herbruikbaarheid van de aanwezige programmacode het meeste bijgedragen aan de zwaarte van de opdracht.

3.5.3 aanwezige kennis

De Big Ben Beheermodule bevat tienduizenden regels programmacode en toch weten de collega programmeurs er zeer veel van. Gelukkig waren ze ook geneigd om hun kennis te delen.

Van de Enterprise edition van Delphi 7 en van DbExpress heeft men pas zeer recent kennis opgedaan. Er is veel kennis aanwezig van Delphi 6 en van de Big Ben database.

3.6 te verrichten werkzaamheden

De student zal de volgende werkzaamheden uitvoeren:

- Plan van Aanpak;
- Feasibility Study;
  - oriënteren op Big Ben, DbExpress, DSDM, programmeerconventies, Delphi 7 en aanwezige scripts;
  - onderzoek naar databases en voornamelijk het gebruik van rules en procedures;
  - inventarisatie aanwezige documenten en programmatuur;
  - onderzoek naar wensen met betrekking tot de applicatie;
  - opstellen van een cascade rapport voor 'verwijderen gegevens t/m een bepaalde datum' waarin staat welke gevolgen het verwijderen van een tabel dient te hebben (deze gevolgen kunnen per functionaliteit verschillen);
  - Feasibility Report;
  - Fast Prototype;
  - Outline Plan for Development;

- Business Study;
  - Business Area Definition;
    - globaal omschrijven belangrijke Use Cases;
    - onderzoek doen naar het gebruik van Design Patterns;
    - maken UML diagrammen(klassendiagram, Toestand diagram[indien nodig], Sequence diagrammen);
    - prioriteiten stellen;
  - System Architecture Definition;
  - Development Plan;
    - prototyping strategy;
    - testing strategy;
    - configuration management plan;
- Functional Model Iteration;
  - programmeren en direct testen van de functionaliteiten met de hoogste prioriteit;
  - Prioritised functions;
  - Functional prototyping review documents;
  - Non-functional requirements;
- Design and Build Iteration;
  - Tested System;
  - design prototypes;
  - design prototyping review documents;
- Implementation;
  - User Documentation
  - Increment Review Document

3.7 methoden en technieken

3.7.1 keuze voor gebruik van DSDM

Bij de uitvoering van de opdracht zal DSDM gehanteerd worden. DSDM is een dynamische, iteratieve versie van SDM. De student is tijdens zijn opleiding alleen vertrouwd geraakt met SDM maar voorziet dat een iteratieve methode beter aansluit op de praktijk binnen het afstudeerbedrijf.

3.7.2 technieken

Te gebruiken technieken zijn UML en eventueel Design Patterns

4 Oriëntatie op Big Ben

4.1 inleiding

Big Ben is het urenverantwoordingssysteem van REM Automatisering dat DbExpress('DBX') gebruikt om een Oracle, PostgreSql of MS-SQLServer database op uniforme wijze te benaderen. In oktober 2003 staat de oplevering van Big Ben versie 2.4.0 gepland. Er zijn verschillende productlijnen van Big Ben en sommige daarvan bestaan uit meerdere modulen. Op elke plaats waar in het verslag Big Ben staat bedoelt de student de Big Ben CS produktlijn. Big Ben CS is het belangrijkste product van REM Automatisering en bestaat uit de modules Invoer, Management, Rapporten en Beheer. De opdracht van de student was 'Het ontwerpen en bouwen bij REM automatisering van een applicatie ter ondersteuning van database beheertaken'. Het was de taak van de student om de Big Ben CS Systeemmodule te maken.

4.2 aanwezige documenten (bijv. ERDs)

Toen de student begon was er een ERD aanwezig van Big Ben 2.3. In dit ERD ontbraken echter nog 3 tabellen. Met behulp van het programma TOAD heeft de student een volledig overzicht van de aanwezige tabellen, de relaties tussen de tabellen en de constraints op de tabellen op kunnen stellen. Ondertussen is er een up-to-date ERD beschikbaar gekomen.

Van belangrijke functies in Big Ben zijn stroomschema's opgenomen die de logica van een procedure of functie aangeven. Er is echter geen klassediagram of use-case diagram aanwezig.

4.3 aanwezige programmatuur

Het aantal tabellen in Big Ben 2.4.0 bedraagt 44. Het aantal regels code in de Invoer module (een van de kleinste) is 40.000. Het programma is nog niet vastgelopen en de student is nog geen bugs tegengekomen.

Helaas staat er in de code zeer weinig nuttig commentaar. Zo is het van een aantal Units niet direct duidelijk wat hun doel is. Hergebruik van code is bijna onmogelijk, al is de code wel te gebruiken als voorbeeld. De verschillende Units zijn allemaal nauw aan elkaar gekoppeld. Het is niet mogelijk om een Unit er tussen uit te halen zonder de gehele module mee te nemen. Dit is een aandachtspuntje voor de te ontwikkelen Systeem module.

4.4 integratie tool in Big Ben

De student dient een Systeem module te schrijven die op zichzelf staat en gemaakt wordt voor systeembeheerders. De Systeem module zal de krachtigste module worden en het is voor grote organisaties waarschijnlijk onwenselijk dat alle systeembeheerders hiermee mogen werken. De Beheer module zorgt bij het opstarten ervoor dat de Invoer, Management en Rapportage modules binnen 5 minuten afgesloten worden. De Systeem module zal mogelijk ook de Beheer module af moeten sluiten. Misschien is het afsluiten van andere modules niet nodig als de Systeem module altijd 's nachts uitgevoerd zal worden.

4.5 database-architectuur

Alle modules van Big Ben CS maken gebruik van dezelfde database-architectuur. Deze bestaat uit 44 tabellen (views niet meegerekend). De Big Ben CS database-architectuur is geimplementeerd in Oracle, PostgreSQL en MS-SQLServer. De tabellen zijn in elke implementatie praktisch gelijk, maar de views kunnen per implementatie verschillen. Er wordt geen gebruik gemaakt van specifieke DBMS features zodat de code van de applicatie zoveel mogelijk ontkoppeld is van het gebruikte DBMS. Rules en Stored Procedures worden niet gebruikt. De documentatie van de Big Ben database is vrij goed en het bedrijf heeft zeer mooie tools om informatie op te vragen over de database-architectuur.

4.6 ongeschreven regels

- De tabellen bevatten nogal eens velden die niet meer of nog niet gebruikt worden. Dit is niet altijd goed beschreven. Als voorbeeld de tabel Budget Periode. In de database kunnen namen staan voor de verschillende versies van een Budget Periode. Deze namen worden echter niet meer gebruikt. Uit de documentatie was het niet duidelijk welke namen ter vervanging gebruikt dienden te worden.
- Sommige tabellen zijn in de praktijk met elkaar gekoppeld op een manier die niet uit de database te halen is. Als voorbeeld Weken-Declaraties:
  Weken is gekoppeld aan Verantwoording is gekoppeld aan Declaraties.
  Toch zijn er declaraties die niet aan een vertantwoording gekoppeld zijn maar direct aan een week! Als de week verwijderd wordt dan moeten dus niet alleen de vertantwoordingen van de week verwijderd worden maar ook de declaraties.
  Een ander voorbeeld is de strengheid van de regels. Als een tabel verwijderd wordt dan mogen van het DBMS de verwijzingen naar deze tabel in andere tabellen leeggemaakt worden(set null). In de praktijk blijkt dit vaak onwenselijk en dienen de gekoppelde records verwijderd te worden. De regels van het DBMS laten dus acties toe die eigenlijk onwenselijk zijn.
- De namen van tabellen en velden in het programma kunnen afwijken van de namen die in de database gebruikt worden.

4.7 hiërarchie van de Big Ben database

Om de tabelstructuur helder te krijgen heeft de student met behulp van werkelijk prachtige tools(TOAD voor Oracle, EMS voor PostgreSQL) per tabel gekeken wat de koppelingen waren met de overige tabellen en welke vreemde sleutels NULL mochten zijn. De student heeft op basis hiervan de tabellen in 3 niveaus verdeeld: wortel, tak en blad.

Het idee is eenvoudig: bladeren kunnen zonder probleem verwijderd worden, takken kunnen pas verwijderd worden als eerst alle gekoppelde bladeren verwijderd zijn en wortels kunnen pas verwijderd worden als eerst alle gekoppelde takken verwijderd zijn (en van deze takken moeten uiteraard eerst de gekoppelde bladeren verwijderd worden).

Helaas was dit idee een te grote versimpeling van de werkelijkheid.. Het model zou iets complexer moeten worden. Er waren namelijk takken die gekoppeld waren aan andere takken. De student heeft hierop besloten om niet meer te spreken van wortels, takken en bladeren maar om elk nivo een eigen nummer te geven. Zo krijgen de wortels nivonummer 0, en alle bladeren krijgen het hoogste nivonummer. De takken zijn opgesplitst in 4 niveaus, de nummers 1 t/m 4. De takken met nivonummer 4 zijn het eenvoudigst om te verwijderen. De bladeren hebben allen het nivonummer 5.

Een voorbeeld: de gebruiker wil records uit de tabel Weken (nivo 3) verwijderen. In de lijst op de volgende bladzijden is te zien dat er records in de tabel Verantwoording (nivo 4) zijn die een vreemde sleutel van de tabel Weken bevatten(met andere woorden er zijn Verantwoordingen gekoppeld aan Weken). Dit houdt in dat er geen records uit de tabel Weken verwijderd kunnen worden voordat de gekoppelde records uit de tabel Verantwoording verwijderd zijn. Er zijn echter records uit de tabel Declaraties die gekoppeld zijn aan Verantwoording. Voordat de Verantwoordingen verwijderd kunnen worden dienen eerst deze Declaraties verwijderd te zijn.

De volgorde van het verwijderen is nu: Declaraties-Verantwoording-Weken, tabellen met het hoogste nivonummer eerst.

legenda

vet = tabel met optionele foreign key. Mag null zijn.

Uitleg: de relatie Aanstelling Type ←→ Aanstelling is vet, dus er zijn Aanstellingen die geen Aanstelling Type bevatten (maar mogelijk wel aan andere tabellen hangen).

Tabellen met nivo 5 kunnen zonder problemen opgeschoond worden. Bij tabellen met een lager nivo zullen waarschijnlijk eerst andere tabellen opgeschoond moeten worden.

Nivo	Naam Tabel	Tabellen die eerst opgeschoond moeten worden (aantal keys)
0	Aanstelling Type	**Aanstelling**, **Code Filter**
0	Aanwezig Code	Aanwezig Detail
0	Attributen	**Activiteit Attributen (2)**
0	Budget Groep	Budget
0	Budget Periode	Budget Versie
0	Code	**Verantwoording (3)**, **Budget Items (3)**, Code Filter
0	Declaratie Soort	Budget Declaratie Soort, Declaratie Filter, Declaraties
0	Laag 1	Activiteiten Schema
0	Laag 2	**Activiteiten Schema**
0	Laag 3	**Activiteiten Schema**
0	Laag 4	**Activiteiten Schema**
0	Laag 5	**Activiteiten Schema**
0	Laag 6	**Activiteiten Schema**
0	Medewerker	Aanstelling, **Budget**, Aanwezig Detail, Autorisatie Aanstelling, Autorisatie Medewerker (2), Autorisatie Organisatie, Autorisatie Project, Rapport Template
0	Rapport Groep	Rapporten
0	Tarief Codes	**Aanstelling (2)**, **Activiteiten Schema**, **Organisatie**, **Aanstellings Activiteiten**, Tarieven
1	Activiteiten Schema	Verantwoording, Aanstellings Activiteiten, Activiteit Attributen. **Budget Detail**, **Budget Items**, **Code Filter**, Organisatie Activiteiten
1	Budget	Budget Versie, Autorisatie Project, Budget Declaratie Soort, Budget Items, **Verlof Aanvraag**, **AlaCarte**
1	Organisatie	Aanstelling, **Budget Versie**, **Organisatie**, **Autorisatie Organisatie**, **Code Filter**, **Declaratie Filter**, **Koppeling**, **Organisatie Activiteiten**
1	Rapporten	Rapport Parameters, Rapport Template
2	Aanstelling	**Budget Versie**, Weken, Aanstellings Activiteiten, Autorisatie Aanstelling, **Code Filter**, **Declaratie Filter**, Declaraties, Jaar, **Koppeling**, Verlof Aanvraag, Werkplan
3	Budget Versie	Budget Detail
3	Weken	Verantwoording

legenda

vet = tabel met optionele foreign key. Mag null zijn.

Uitleg: de relatie Aanstelling Type ←→ Aanstelling is vet, dus er zijn Aanstellingen die geen Aanstelling Type bevatten (maar mogelijk wel aan andere tabellen hangen).

Tabellen met nivo 5 kunnen zonder problemen opgeschoond worden. Bij tabellen met een lager nivo zullen waarschijnlijk eerst andere tabellen opgeschoond moeten worden

Nivo	Naam Tabel	Tabellen die eerst opgeschoond moeten worden (aantal keys)
4	Jaar	AlaCarte
4	Verantwoording	**Declaraties**
5	Activiteit Attributen	
5	AlaCarte	
5	Autorisatie Aanstelling	
5	Autorisatie Medewerker	
5	Autorisatie Organisatie	
5	Autorisatie Project	
5	Budget Detail	
5	Budget Declaratie Soort	
5	Budget Items	
5	Code Filter	
5	Declaratie Filter	
5	Declaraties	
5	Koppeling	
5	Organisatie Activiteiten	
5	Rapport Parameters	
5	Rapport Template	
5	Tarieven	
5	Verlof Aanvraag	
5	Werkplan	

5 Feasibility Study

5.1 inleiding

In dit hoofdstuk is het volledige Feasibilty Report opgenomen. Dit rapport is geschreven voordat er kleine wijzigingen in de definitieve opdrachtsomschrijving optraden. Zo worden in dit rapport ook de reorganiseer problemen genoemd terwijl deze later zijn weggelaten. Het uitvoeren van de Feasibility Study heeft ongeveer 4 weken in beslag genomen. Tijdens deze periode heeft de student zich op allerlei zaken (waaronder Big Ben) georienteerd en heeft hij een prototype gemaakt op basis waarvan de meeste requirements naar voren kwamen.

5.2 doel

De Feasibility Study heeft een aantal doelen:

1) de opdracht helder krijgen
2) onderzoeken of de opdracht uit te voeren is
3) vaststellen of DSDM de juiste methode is voor de opdracht

Verder heb ik in de Feasibility Study mijn bevindingen opgenomen die ik opgedaan heb tijdens de inwerkperiode.

5.3 de opdracht

5.3.1 probleemstelling (uit Plan van Aanpak)

Er zijn 7 problemen:

1) Verwijderen van gegevens tot een bepaalde datum
2) Verwijderen van gegevens per tabel
3) Verwijderen van gegevens die niet gebruikt worden
4) Reorganiseren van Aanstellingen en Afdelingen
5) Reorganiseren van Activiteiten Schema's en Lagen
6) Reorganiseren van Afdelingen en Organisaties
7) Reorganiseren van Budgetten en Budget Groepen

5.3.2 doelstelling (uit Plan van Aanpak)

Ontwikkelen van een nieuwe Big Ben module('Big Ben Systeem') voor systeembeheerders. Deze module is voor systeembeheerders met de hoogste rechten en zij zullen de module waarschijnlijk slechts eenmaal per jaar gebruiken. Deze module stelt hen in staat om grote hoeveelheden gegevens te verwijderen of om grote reorganisaties van gegevens door te voeren.

5.4 keuze van de projectbeheersingmethode

De student meent dat DSDM goed aansluit bij zijn afstudeerproject omdat RAD binnen het afstudeerbedrijf de gangbare ontwikkelmethode is. De 9 geboden van DSDM heb ik hieronder opgenomen met commentaar over de toepasbaarheid eronder. (Een beknopte uitleg over de principes achter DSDM is te vinden in Bijlage A)

5.4.1 de 9 DSDM geboden:

1) Active user involvement is imperative
2) DSDM teams must be empowered to make decisions
3) The focus is on frequent delivery of products
4) Fitness for business purpose is the essential criterion for acceptance of deliverables
5) Iterative and incremental development is necessary to converge on an accurate business solution
6) All changes during development are reversible
7) Requirements are baselined at a high level
8) Testing is integrated throughout the lifecycle
9) A collaborative an co-operate approach between all stakeholder is essential

5.4.2 relatie tussen de DSDM geboden en de situatie van de student:

1) Elke week heeft de student de gelegenheid om een prototype of document voor te leggen aan zijn collega's. De opdrachtgever is bijna elke dag beschikbaar om de applicatie of een document te beoordelen.
2) De student heeft weliswaar geen bevoegdheid om beslissingen te nemen maar dit wordt gecompenseerd door de directe feedback van collega's en de opdrachtgever.
3) De student wil elke week een nieuwe versie van de applicatie of van documentatie tonen aan collega's op de vergadering. De tijdspanne tussen verschillende versies is dus zeer klein.
4) De student hoeft zich niet bezig te houden met performance van de applicatie
5) De student ontwikkeld de applicatie en de documentatie in iteraties.
6) Oudere versies van de documentatie en de applicatie worden op het network opgeslagen en getaped. Door de frequente feedback het niet gebeuren dat de student veel werk overnieuw moet doen.
7) Dit is niet helemaal het geval. Zo is het nu voor de student duidelijk hoe de functionaliteit 'verwijderen van gegevens t/m een bepaalde datum' in zijn werk moet gaan maar de overige functionaliteiten zullen pas later aan bod komen. Het is voor deze functionaliteiten daarom nog niet mogelijk om een globale planning te maken.
8) De student voert elke dag slechts kleine wijzigingen in de programmatuur door en test deze wijzigingen goed. Als er bugs opduiken dan zullen deze waarschijnlijk in de nieuwe toegevoegde code zitten. Dit maakt het opsporen en elimineren van de bugs zeer eenvoudig.
9) De student heeft goede relaties met de collega's en de opdrachtgever en verwacht hier geen problemen.

5.5 oriëntatie op DbExpress, aanwezige scripts, databases

Er zijn binnen REM twee zeer informatieve artikelen over DbExpress. Verder heeft de student het script gekregen dat gebruikt werd om de database van de gemeente Doetinchem schoon te maken. De student heeft onderzoek gedaan naar het gebruik van rules en procedures om tenslotte te concluderen dat het gebruik hiervan mogelijk zou zijn.

De student was echter vergeten om de documentatie van een oude versie van SQL-Server door te nemen en deze blijkt rules en procedures niet te ondersteunen. De opdrachtgever heeft laten weten dat rules en procedures geen goede oplossing zijn.

5.6 oriëntatie op programmeerconventies REM

Binnen REM wordt gebruik gemaakt van programmeerconventies die gebruik maken van vele prefixes en die anders zijn dan de student gewend was. De enige zaken waar de student echt problemen mee heeft is het schrijven van constanten in de KamelenNotatie, het niet mogen gebruiken van de namen i en j voor tellertjes en het niet mogen gebruiken van lege haakjes bij het aanroepen van een procedure of functie. Tijdens een vergadering is hierover gediscussieerd.

5.7 oriëntatie op Delphi 7

Het belangrijkste van het inwerken in Delphi 7 is het werkend krijgen van de Delphi IDE. Omdat er veel Environment Variables verkeerd stonden heeft dit 2 dagen geduurd. Bij REM is een goed Delphi 7 boek aanwezig en de student heeft een lijstje gemaakt met de interessantste pagina's en is deze gaan lezen.

5.8 Fast Prototype

Dit is optioneel maar wel uitgevoerd. De student heeft een prototype gemaakt zodat het programma tastbaarder zou worden voor de gebruikers. Ook is het maken van schermlayouts in Delphi zeer eenvoudig en waarschijnlijk sneller dan het tekenen in Word of Visio.

De eerste versie van het prototype bestond uit enkele schermen die op papier getekend waren. Deze schermen zouden de gebruiker als een soort wizard door het programma moeten leiden. Op basis van deze schermen besloot de opdrachtgever dat het echter beter was om alle informatie in een hoofdscherm te tonen. Dit hoofdscherm is vervolgens geprogrammeerd in Delphi.

Een belangrijke keuze is wel in dit vroege stadium gemaakt, namelijk om elke functionaliteit in een eigen frame te tonen. Zo is de functionaliteit 'Verwijderen van gegevens t/m een bepaalde datum' direct gekoppeld aan een frame TFrameVerwijderenTmDatum (een klasse). Dit frame staat in het bestand FrameVerwijderenTmDatum.pas. (.pas staat achter elke Delphi source code file). Het voordeel van deze methode is dat vanuit het hoofdscherm eenvoudig meerdere frames aangeroepen kunnen worden die met enkele regels code geladen kunnen worden. Het alternatief was om forms in plaats van frames te gebruiken. Forms hebben het nadeel dat ze bij het laden als apart window in het scherm geplaatst worden, bovenop het hoofdscherm en niet erin. Omdat er binnen REM en bij de student geen ervaring bestond met frames was het maken van het prototype zowel voor de student als voor de collega's toch interessant.

Nog een belangrijke keuze is het toevoegen van een geheugenmetertje geweest. Deze toonde na elke actie van de gebruiker het totale geheugengebruik van de module voor en na de actie. Het implementeren van dit geheugenmetertje zorgde ervoor dat elk geheugenlek direct ontdekt werd.

Het ontwikkelde prototype was een eenvoudige Delphi applicatie met het verwijderen van gegevens t/m een bepaalde datum scherm inclusief buttons. De gebruiker kon nog geen acties uitvoeren.

5.9interactiviteit van het programma (voor het Verwijderen van gegevens t/m een bepaalde datum)

De gebruiker heeft de mogelijkheid om uit een van te voren samengestelde lijst met tabellen een tabel te kiezen en om een datum in te voeren. Het programma geeft nu feedback aan de gebruiker. Als de gebruiker nu kiest voor "Uitvoeren" dan worden alle gegevens uit de gekozen tabel tot de gekozen datum verwijderd. Er is geen "Abort" of "Rollback" mogelijkheid nodig. Alleen foutmeldingen worden aan de gebruiker getoond in een pop-up scherm, de overige feedback zal de gebruiker eenvoudig naast zich neer kunnen leggen.

5.10 requirements (voor het Verwijderen van gegevens t/m een bepaalde datum)

- de functionaliteiten dienen in een aparte module ontwikkeld te worden
- het programma dient te voldoen aan de code conventies
- zoveel mogelijk informatie tonen in het hoofdscherm
- gebruik van stored procedures van het DBMS is verboden
- benadering van het DBMS via DbExpress
- verwijderen van gegevens moet in de juiste volgorde gebeuren
- performance is niet belangrijk

5.11 haalbaarheid van de opdracht

De student meent dat de opdracht uitvoerbaar is. Er is binnen REM veel ervaring met Delphi en databases. De te ontwerpen applicatie is qua complexiteit niet groter dan de al bestaande Big Ben modules en deze werken ook met behulp van DbExpress(DBX). Een dergelijk programma is dus al eens eerder ontwikkeld. Performance is minder belangrijk voor dit project omdat de applicatie waarschijnlijk slechts eenmaal per jaar gebruikt zal worden op een tijdstip dat het rustig is op het netwerk.

Verder heeft de student de documentatie van de Big Ben database doorgenomen en weet nu welke gevolgen het verwijderen van een willekeurig gekozen tabel dient te hebben.

Het is nog onduidelijk of alle problemen opgelost kunnen worden binnen de gestelde tijd. De student zal zich eerst concentreren op 'het verwijderen van gegevens t/m bepaalde datum'. Hierna zal de student zich gaan richten op 'het verwijderen van ongebruikte records'. Voor de overige functionaliteiten is nog nader overleg met de opdrachtgever nodig.

5.12 Outline Plan for Development

- Plan van Aanpak	½ week
- Feasibility Study	3 ½ week
- Business Study	3 weken
- Functional Model Iteration	4 weken
- Design and Build Iteration	5 weken
- Implementation	3 weken
- Werken aan het verslag	3 weken
TOTAAL	22 weken

6 Business Study

6.1 inleiding

In dit hoofstuk wordt een selectie getoond uit de rapporten Business Area Definition, System Architecture Definition en Development Plan. Deze rapporten zijn te vinden in de bijlagen.

6.2 doel

Nu vaststaat dat DSDM een geschikte methode is voor het project is het nodig om een goed begrip te verkrijgen over de bedrijfsprocessen die geautomatiseerd moeten worden en over de informatiebehoefte van deze bedrijfsprocessen. Alle producten die tijdens de Business Study geproduceerd worden mogen op een later tijdstip gewijzigd worden.

6.3 Business Area Definition

6.3.1 doel

Op hoog detailnivo vastleggen welke processen geautomatiseerd dienen te worden.
Er dient een eerste versie van het object model met een globale schets van de belangrijke Use Cases gemaakt te worden.

6.3.2 werkwijze

- Onderzoek naar het gebruik van design patterns
  Vlak voor de afstudeerperiode heeft de student kennis gemaakt met design patterns. De student had graag praktijkervaring opgedaan met het gebruik van design patterns, maar heeft helaas geen design patterns gevonden die raakvlakken vertoonden met zijn opdracht. De design patterns die de student in nadere overweging heeft genomen zijn bridge, chain of responsibility en template method. Uiteindelijk heeft template method wel tot grote inspiratie geleid voor het maken van de verwijderprocedure.
  Een basisidee van de student was om een klasse REMtabel te maken die zou bevatten welke koppelingen er waren met de overige tabellen. Als de gebruiker nu bijvoorbeeld de tabel Medewerker zou kiezen om uit te verwijderen dan zou het programma een instantie van REMtabel maken voor Medewerker, runtime opvragen welke koppelingen er met overige tabellen zijn en zelf de SQL statements genereren.
  Tijdens dit onderzoek kwamen er echter nieuwe requirements naar voren en deze zorgden ervoor dat de student het niet wenselijk meer achtte om elke tabel zijn eigen klasse te geven.

- Nieuwe requirements
  - per functionaliteit wordt opnieuw gekeken welke gevolgen het wijzigen/verwijderen van een tabel dient te hebben voor de overige tabellen. Deze gevolgen zijn niet dynamisch op te vragen aan het DBMS. Bijvoorbeeld het verwijderen van Organisatie t/m een bepaalde datum. Volgens het DBMS moeten dan eerst alle Aanstellingen van deze Organisatie verwijderd worden (cascade delete) maar wat de opdrachtgever wil is dat een Organisatie niet verwijderd mag worden zolang er Aanstellingen gekoppeld zijn aan de Organisatie (cascade no action).
  - alle communicatie met de database in één datamodule (klasse/bestand) opnemen.
  - rekening houden met mogelijke vertalingen van het programma in het Engels

- Maken belangrijke Use Cases

  De student heeft nader overleg gevoerd met de opdrachtgever en zijn collega's over de layout van het programma en de presentatie van de data. Er is besloten dat in de linkerkant van het hoofdscherm een balk komt die aangeeft in welk frame de gebruiker zich bevindt. Onderaan het scherm komt een statusbalk.
  In het verwijderen t/m een bepaalde datum scherm zijn er 3 belangrijke acties gedefinieerd, te weten:

  i. het kiezen van een tabel waaruit verwijderd moet worden
  ii. het kiezen van een datum
  iii. het drukken op de knop 'uitvoeren'

  Deze Use Cases worden nu in detail beschreven (Nb. dit zijn de eerste versies).
  Voor alle Use Cases geldt dat de gebruiker de systeembeheerder van de organisatie is.

Uc1 Het kiezen van een tabel waaruit verwijderd moet worden
Proces
1. Selecteer een tabel in de treeview

Postcondities
Het systeem toont feedback in een memoveld over de gevolgen van deze keuze

Uc2 Het kiezen van een datum
Precondities
Er is een tabel geselecteerd

Proces
1. Selecteer een datum in de datumeditbox

Postcondities
- Het systeem toont feedback in een memoveld over de gevolgen van deze keuze
- De knop 'uitvoeren' wordt selecteerbaar

Uc3 Het drukken op de knop uitvoeren
Precondities
Er is een tabel en een datum geselecteerd

Proces
1. Selecteer de knop uitvoeren

Postcondities
- De gegevens zijn verwijderd
- Het systeem toont de voorgang in de statusbalk

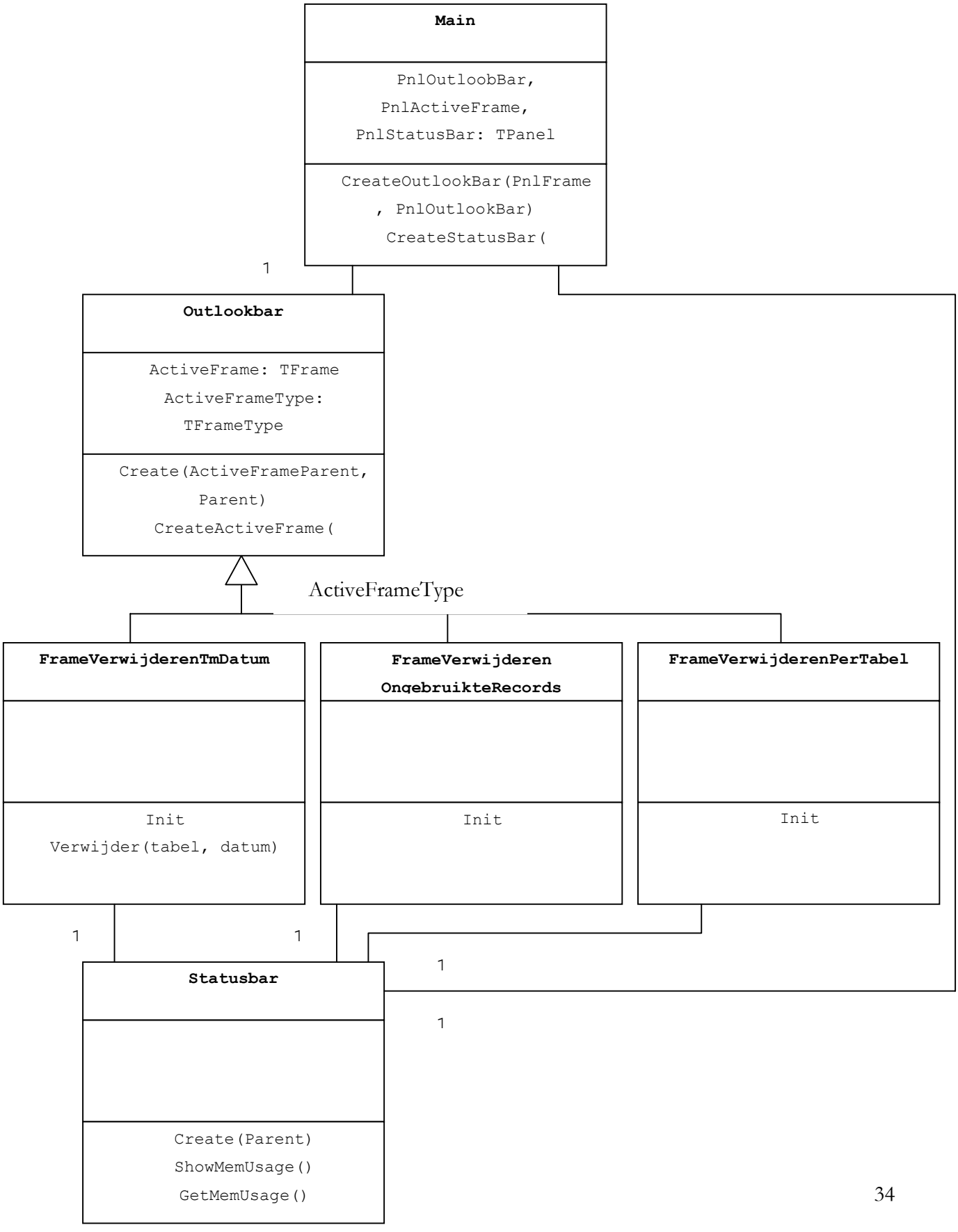
- Maken van het objectmodel

  Nadat de optie dynamisch de sql statements te genereren onhaalbaar bleek en de eis kwam dat alle communicatie met het DBMS in de datamodule klasse plaats diende te vinden heeft de student besloten om af te zien van het plan om een klasse te maken voor de geselecteerde tabel (REMtabel). Voor de duidelijkheid toon ik hieronder een grove schets van het hoofdscherm:

<table>
<tr><td colspan="2">MENUBALK</td></tr>
<tr><td>OUTLOOKBAR PANEL</td><td>ACTIVE FRAME PANEL</td></tr>
<tr><td colspan="2">STATUSBAR PANEL</td></tr>
</table>

Het idee is als volgt: Het frame Outlookbar toont in welk scherm de gebruiker zich bevindt en kan ook gebruikt worden om van scherm te wisselen. De klasse FrameOutlookBar creert het active frame en toont dit. De statusbar is gekoppeld aan het hoofdscherm (want hier wordt het FrameStatusBar gecreeerd) en aan het active frame. Dit active frame is bijvoorbeeld een instantie van FrameVerwijderenTmDatum of een instantie van FrameVerwijderenOngebruikteRecords. De eerste versie van het definitieve klassediagram vindt u op de volgende pagina.

6.3.3 De eerste versie van het definitieve klassediagram



6.4 System Architecture Definition

6.4.1 doel

Benoemen van de ontwikkel-en doelplatforms en de architectuur van de te ontwikkelen software (belangrijke componenten en interfaces).

6.4.2 werkwijze

De student heeft in overleg met de opdrachtgever en collega's het volgende vastgesteld:

- ontwikkelplatform: Pentium III PC met Windows 2000, Delphi 7 en PostgreSQL
- doelplatforms: PC met 32 bits Windows en PostgreSQL, Oracle of MS-SQLServer
- architectuur van de te ontwikkelen software: de DbExpress connectie bestaat uit de volgende Delphi componenten (op te nemen in de datamodule): TCRSQLConnection, TSQLDataSet, TDataSetProvider, TClientDataSet.

6.5 Development Plan

6.5.1 doel

Het Outline Plan van de Feasibility Study verfijnen met een prototyping strategy, een testing strategy en een configuration management plan.

6.5.2 werkwijze

De software wordt incrementeel ontwikkeld. De student streeft ernaar om in incrementen van een week te werken zodat tijdens elke softwarevergadering feedback verkregen kan worden op het nieuwste increment. De student zal geen prototype maken van het gehele programma omdat hij hier geen reden toe ziet.

De student heeft onderzoek verricht naar het werken met Unit Tests. Dit is een onderdeel van Extreme Programming (XP) en houdt in dat elk stukje code ook een test voor die code bevat. Met een Unit Test Framework is het mogelijk om na elke wijziging in het programma automatisch alle testen opnieuw uit te voeren. Als de wijziging tot een nieuwe programmafout leidt dan wordt dit direct herkend, en kan de wijziging zonder veel moeite worden teruggedraaid. De student heeft op internet enkele excellente Unit Test FrameWorks gevonden doch alle voor Java of SmallTalk. De Unit Test Frameworks voor Delphi die de student vond waren alle echter van slechte kwaliteit. De student heeft nog overwogen om zelf een Unit Test Framework te schrijven maar uiteindelijk, met tegenzin, toch besloten dat de tijdwinst van de Unit Tests niet tegen de moeite op zou wegen.

De acceptatietest van de applicatie zal niet plaatsvinden door de student maar door de opdrachtgever. Ook de tests met Oracle en SQLServer zullen door de opdrachtgever gedaan worden. De student zal nieuwe functionaliteiten onmiddelijk testen met een kleine productiedatabase en zal wel de pre-acceptatietest zelf uitvoeren. Voor de pre-acceptatietest zal de student een kleine database handmatig vullen en zo de resultaten van het verwijderen controleren. De opdrachtgever zal het programma testen met een grote productiedatabase.

De student kan zich van IP-03 nog herinneren dat een solide configuration management plan een project kan maken of breken (positieve herinneringen). Dit is echter een individueel project en daarom lijkt het maken van een apart document de student niet erg zinvol. De student zal aan het eind van elke dag de source code opslaan in een .zip bestand en zal verder alle documentatie nummeren en wekelijks opslaan in een .zip bestand. Alle belangrijke bestanden worden op twee computers opgeslagen.

7 Functional Model Iteration

7.1 doel

De documentatie en software die tijdens de Business Study geproduceerd is verder verfijnen.

7.2 werkwijze

Begin met het verfijnen van de documentatie. In een later stadium zullen de documentatie en de software naast elkaar ontwikkeld worden. De Funcional Model Iteration bestaat uit 4 activiteiten:

1) bepaal wat je doen moet in de cyclus
2) bepaal hoe je het gaat doen
3) doe het
4) controleer of je het goed gedaan hebt

7.2.1 wat er gedaan moet worden en hoe het wordt gedaan

a) Er moet verbinding gelegd worden met de database;
   De code van de Beheermodule wordt bestudeerd en het gedeelte dat de koppeling met de database legt wordt gekopieerd
b) De datamodule moet opgenomen worden in het klassediagram;
   Evenals FrameStatusBar is de DataModule gekoppeld aan zowel Main als aan de instantie van ActiveFrame, te weten FrameVerwijderenTmDatum, FramaVerwijderenOngebruikteRecords of FrameVerwijderenTmTabel.
c) Er moet vastgelegd worden hoe de gekozen tabel en de gevolgen hiervan tijdens de selectie aan de gebruiker getoond gaan worden;
   In een treeview met radiobuttons en een gekleurd vierkant om de selectie.
d) Er moet een onderverdeling komen in verwijderen t/m datum en verwijderen t/m week;
   FrameVerwijderenTmDatum bevat een variabele die het type aangeeft (week of datum).

e) Er moet definitief vastgelegd worden welke tabellen er gekozen kunnen worden om te verwijderen en wat de gevolgen van elke keuze zullen zijn;

   De student heeft hiervoor drie ideeen aangedragen waarvan er een uitgekozen is. Na overleg met de databasebeheerder zijn de namen die de gebruiker te zien krijgt op het scherm vastgesteld.

f) Er moet feedback getoond worden na een selectie van een tabel/datum. In het geval van een cascade no action dienen de records die het verwijderen tegenhouden getoond te worden aan de gebruiker (mits minder dan 10 records);

   De feedback kan globaal of gedetailleerd zijn. Dit is in te stellen door de gebruiker. De gedetailleerde beschrijving is technischer en staat dichter bij de representatie in het DBMS. De globale beschrijving beschrijft meer concepten die bij de gebruiker bekend zijn en laat de minder belangrijke informatie weg. Het tonen van de records bleek niet eenvoudig omdat deze bij voorkeur in een memoveld getoond dienden te worden aangezien alle overige feedback hier ook staat. De student heeft een component gevonden dat records converteert naar tekst.

g) De use case voor het selecteren van een tabel moet worden uitgebreid;

   Allereerst moeten nu eerst alle voorgaande selecties ongedaan gemaakt worden. Verder moeten alle onderliggende tabellen ook geselecteerd worden en dient er nu een controle plaats te vinden of de tabel wel verwijderd mag worden (cascade no action) en er dient bepaald te worden of de tabel week- danwel datumgebonden is.

h) Er moet getest worden of het mogelijk is om alle gegevens uit een tabel te verwijderen t/m een bepaalde week (de tabel Week is de testtabel);

   Deze test is geslaagd. Wel is er grote onenigheid tussen de collega's ontstaan over de te volgen methode hiervoor. De ene groep wil dat het verwijderen in een groot commando geschiedt, de andere groep wil dat dit door middel van duizend kleine commando's gebeurt. De student heeft zijn voorkeur maar heeft instructies gekregen dat de duizend kleine commando's methode gevolgd dient te worden omdat er zo voortgangsinformatie getoond kan worden.

7.2.2 controleer of je het goed gedaan hebt

a) ok
b) ok
c) niet ok. de recursieve procedure blijkt niet te werken als de volgorde van de tabellen wisselt
d) ok
e) ok, zie het document 'Consequenties van de verschillende opties bij het opschonen op datum' in de bijlagen
f) nog niet ok wat het tonen van records als tekst. Het gebruikte component blijkt vol met bugs te zitten waar omheen geprogrammeerd moet worden.
g) ok
h) ok, de student is bezig met het testen van de snelheid van de verschillende mogelijkheden. Het verwijderen in een groot commando kost 7 seconden terwijl de duizend kleine commando's 32 seconden duren.

7.3 Prioritized functions

Hier moet de student beslissen welke functionaliteit na deze fase gereed zal zijn. Omdat de student veel problemen tegenkomt bij de no action controle zal deze nog niet gereed zijn aan het eind van deze fase. Het verwijderen is geimplementeerd maar de student is ook hierover nog niet 100% tevreden. Dit zal slechts geimplementeerd worden voor de tabel Weken. Als de student een betere methode gevonden heeft zal hij ook de code schrijven voor de overige tabellen.

7.4 Funcional Prototyping Review Documents

Het commentaar van gebruikers dient bewaard te worden. De student maakt tijdens elke softwarevergadering aantekeningen die hij daarna digitaal opslaat op datum.

7.5 Non-functional requirements

- Er dient voortgangsinformatie getoond te worden
- Records die het verwijderen tegenhouden moeten getoond worden
- De gebruiker moet 2x inloggen, de eerste keer met het DBA password, de tweede keer met het Big Ben password.
- (wens van de student) De student hoeft nooit in te loggen als hij het programma met een bepaalde vlag opstart
- (wens van de student) Het programma speelt een geluidje als het verwijderen klaar is. Dit doen sommige CD-branders ook als het brandproces voltooid is en maakt het mogelijk om even weg te lopen van de PC en direct weer terug te keren als het werk voltooid is. Sommige collega's zijn hierop tegen, daarom zal dit een optie worden in het menu
- (wens van de student) Het programma is ook goed bestuurbaar met het toetsenbord. Dit heeft nogal wat voeten in de aarde omdat Delphi op een dubieuze manier met TABS omgaat.
- Het programma slaat informatie over acties en resultaten op in een logfile (het TSQLMonitor component van Delphi 7 Enterprise Edition slaat alle communicatie met het DBMS op)
- De gebruiker dient de grootte van de componenten op het scherm aan te kunnen passen (het programma moet deze wijzigingen onthouden).
- Het programma bevat een Help→About scherm
- De student dient een exotisch kleurenschema te kiezen in windows om te testen of alle teksten nog steeds goed leesbaar zijn.

7.6 Implementation Plan

Dit rapport is niet van toepassing omdat de te ontwikkelen module pas aan de klanten uitgeleverd zal worden als alle functionaliteiten af zijn, en dit is niet te halen binnen deze afstudeerperiode. Vanwege het testen van het programma door de student op de PC in de vergaderruimte en door de opdrachtgever op diens PC is de student er overigens van overtuigd dat het niet moeilijk zal zijn om het programma te implementeren. Het programma is meermalen opnieuw geinstalleerd op een 'kale' PC en alle benodigde bestanden staan bij elkaar

in een map. De opdrachtgever heeft in zijn eigen rooster al ruimte vrijgemaakt voor het testen van het programma.

8 Design and Build Iteration

8.1 doel

Het systeem zover ontwerpen dat het de meest gewenste functionaliteiten van de gebruiker bevat. Aan het eind van deze fase moet het Tested System gereed zijn. Dit is de geteste applicatie.

8.2 werkwijze

Het verrichten van alle werkzaamheden die zijn blijven liggen tijdens de vorige fase. Het testen op bugs en het verwerken van nieuwe requirements.

8.3 nieuwe requirements

- als het inloggen mislukt genereert het programma een systeemfout. Deze bug moet er uit!
- het programma maakt niet alleen onderscheid tussen verwijderen t/m datum en verwijderen t/m week maar ook tussen verwijderen t/m budgetperiode
- het programma dient een lijst van alle budgetperioden te kunnen tonen
- het programma dient de namen van de dynamische codetabellen dynamisch in te lezen
- de feedback moet minder technisch worden en meer in gebruiktertaal
- de gebruiker moet op kunnen vragen in welke database hij werkt
- het TSQLMonitor component, gebruikt voor de logfile, mag niet meer gebruikt worden
- het niet-inlog achterdeurtje van de student moet dichtgetimmerd worden
- de no action controle moet sneller (duurt 3 minuten bij een reusachtige database)
- het programma moet voor het verwijderen een popup waarschuwing geven
- het programma moet tijdens het verwijderen controle teruggeven aan windows
- het programma mag tijdens het verwijderen niet afgesloten kunnen worden, want dit kan de database corrumperen.
- het programma moet na het verwijderen een popup tonen met de resultaten

8.4 Selectie van code van verwijderen t/m datum procedure

Het basisidee is dat de Uitvoeren actie in het Verwijderen t/m datum scherm meegeeft wat voor commando er uitgevoerd dient te worden, op welke tabel, en met welke parameters (variabel). De commando's in het vet verwijzen naar abstracte procedures die afhankelijk van het type commando de juiste statements inlezen. Voor het toevoegen van een update functionaliteit hoeft er dus slechts in elke vetgedrukte proecedure een actie gedefinieerd te worden voor de update. De rest van de gehele procedure kan dus in zijn geheel hergebruikt worden. De rest van de procedure regelt onder andere het tonen van de voortgangsinformatie, het tonen van errors, het tonen van de resultaten, het bijwerken van de logfile en het afspelen van een geluid als de databasebewerking klaar is.

Het verwijderen geschiedt niet per record maar per aantal records tegelijk(zie 8.5.13 Extract uit de logfile). Deze methode combineert de snelheid van een groot commando met de mogelijkheid tot voortgangsinformatie en de tussentijdse afhandeling van windowsmessages.

```
  function TFrmDataModule.Execute( const PenSqlCommandType: TenSqlCommandType; const PcsTable:
string; const PvarParams: Variant ): TrecExecuteCommandResult;
  var
    XinParamIndex, XinCommandTextIndex: Integer; // nodig voor het stappen door de master
tabel
    XinStepSize: Integer; // stapgrootte voor de master tabel
    XrecSqlCommandMaster, XrecSqlCommand: TrecSqlCommand; // regels met commandtexts en params
    XcsParams: string; // lijst met params als string: bijv. ':Param0, :Param1, :Param2'
    XlgStop: Boolean;
  begin
    XdtBeginTimeFunction := Now;
    XrecSqlCommandMaster := GetMasterCommandRec( PenSqlCommandType, PcsTable, PvarParams );
    XdtBeginTimeFirstStep := Now;

          // Walk with XinStepSize paces through the records of master table.
          // Every 'stepped' record of master table may have coupled records in other tables
          // which will have to be affected first ( in the correct order ).
          // After the 'affection' of the coupled records, affect the stepped records of
master table
          while (( not TblMaster.Eof )) do
          begin
            // if ( at the beginning of a new step )
            if ( XinParamIndex = 0 ) then
            begin
              // create the start of every command text sentence
              XrecSqlCommand.lstCommandText := GetExecuteCommands( PenSqlCommandType, PcsTable
);
```

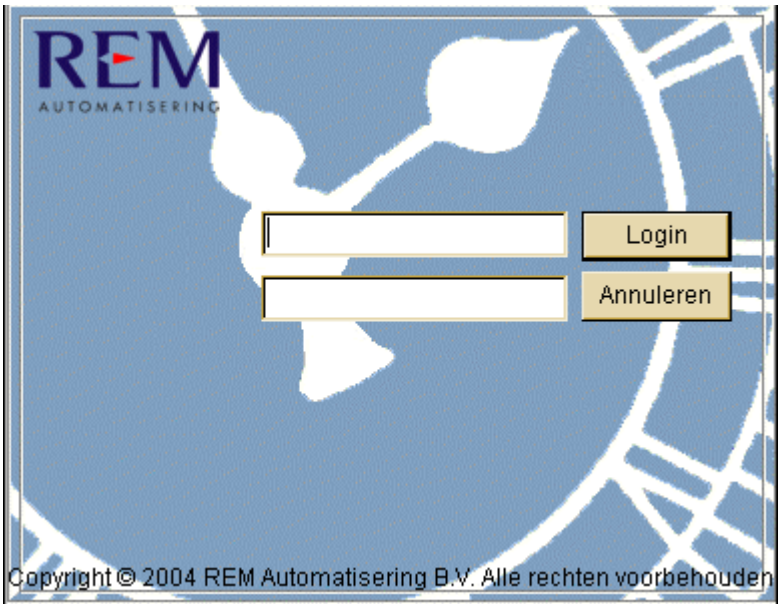
```
              XcsParams := ':Param' + IntToStr( XinParamIndex );
            end
            else
            begin
              // add an extra parameter to the command texts
              XcsParams := XcsParams + ', :Param' + IntToStr( XinParamIndex );
            end; // if

            if not ( XlgStop ) then
            begin
              // fill the parameter
              AddParam( PenSqlCommandType, PcsTable, XinParamIndex, XrecSqlCommand.objParams
);

              // if (at end of step) or (at end of master table)
              // Without the test 'at end of master table' the last step may not be properly
closed
              // when mastertable.recordcount mod stepsize <> 0.
              if ( ( XinParamIndex = XinStepSize - 1 ) or ( TblMaster.RecNo =
TblMaster.RecordCount ) ) then
              begin
                // replace the ':Param' string with XcsParams, execute the commands and save
the number of records affected
                for XinCommandTextIndex := 0 to XrecSqlCommand.lstCommandText.Count - 1 do
                begin
                  Inc( RESULT.inRecordsAffected, UDataModuleFuncties.ExecuteCommand( CmdTemp,
StringReplace( XrecSqlCommand.lstCommandText.Strings[XinCommandTextIndex], ':Param',
XcsParams, [rfReplaceAll, rfIgnoreCase] ), XrecSqlCommand.objParams, FALSE ) );

                // process progress info and show it
                if ( CanShowProgressInfo ) then
                begin
                  XflMsecSpentSoFar := MilliSecondSpan( Now, XdtBeginTimeFirstStep );
                  XflEstimatedMsecLeft := ( XflMsecSpentSoFar / TblMaster.RecNo ) * (
TblMaster.RecordCount - TblMaster.RecNo );
                  FrameStatusBar.StepIt;
                  FrameStatusBar.ShowProgressInfo( UFuncties.GetMinSecTimeStr( Round(
XflEstimatedMsecLeft ) ) );
                end; // if

                // begin a new step
                XinParamIndex := 0;
              end
              else
              begin
                // stay in the same step
                Inc( XinParamIndex );
              end; // if

              TblMaster.Next;
```

```
          end; // if
        end; // while

    {-------------------------------------------------------------------
    ------------------- START AFTER EXECUTE ----------------------------
    -------------------------------------------------------------------
      De acties op de master table en de gekoppelde tabellen zijn nu voltooid.
      Voor de meeste waarden van PcsTable hoeft er geen After Execute
      actie plaats te vinden.

      Soms geldt echter:
        'TblMaster <> PcsTable'

      In deze gevallen moet er nog een AfterExecute actie uitgevoerd
      worden op PcsTable.

      Bijvoorbeeld bij TBBudgetPeriode:
      Omdat TblMaster 1 regel zou bevatten als deze gekoppeld zou zijn aan TBBudgetPeriode
      is ervoor gekozen om op de regels van TBBudgetVersie te lopen ('TblMaster :=
TBBudgetVersie').
      Na de Exectue actie is de regel uit de TBBudgetPeriode nog niet bijgewerkt.
      Het bijwerken van deze regel geschiedt in de AfterExecute actie.
    -------------------------------------------------------------------}
    XrecSqlCommand := GetAfterExecuteCommandRec( PenSqlCommandType, PcsTable, PvarParams );

        Inc( RESULT.inRecordsAffected, UDataModuleFuncties.ExecuteCommand( CmdTemp,
XrecSqlCommand.lstCommandText.Strings[0], XrecSqlCommand.objParams, FALSE ) );

    {-------------------------------------------------------------------}
    {------------------ AFTER EXECUTE FINISHED -------------------------}
    {-------------------------------------------------------------------}

    // save total number of affected records and the elapsed time
    // Time Elapsed = Now - XdtBeginTime
    DecodeTime( Now - XdtBeginTimeFunction, XinHour, XinMin, XinSec, XinMSec );
    RESULT.csTimeElapsed := CcsTimeElapsed + UFuncties.GetTimeElapsedStr( XinHour, XinMin,
XinSec, XinMSec ) + #13#10 + CcsHourMinSec;

    // als er minimaal 1 regel bijgewerkt is speel dan een geluid om aan te geven dat het
klaar is
    if ( ( RESULT.inRecordsAffected > 0 ) and ( GlgSoundEnabled ) ) then
    begin
      try
        MMSystem.PlaySound( 'tada.wav', SND_ALIAS, SND_ASYNC );
      except
        Application.ProcessMessages;
      end; // try
    end; // if
```

8.5 Tested System

Deze paragraaf zal de schermen van de applicatie tonen met uitleg over het gebruik.

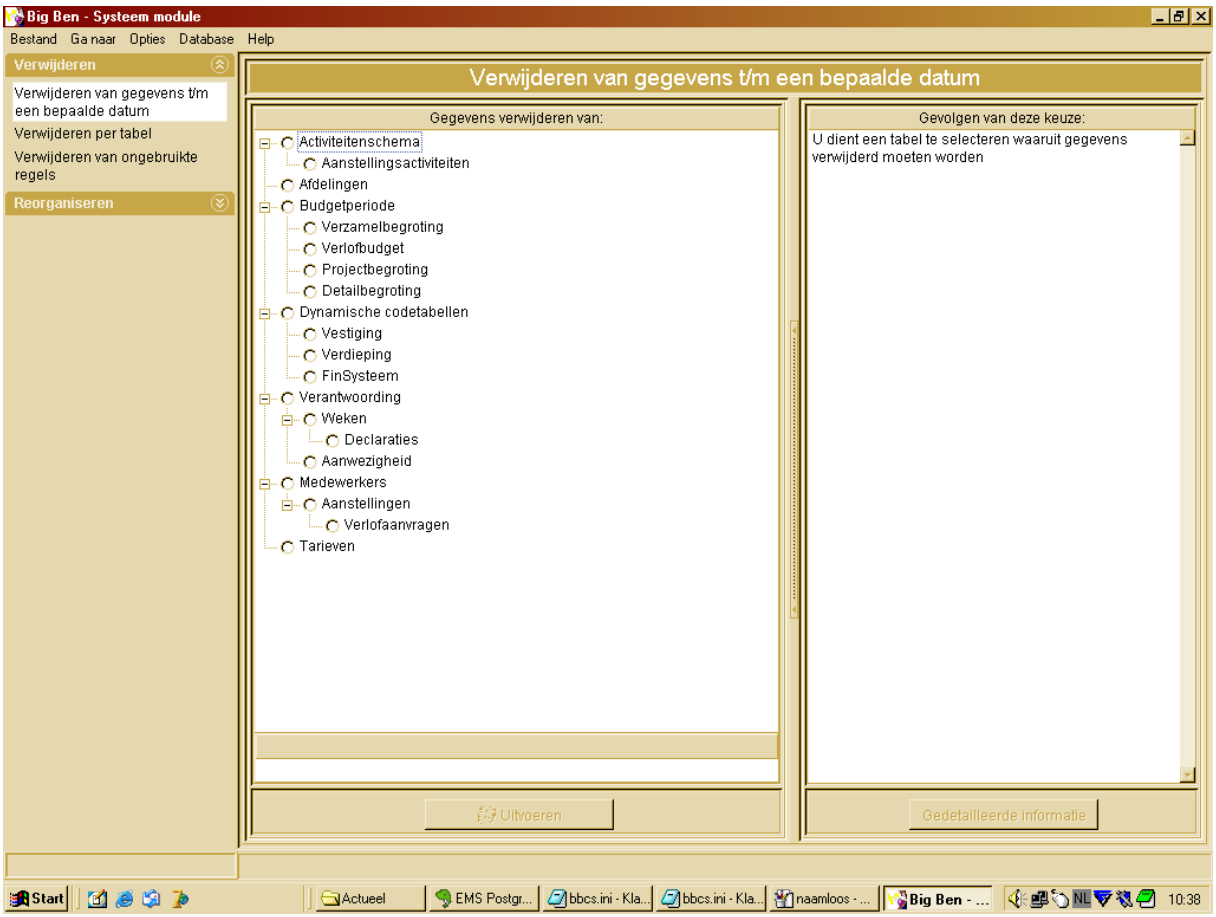
8.5.1 Inloggen met Big Ben wachtwoord



8.5.2 Niet in onderhoud controle heeft gefaald

8.5.3 Opstartscherm

Er is standaard nog geen tabel geselecteerd. Links in het scherm ziet u het FrameOutlookBar, onderin het FrameStatusBar en de rest van het scherm wordt in beslag genomen door FrameVerwijderenTmDatum.

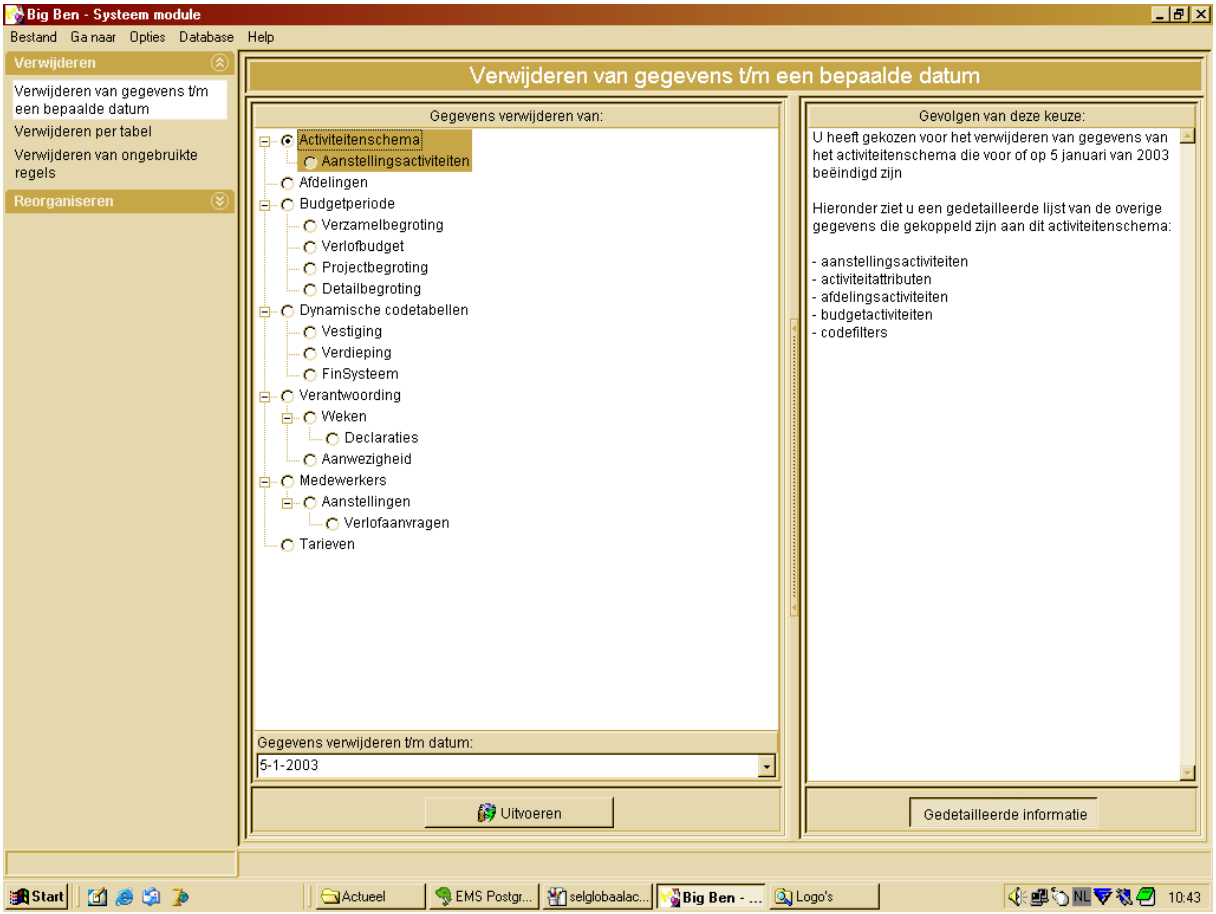


8.5.4 Selectie van de tabel Activiteitenschema met gedetailleerde feedback

De gekoppelde tabel Aanstellingsactiviteiten wordt ook geselecteerd. In het document 'Consequenties van de verschillende opties bij het opschonen op datum' is het volgende te lezen over het verwijderen van Activiteitenschema:

```
ActiviteitenSchema (Activiteitenschema)
      X Verantwoording
      D Aanstellings Activiteiten
      D Activiteit Attributen
      X Budget Detail
      D Budget Items
      D Code Filter
      D Organisatie Activiteiten
```

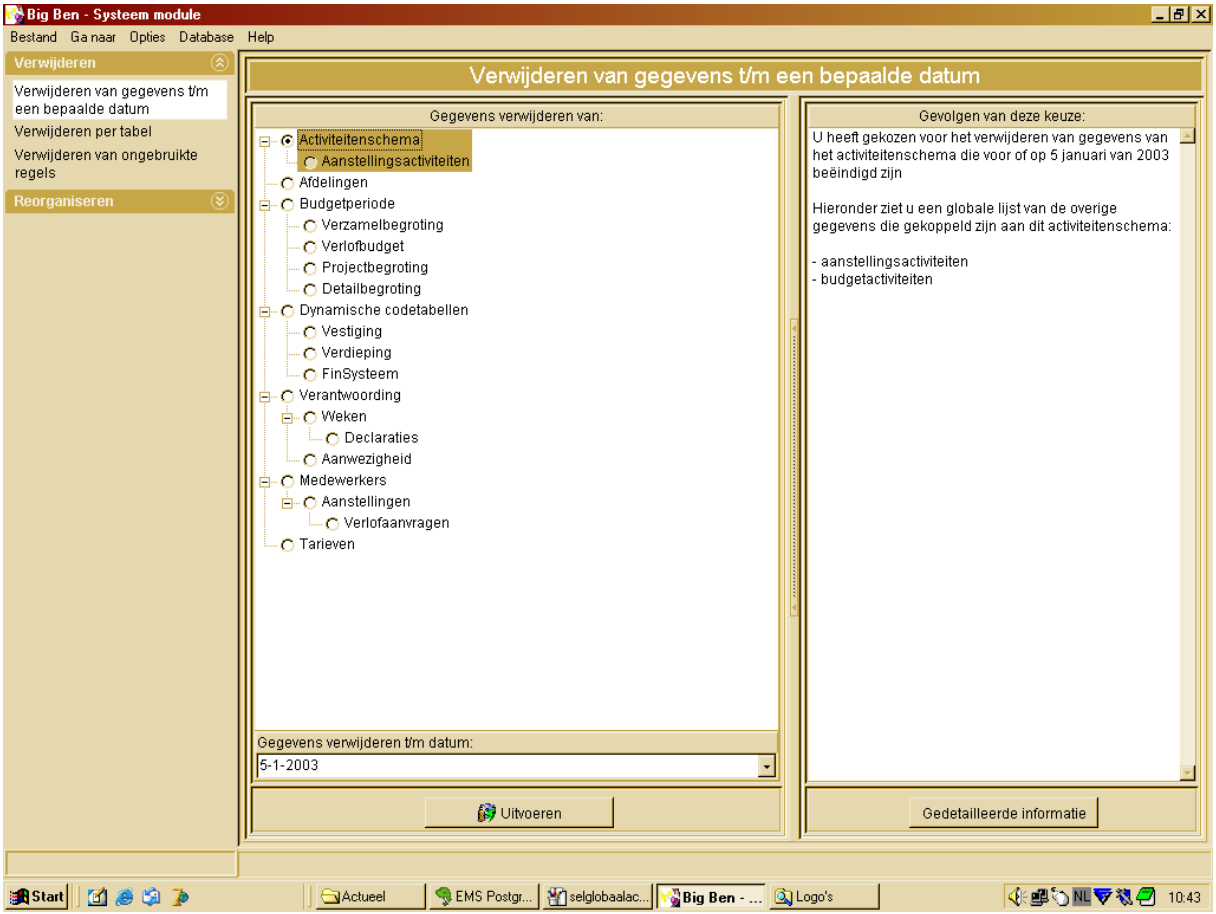
Vergelijk dit eens met de feedback rechts in het scherm. Wat opvalt is dat Budget Items in de feedback 'budgetactiviteiten' wordt genoemd en dat Organisatie Activiteiten bij de gebruiker bekend is als 'afdelingsactiviteiten'. De tabellen Verantwoording en Budget Detail worden in het geheel niet genoemd. Dit komt omdat deze tabellen het verwijderen blokkeren als ze gekoppeld zijn aan Activiteitenschema. De no action controle heeft geen koppelingen met Verantwoording en Budget Detail gevonden dus het verwijderen is toegestaan.



8.5.5 Selectie van de tabel Activiteitenschema met globale feedback

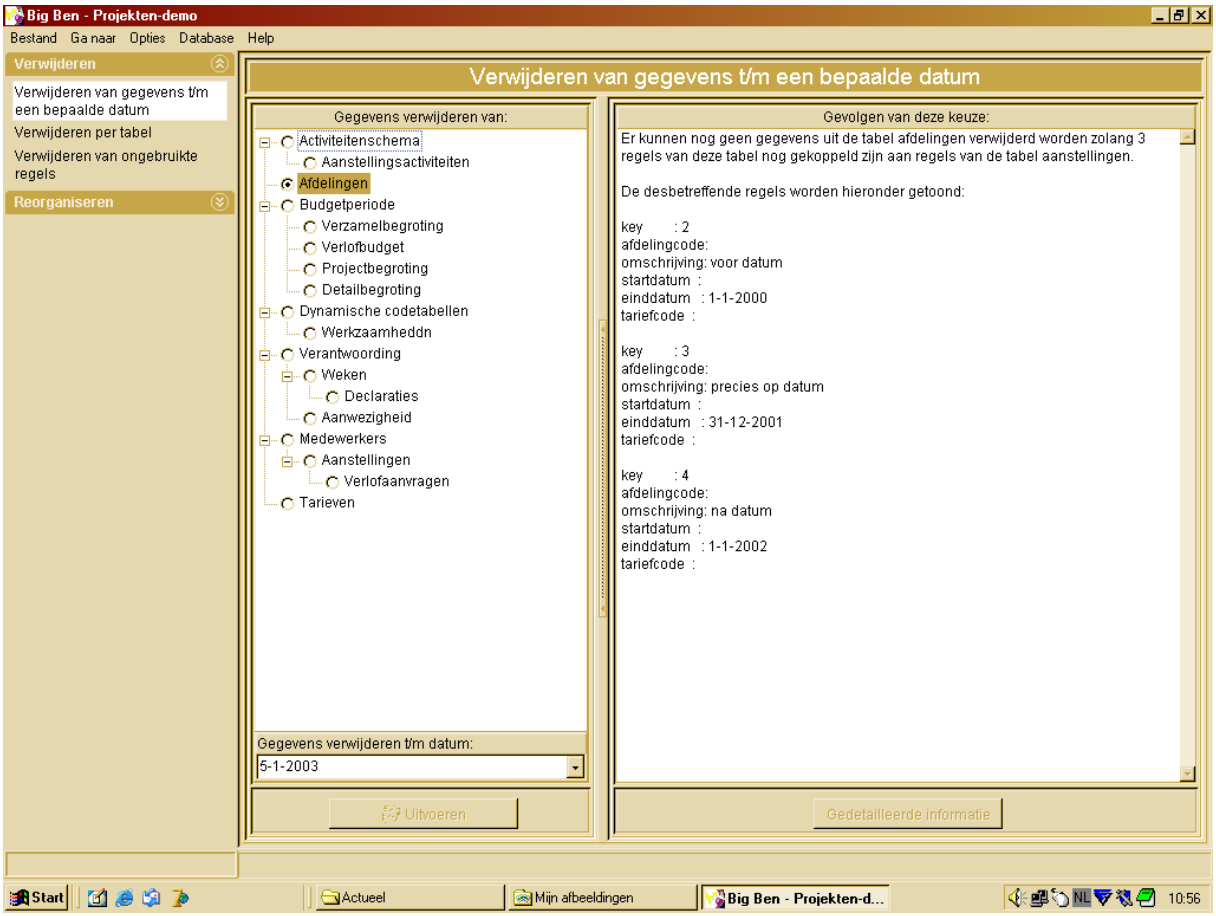
Een van de lastigste keuzes die tijdens het afstudeertraject gemaakt moesten worden bleek de presentatie van de data te zijn. Wat de student graag had gewild was dat in dit voorbeeld ook de tabel Aanstellingsactiviteiten een zwart selectiebolletje in de radio button gehad kon hebben. Dit zou achter tot inconsistentie leiden omdat als bijvoorbeeld de tabel Budgetperiode geselecteerd zou worden alle vier de onderliggende tabellen een zwart bolletje zouden moeten krijgen. Omdat de vier tabellen echter op hetzelfde nivo in de tree staan kan slechts één tabel een bolletje krijgen.

Een andere mogelijkheid is het gebruik van checkboxes in plaats van radio buttons geweest. Dit lijkt echter de mogelijkheid van multiselect te beloven, en het was een specifieke eis van de opdrachtgever dat dit niet mogelijk mocht zijn. Al met al is de huidige oplossing de beste die de student kon bedenken, maar zeker niet ideaal. Het omgeven van de selectie met een gekleurd vakje bleek overigens niet zo eenvoudig. Het Treeview component had hier geen property voor en er moest speciaal een OnBeforePaint event voor aangeroepen worden.



8.5.6 Selectie van de tabel Afdelingen

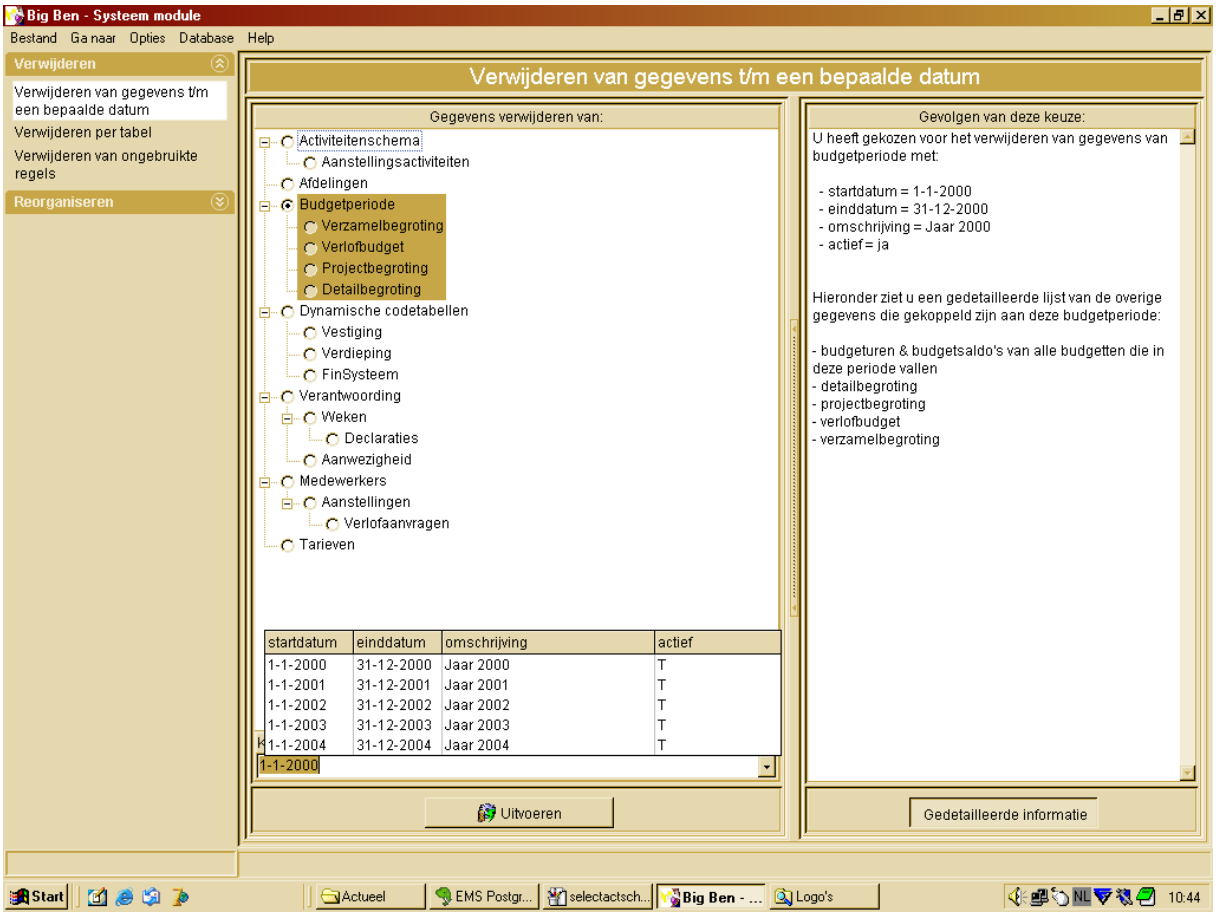
Een afdeling mag niet verwijderd worden als er aanstellingen gekoppeld zijn aan de afdeling. Dit is hier het geval. Omdat het aantal klein is wordt de inhoud van de records getoond. Met behulp van een overige Big Ben module kunnen deze records indien gewenst verwijderd worden.



8.5.7 Selectie van de tabel Budgetperiode

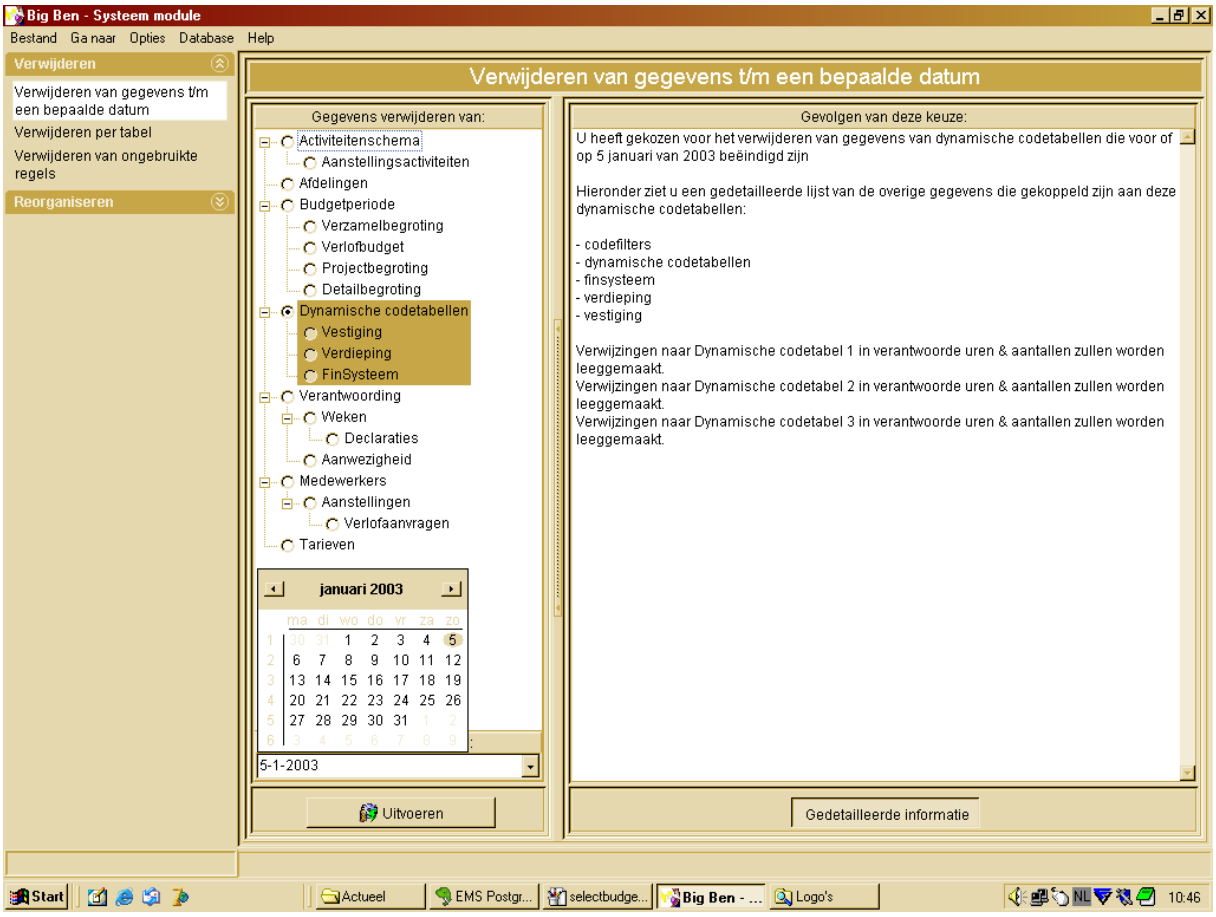
De inhoud van de tabel Budgetperiode wordt getoond. De gebruiker moet een record kiezen.

Als het laatste record verwijderd wordt dan verdwijnt het item Budgetperiode uit de tree.



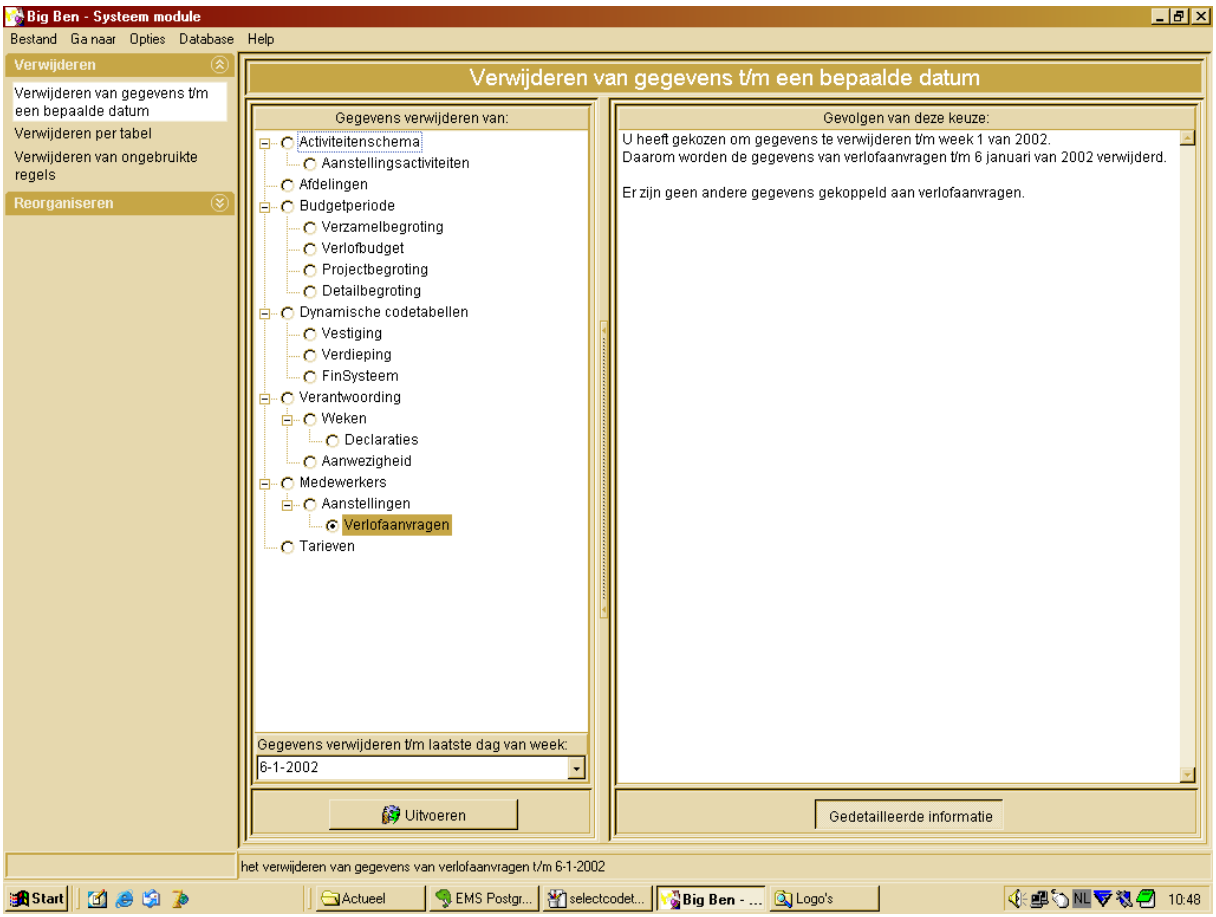
8.5.8 Selectie van de tabel Dynamische Codetabellen

Hier ziet u de datumselectiebox. In de feedback is te lezen hoe cascade set null wordt uitgelegd. De namen 'Vestiging', 'Verdieping' en 'FinSysteem' worden dynamisch ingelezen.



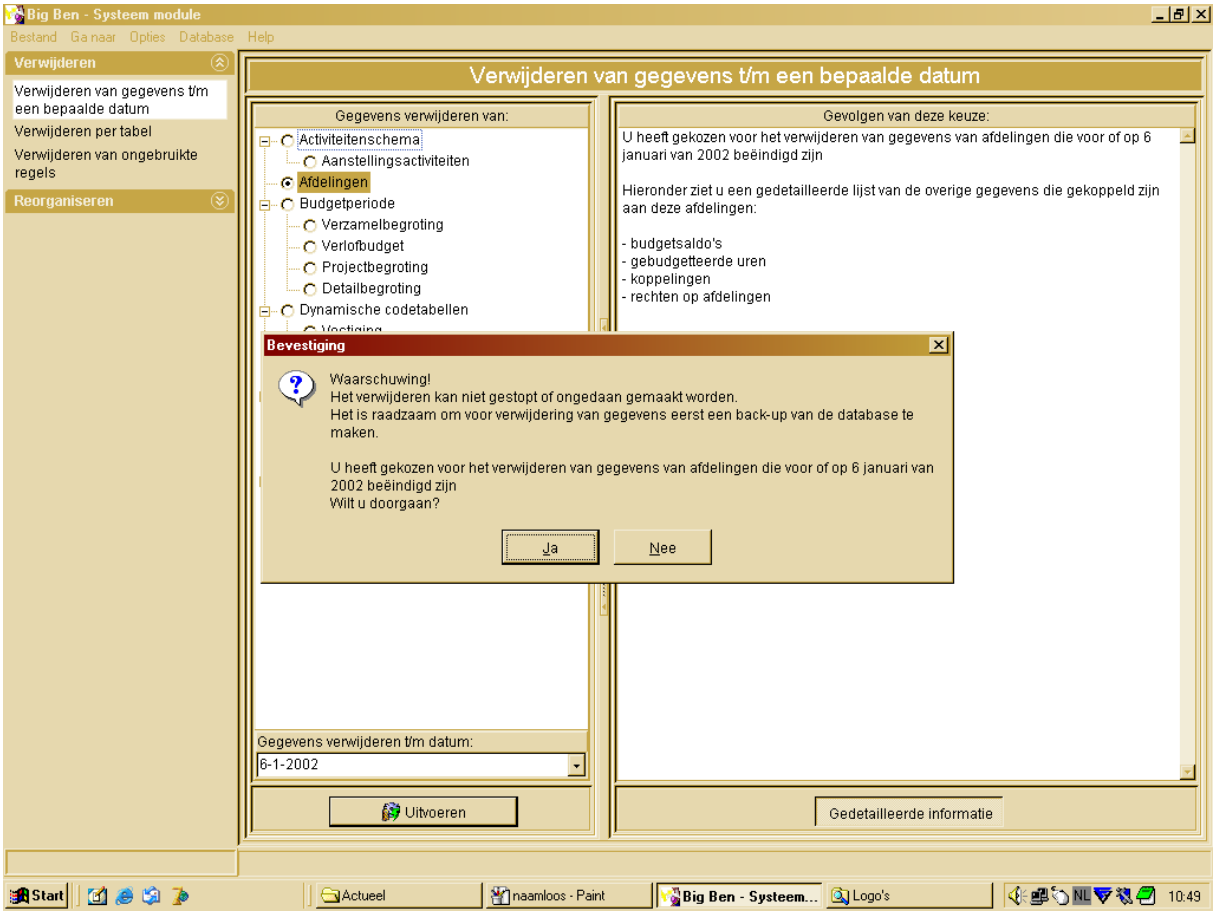
8.5.9 Selectie van de tabel Verlofaanvragen

Dit is de eerste tabel die als voorbeeld gebruikt wordt die per week verwijderd wordt. De gebruiker kiest een datum in de datumselectiebox en automatisch wordt dit veranderd in de laatste dag van die week. De statusbar toont een hint want de cursor staat bovenop de knop 'Uitvoeren'.



8.5.10 Selectie van de knop Uitvoeren

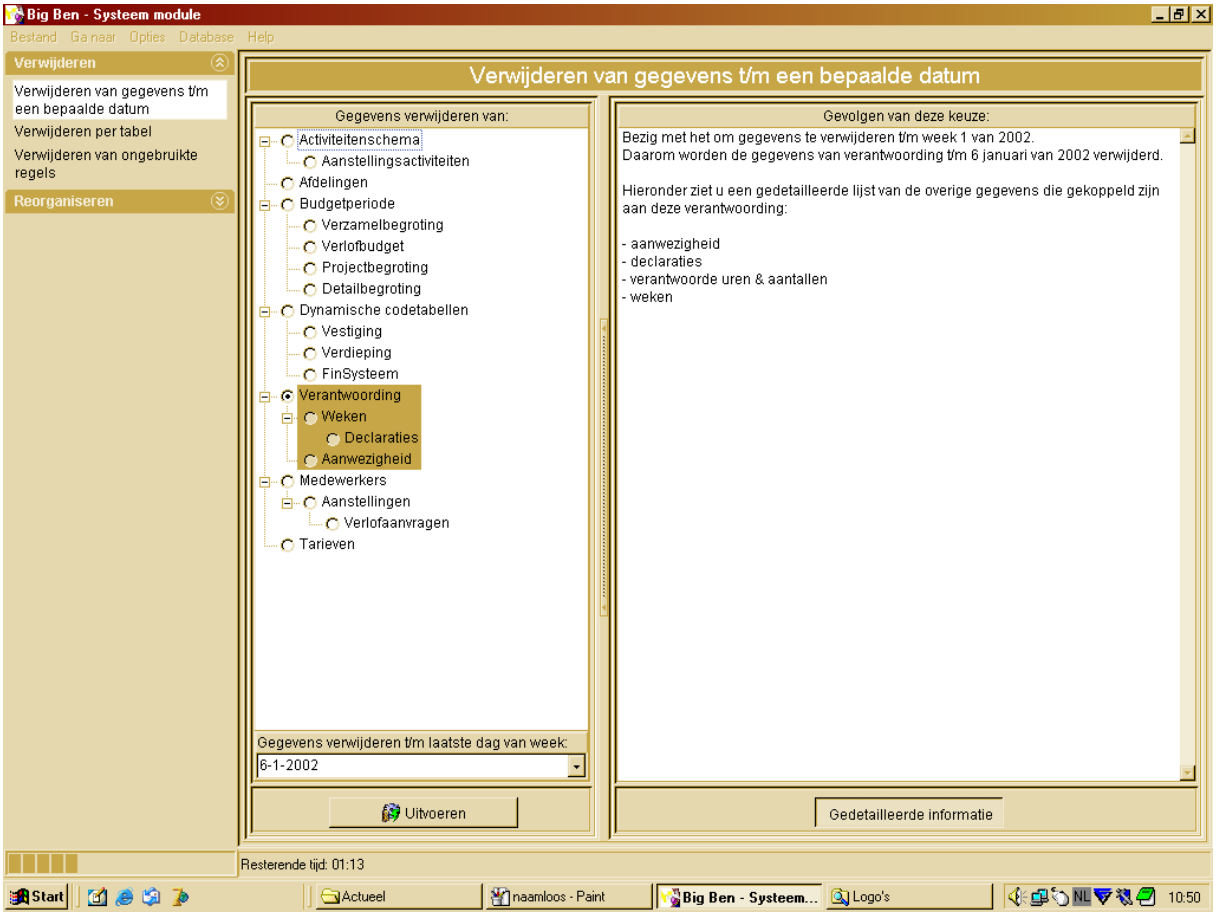
De waarschuwing is misschien vervelend voor de gebruiker maar de Systeemmodule is niet bedoeld voor dagelijks gebruik. Het is de bedoeling dat de gebruiker zeer goed nadenkt over zijn acties, want het terugdraaien van een verwijdering (rollback) is niet mogelijk. De hoeveelheid verwijderde data is in het algemeen namelijk veel te groot voor de rollback-segmenten van een DBMS. Na korte tijd zou het het programma dan overspoeld worden door het DBMS met foutmeldingen dat alle rollback-segmenten volzitten.



8.5.11 Bezig met het verwijderen

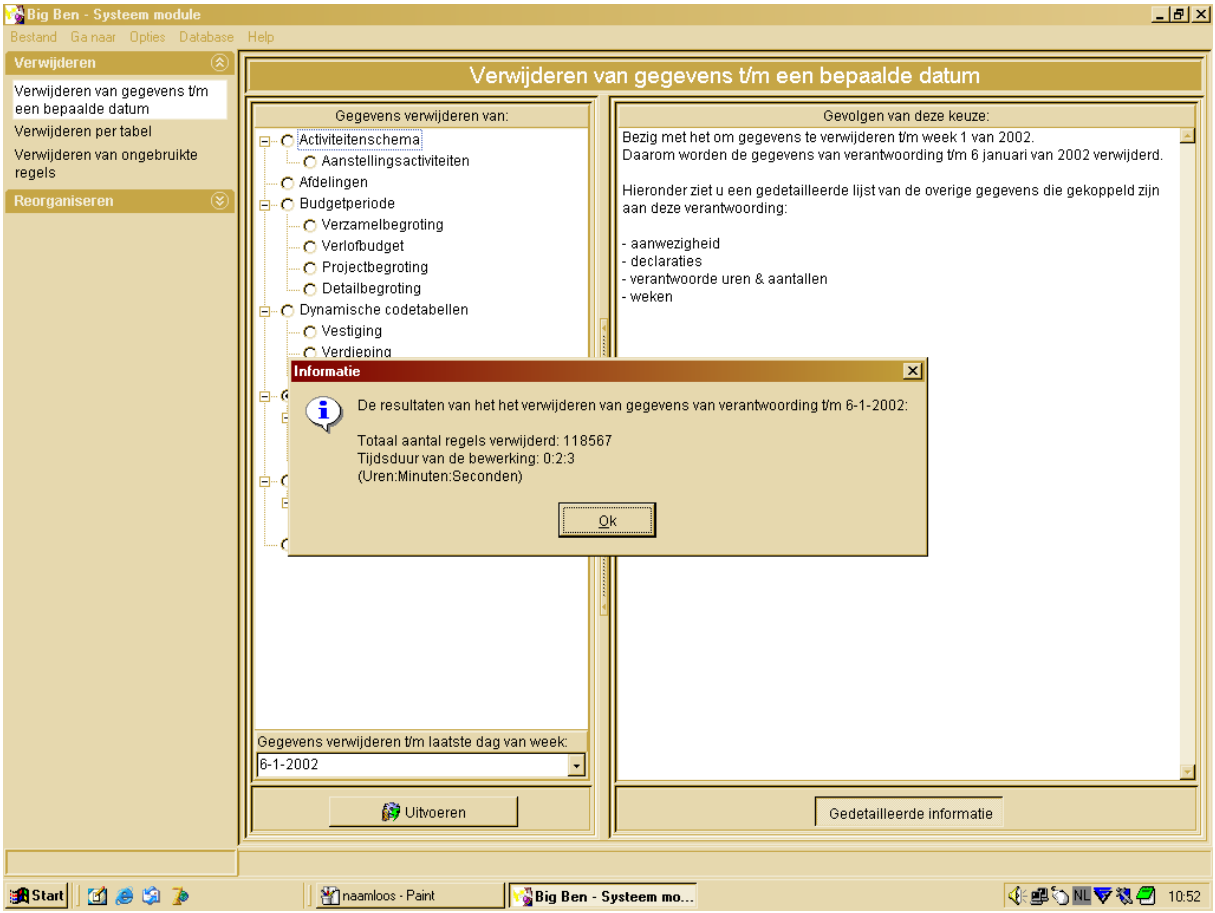
Linksonder in de statusbalk ziet u de voortgans van het verwijderen. De resterende tijd wordt ook ingeschat al is deze inschatting in het begin vaak te pessimistisch. Als het programma bijvoorbeeld klaar is met het verwijderen van 1% van alle Verantwoordingen en dit duurde 3 seconden, dan rekent het uit dat er nog 99% maal 3 seconden resterende tijd nodig is.

Het programma is getest met een reusachtige database en sommige operaties konden wel 7 minuten duren. Wat hierbij naar voren kwam was dat het programma geen commando's van de gebruiker accepteert tijdens een databasebewerking, behalve het commando 'scherm minimaliseren'. Dit bleek op de valreep nog een fors probleem te zijn aangezien het verwijderen in geminimaliseerde toestand niet altijd goed ging. De oplossing bestond uit het afhandelen van windows messages na een bepaald aantal verwijderde records. Deze oplossing hield echter in dat alle knoppen op non-actief gezet moesten worden tijdens een databasebewerking en dat het programma niet zomaar afgesloten mag worden. Bij het afsluiten wordt er nu een dialoog getoond die waarschuwt voor het afsluiten tijdens een databasebewerking.



8.5.12 Resultaten van het verwijderen

Zoals u kunt zien is het programma razendsnel. Als er errors hadden plaatsgevonden dan werden deze automatisch getoond. Na het sluiten van deze dialoog is er geen tabel meer geselecteerd.



8.5.13 Extract uit de logfile

De logfiles hebben allen een naam als bbcs_jaren-maanden-dagen_uren-minuten-seconden-milliseconden. Een logfile kan 1000 items bevatten. De milliseconden zijn opgenomen in de naam om ervoor te zorgen dat elke logfile uniek is. Bij de eerste keer opstarten van het programma wordt een leeg tekstbestand 'bbcs_logtest' geschreven om te testen of de gebruiker schrijfrechten heeft. Zo niet dan weigert het programma om te starten omdat loginformatie onmisbaar wordt geacht.

```
  Bezig met het het controleren of er uit de tabel activiteitenschema verwijderd mag
worden.
  select count(DnKey) as RecCount from  tbactiviteitenschema where DdEind is not
null and DdEind <= '2003-01-05 23:59:59' and exists (select DnKeyActiviteitenSchema
from TbVerantwoording where TbActiviteitenSchema.DnKey =
TbVerantwoording.DnKeyActiviteitenSchema)
  Got 1 tuples from server
  Aantal regels geteld: 0
  select count(DnKey) as RecCount from  tbactiviteitenschema where DdEind is not
null and DdEind <= '2003-01-05 23:59:59' and exists (select DnKeyActiviteitenSchema
from TbBudgetDetail where TbActiviteitenSchema.DnKey =
TbBudgetDetail.DnKeyActiviteitenSchema)
  Got 1 tuples from server
  Aantal regels geteld: 0

  De resultaten van het het verwijderen van gegevens van verantwoording t/m 6-1-
2002:
  select distinct(DnJaarWeek) from  tbweken where DnJaarWeek is not null and
DnJaarWeek <= 200201
  Got 105 tuples from server
  delete from TbDeclaraties where DnJaarWeek in ( 200001, 200002, 200003, 200004,
200005, 200006, 200007, 200008, 200009, 200010 )
  Command executed sucessfully
  0 row(s) affected
  ...
  ...
  ...
  Totaal aantal regels verwijderd: 118567
```

9 Implementation

9.1 doel

Het trainen van de gebruikers en het systeem inclusief alle documentatie installeren

9.2 werkwijze

Aangezien slechts een deel van de Systeemmodule is afgekomen wordt het systeem nog niet uitgeleverd. De student heeft ervoor gezorgd dat alle documentatie en source code op het netwerk geplaatst is en heeft een presentatie van zijn programma gegeven. Het programma werkte correct tijdens de presentatie (wat eerder uitzondering dan regel schijnt te zijn) maar de interface leverde wel kritiek op. De student is het eens met de kritiek maar blijft erbij dat de gekozen interface de minst slechte is.

9.3 User Documentation

Dit document heeft de student niet gemaakt omdat het tijdens zijn vertrek bij het bedrijf zeer hectisch was in verband met update 2.4.0 van Big Ben. De student zat in tijdnood en de collega's die zich normaal bezighouden met gebruikershandleidingen waren druk bezet en het leek de student beter om zich te concentreren op het vinden van bugs dan documentatie te schrijven die niet beoordeeld kon worden.

9.4 Increment Review Document

In dit document wordt een oordeel over het systeem geveld. Dit oordeel vindt plaats door alle requirements onder de loep te nemen en te bepalen in welke mate het systeem aan de requirements voldoet. Naar de mening van de student voldoet het systeem volledig aan de requirements, tenminste het deel dat afgekomen is.

De beslissing is hier echter zeer eenvoudig aangezien het niet gelukt is om alle functionaliteiten te realiseren. Alleen de functionaliteit 'Verwijderen van gegevens t/m een bepaalde datum' is gerealiseerd. Dit was de functionaliteit die de opdrachtgever de hoogste prioriteit had meegegeven. De opdrachtgever is derhalve tevreden over de resultaten.
Voor het vervolgtraject acht de student het nodig om de Design and Build Iteration opnieuw uit te voeren voor de overige gewenste functionaliteiten.

10 Evaluatie

10.1 proces

10.1.1 Feasibility Study

De student heeft erg lang over deze fase gedaan. Dit kwam door de onbekendheid met Big Ben, DbExpress en Oracle. Het inwerken heeft veel tijd gekost maar de student is hier achteraf niet rouwig om. Wel zou de student in het vervolg nog sneller komen met een Fast Prototype, omdat dit het startsein voor de feedback bleek. De student heeft zichzelf in deze fase geen deadlines gesteld omdat het niet op voorhand te zeggen was hoe lang een orientatie zou duren. De student is blij met dit besluit.

10.1.2 Business Study

Deze fase heeft ook lang geduurd en is niet naar tevredenheid van de student verlopen. Omdat de student een goed klassediagram als een hoeksteen voor het succes van het project zag heeft hij te lang gedraald met het maken hiervan. De student heeft teveel mogelijke klassediagrammen geprobeerd en veel tijd gestoken in het onderzoek naar Design Patterns. Omdat dit laatste onderwerp door de student als zeer moeilijk en abstract wordt ervaren vlotte het niet zoals hij wilde. Het lezen over Design Patterns heeft de student echter wel de vonk van inspiratie gegeven die hij nodig had.

De student is niet zo gecharmeerd meer van DSDM. De flexibiliteit van de methode sluit naadloos aan bij de praktijk binnen REM Automatisering, maar de student mist de preciese eisen die SDM stelt aan de documentatie. Bij DSDM is teveel voor invulling van de gebruiker opengelaten en de methode is zeer commercieel. Het verkrijgen van meer documentatie zou de student 135 euro kosten. Daar de student dit veel geld vind en hij ontevreden is over zijn huidige DSDM boek (de schrijfster van het boek biedt de extra documentatie aan) besluit de student de aanschaf als te riskant en ziet ervan af.

Ook in deze fase heeft de student geen deadlines gesteld. Hier is hij achteraf niet tevreden over.

10.1.3 Functional Model Iteration

In deze fase heeft de student afwisselend geprogrammeerd en gedocumenteerd. De student heeft per gekozen tabel vastgelegd wat de gevolgen moesten zijn als de er uit de tabel verwijderd wordt en ervoor gekozen om de feedback dynamisch op te bouwen. Verder is de student geconfronteerd met enkele eigenaardigheden van Delphi die naar verhouding zeer veel tijd hebben gekost. De student is verder zeer blij met de keuze om eerst het verwijderen van de tabel Weken te regelen en daarna pas de code generiek te maken. De student denkt momenteel nog dat hij meerdere functionaliteiten af zal krijgen.

10.1.4 Design and Build Iteration

In deze fase worden de sequence diagrammen gemaakt voor de belangrijkste functionaliteiten. Ook maakt de student een Use Case Diagram omdat het aantal Use Cases nogal gegroeid is. De student komt minder eigenaardigheden van Delphi tegen en het coderen verloopt dan ook vlot. Een grote fout(?) die de student maakt in deze fase is dat hij de gevaren van de kracht van DSDM niet onderkent. De student is zo gecharmeerd van de flexibiliteit van de methode en de mogelijkheid om informeel nieuwe wensen te verwerken dat hij verzuimt om een wensenstop in te stellen. Voor elke gerealiseerde functionaliteit komt er vrijwel onmiddelijk een nieuwe wens in de plaats.

Dit houdt uiteindelijk in dat de student aan het eind van het traject krap in de tijd komt te zitten en afhankelijk is van het ontbreken van groten bugs. De student heeft wat dit betreft echter geluk gehad. Er zaten slechts een aantal kleine bugs in het programma, de meeste na het uitvoeren van 'onlogische acties' zoals na het misklikken bij het selecteren van een tabel.

10.1.5 Implementation

De student is te lang bezig geweest aan de Design en Build Iteration en is in tijdnood gekomen. Omdat deze fase voor dit project eigenlijk grotendeels niet van toepassing uit kan het echter geen kwaad. De student vindt het wel jammer dat er geen tijd meer over was voor het maken van de gebruikershandleiding. Hij vraagt zich af of het niet beter zou zijn om hier in het vervolg in een eerdere fase mee te beginnen.

10.2 product

10.2.1 Opdrachtsomschrijving

Deze bleek achteraf te optimistisch te zijn opgesteld omdat het grootste deel van de wensen en eisen pas in een laat stadium naar voren kwam.

10.2.2 Feasibility Report

De student is tevreden over het Feasibility Report, ook met de conclusie dat de opdracht haalbaar zou zijn. De opdracht is uiteindelijk niet te ingewikkeld maar wel te groot gebleken. Tijdens het maken van dit rapport was er echter nog te weinig zicht op de totale scope van de opdracht, al had het gebrek aan herbruikbare code de student moeten waarschuwen.

10.2.3 Fast Prototype

De student is zeer blij dat hij dit product gemaakt heeft en de basisstructuur bleek ook een goede hoeksteen te zijn waarop verder gebouwd kon worden. Al met al is de student erg tevreden met dit product. Er is wel een minpuntje: het product is in een te laat stadium gemaakt. Pas na het maken van dit product kwamen er veel extra wensen en eisen binnen.

10.2.4 Outline Plan for Development

Dit product is zeer kort en globaal. Het zou misschien beter geweest zijn om tijdens de orientatieperiode ook de connectie met de database te leggen. Dit maakt het mogelijk om gedetailleerder te plannen.

10.2.5 Business Area Definition

Dit is van alle producten het minst geslaagd volgens de student. De student heeft teveel vreemde klassediagrammen geprobeerd voordat hij uiteindelijk een keuze heeft gemaakt. De meeste van deze klassediagrammen heeft hij echter weggegooid. Een exemplaar heeft de student echter over het hoofd gezien en deze is opgenomen in de bijlagen. De student is wel tevreden over de keuze om alle frames via het FrameOutlookBar aan te roepen.

10.2.6 System Architecture Definition

Dit product vormde de basis voor het leggen van een koppeling met de database en is alleen daarom waardevol geweest.

10.2.7 Development Plan

Het was pech dat er geen goed Unit Frame Testwork voor Delphi beschikbaar was want dit had veel tijd kunnen schelen. De student vindt het een veilig idee dat er al in zo'n vroeg stadium is vastgelegd hoe er getest gaat worden.

10.2.8 Implementation Plan

De student vindt een gedetailleerd Implementation Plan in zijn situatie onnodig.

10.2.9 Tested System

De student is zeer tevreden over de snelheid van het programma, over de losse koppelingen tussen de verschillende klassen en over het gebruiksgemak van de applicatie. De student heeft echter grote spijt van het coderen van elke te tonen string als een constante om zo de vertaling zo eenvoudig mogelijk te maken. De student meent dat de eis dat het programma rekening houdt met een vertaling voor een niet gebruikte feature heeft gezorgd en vindt het zonde van de tijd geweest. Ook is de code iets minder leesbaar geworden door het gebruik van de constanten. De student is blij met de robuustheid van het programma en de generiekheid van de Execute methode die de bewerkingen op de database uitvoert.

10.2.10 User Documentation

Het besluit om dit niet te maken hangt samen met de tijdnood van de student aan het eind en de mening dat het nuttiger was om bugs uit de code te halen en extra commentaar toe te voegen. Aangezien de module nog verder ontwikkeld moet worden voordat deze uitgeleverd wordt staat de student achter deze keuze.

10.2.11 Increment Review Document

De student is tevreden over dit document maar ergens wel teleurgesteld dat er slechts een functionaliteit af is gekomen terwijl het programma zo gemaakt is dat het eenvoudig uitbreidbaar moet zijn. Met de opdrachtgever is gesproken over het verwijderen van ongebruikte records en de student had hier zo mee aan de slag kunnen gaan.

De overige functionaliteiten dienen echter nog bijna volledig ingevuld te worden. Het verwijderen per tabel is alleen bij de titel bekend.Van de reorganiseeropties is het slechts globaal duidelijk wat er moet gebeuren. Technisch gezien zou het eenvoudig moeten zijn om deze toe te voegen, maar er dient heel goed gekeken te worden naar de omgang met historische gegevens.

Literatuurlijst / Bronvermeldingen

Stapleton, Jennifer (2003) *DSDM: Business Focused Development,* Addison Wesley
ISBN 0-321-11224-5.

Oestereich, Bern (1999) *Developing Software with UML,* Addison Wesley
ISBN 0-201-36826-5.

Gamma, Erich et Al. (1995) *Design Patterns: Elements of Reusable Object-Oriented Software,*
Addison Wesley, ISBN 0-201-63361-2.1

Beck, Kent (2000) *Extreme Programming Explained: Embrace Change,* Addison Wesley
ISBN 0-201-61641-6.

Bijlage A: Korte beschrijving van DSDM

DSDM staat voor Dynamic Systems Development Method. De principes achter DSDM zijn:

- ontwikkelen is teamwerk
- hoge kwaliteit houdt in geschiktheid voor het doel en technische robuustheid
- ontwikkeling kan incrementeel zijn – niet alles hoeft in een keer geleverd te worden, en een klein deel in een vroeg stadium leveren is vaak beter dan alles later leveren
- concentreer je op de meest waardevolle 'features'

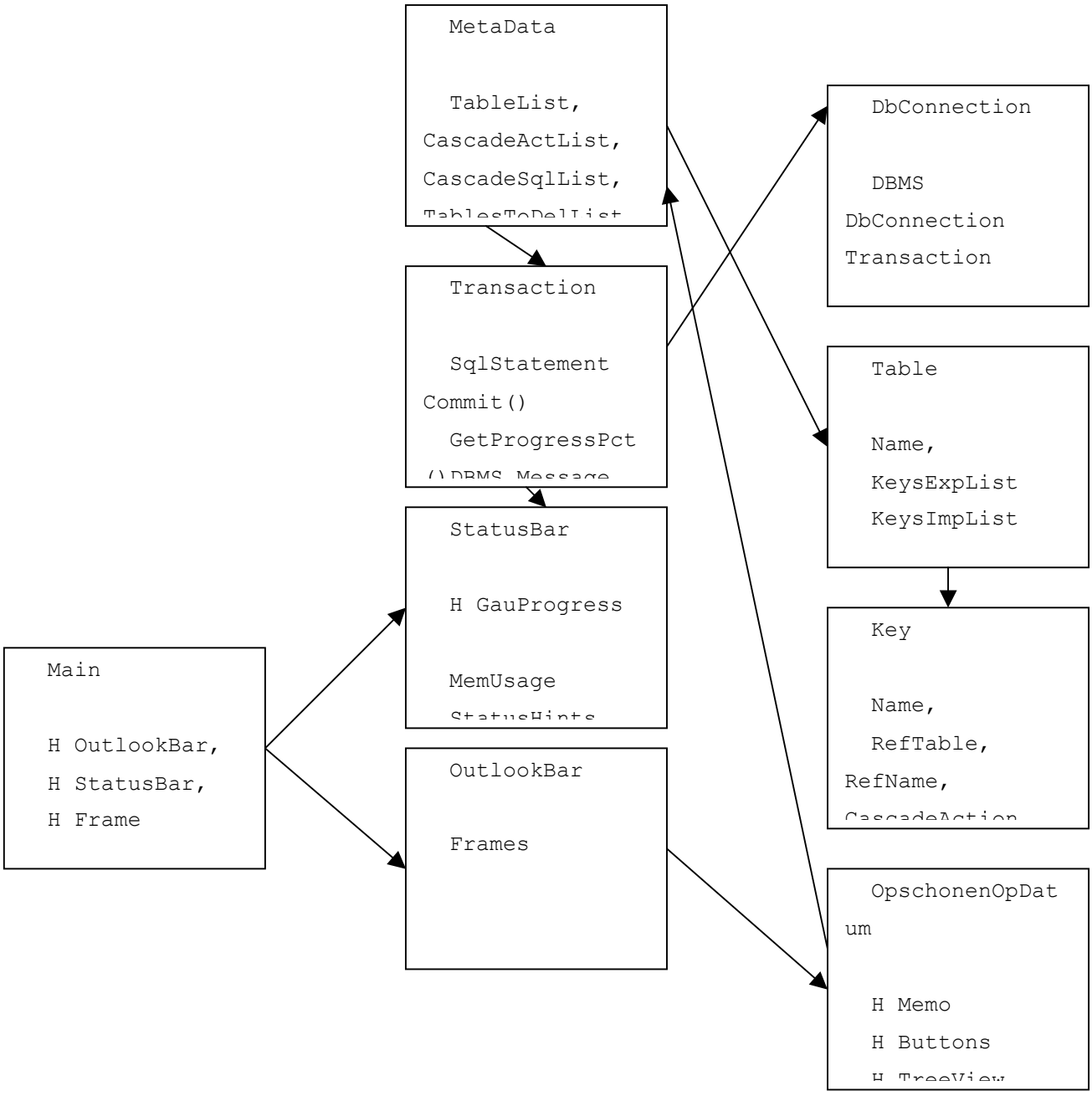
DSDM defineert de processen die gevolgd moeten worden voor snelle software-ontwikkeling en de producten die het resultaat zijn van deze processen. Het detailniveau van deze beschrijvingen is zeer hoog. DSDM schrijft geen te gebruiken technieken voor. De belangrijkste redenen waarom DSDM sneller werkt dan de traditionele waterval methodes zijn

1) alleen de noodzakelijke features worden ontwikkeld
2) problemen kunnen eerder herkend worden

Voor meer informatie, zie http://www.dsdm.org/

Bijlage B: Een vroeg klassediagram

Dit klassediagram is gemaakt in het begin van de Business Area Definition en is uiteindelijk verworpen omdat de metadata niet dynamisch ingelezen mocht worden. Dit was nu juist de grote charme van dit model.



Bijlage C: Consequenties van de verschillende opties bij het opschonen op datum

legenda

Tabel in het vet geschreven: Zichtbaar voor de gebruiker

Budget Versie(Begroting): Tabel TbBudgetVersie staat onder de naam 'Begroting' op het scherm

X = cascade no action; in dit voorbeeld: weiger Activiteiten Schema te verwijderen als er Verantwoordingen onder hangen

D = cascade delete; verwijder eerst alle Aanstellings Activiteiten als je het Activiteiten Schema opschoont.

S = cascade set null; maak eerst verwijzing(en) naar Organisatie 'null' in de tabel Organisatie

---

ActiviteitenSchema (Activiteitenschema)

X Verantwoording

D Aanstellingsactiviteiten

D Activiteit Attributen

X Budget Detail

D Budget Items

D Code Filter

D Organisatie Activiteiten

Organisatie (Afdelingen)

X Aanstelling

D Budget Versie

D Budget Detail

S Organisatie

D Autorisatie Organisatie

D Koppeling

Budget Periode

D Budget Versie (Verzamelbegroting)

D Budget Detail

D Budget Versie (Verlofbudget)

D Budget Detail

D Budget Versie (Projectbegroting)

D Budget Detail

D Budget Versie (Detailbegroting)

D Budget Detail

Code Tabellen (Dynamische Codetabellen)

D Code (Code Tabel 1, naam wordt dynamisch ingelezen)

S Verantwoording

X Budget Items

D Code Filter

D Code (Code Tabel 2, naam wordt dynamisch ingelezen)

S Verantwoording

X Budget Items

D Code Filter

D Code (Code Tabel 3, naam wordt dynamisch ingelezen)

S Verantwoording

X Budget Items

D Code Filter

Geschreven Uren (Verantwoording)

D Weken

D Verantwoording

D Declaraties

D Declaraties

D Aanwezig Detail (Aanwezigheid)

Medewerker (Medewerkers)

D Aanstelling (Aanstellingen)

D Budget Versie

D Budget Detail

D Weken

D Verantwoording

D Declaraties

D Aanstellings Activiteiten

D Autorisatie Aanstelling

S Code Filter

S Declaratie Filter

D Declaraties

D Jaar

D AlaCarte

D Koppeling

D Verlof Aanvraag (Verlofaanvragen)

D Werkplan

S Budget

D Aanwezig Detail

D Autorisatie Aanstelling

D Autorisatie Medewerker

D Autorisatie Organisatie

D Autorisatie Project

D Rapport Template

Tarieven