Afstudeerverslag

Ontwikkelen en realiseren van een conversietool tussen de Knaakbaak en Calkey.

Auteur :	J. Beukers 20008203
Examinatoren:	S. Mimoun
	P. Breukel
Plaats, datum	Woerden, 10 juni 2005
Opdrachtgever:	ALFA Development.

Referaat

Beukers, J., Afstudeerverslag, ontwikkelen en realiseren van een conversiesysteem tussen de begrotingspakketten de Knaakbaak en Calkey, Woerden, ALFA Development, 2005.

Dit eindverslag beschrijft de afstudeerstage van J. Beukers bij ALFA Development te Woerden. De afstudeeropdracht is uitgevoerd als afsluiting van de studie Informatica en Informatiekunde (afstudeerrichting Management en Beheer van InformatieVoorziening) aan de Haagse Hogeschool.

Dit eindverslag behandelt de volgende onderwerpen:

- De organisatie
- De afstudeeropdracht
- De uitgevoerde werkzaamheden
- Evaluatie van het afstuderen

Voorwoord

Als afsluiting van mijn opleiding informatica en informatiekunde heb ik een afstudeeropdracht bij ALFA Development te Woerden uitgevoerd. De afstudeerperiode begon 7 februari 2005 en eindigde 10 juni 2005. Naar aanleiding van de uitgevoerde opdracht is dit verslag opgesteld.

Het verslag is bestemd voor de examinatoren en de gecommitteerden die zicht op de omvang, diepgang en uitwerking van de opdracht dienen te verkrijgen. Ook de wijze waarop de producten tot stand zijn gekomen (het gevolgde proces) en de uiteindelijk opgeleverde producten bestemd voor het bedrijf komen in dit verslag aan bod.

Verschillende personen wil ik bij deze bedanken voor de geboden hulp tijdens mijn afstudeerperiode. Mijn dank gaat uit naar de bedrijfsmentor R. Woolderink en manager R. Pleijsier. Zij hebben mij begeleid tijdens het uitvoeren van mijn opdracht. De begeleiding vanuit de Haagse Hogeschool is uitgevoerd door de heer Breukel en mevrouw Mimoun. Ook naar deze personen gaat mijn dank uit.

Jan Beukers

Woerden, Juni 2005.

Inhoudsopgave

1. Inleiding

Dit document is het afstudeerverslag dat betrekking heeft op mijn afstudeertraject. Ik ben Jan Beukers, een student aan de afdeling Informatica en Informatiekunde van de Haagse Hogeschool. Het doel van dit afstudeerverslag is om de lezer een beeld verschaffen van de door mij uitgevoerde activiteiten, de gemaakte keuzes, de gehanteerde methodes en werkwijzen, kortom: welke activiteiten, hoe de activiteiten zijn uitgevoerd, waarom ze zo zijn uitgevoerd en wat het resultaat was de uitgevoerde activiteiten. Dit is gedaan om het mogelijk te maken dat het HBO-niveau kan worden vastgesteld en getoetst. Daarnaast wordt in dit document ook aandacht besteed aan de geboekte leerwinst.

Van 7 februari 2005 tot en met 10 juni 2005 heb ik gewerkt aan mijn afstudeeropdracht bij ALFA Development, een software ontwikkelaar van bouwkundige CAD software. ALFA Development is gevestigd te Woerden. De afstudeeropdracht luidde als volgt: Het ontwikkelen en realiseren van een conversiesysteem tussen de begrotingspakketten de Knaakbaak en Calkey. Hierbij is de Knaakbaak de bron en Calkey het doel.

Dit document is gestructureerd naar de gevolgde stappen uit delen van SDM (System Development Methodology). Uit deze methode zijn de belangrijkste elementen gehaald en gebruikt. Voor deze structurering van dit document door middel van SDM is gekozen zodat er een chronologisch beeld wordt weergegeven van de uitgevoerde processen.

In de eerste twee hoofdstukken wordt de omgeving van de opdracht weergegeven namelijk ALFA Development en het Digitale Huis. Vervolgens zal er in hoofdstuk drie een omschrijving van de opdracht gegeven worden waarna in hoofdstuk vier de aanpak van de opdracht wordt beschreven.

In hoofdstuk vijf wordt de eerste fase van het doorlopen traject behandeld, hetwelk de definitiestudie is. In het hoofdstuk na de definitiestudie wordt er overgestapt op het ontwerp. Hoewel SDM het ontwerp heeft opgesplitst in twee fasen, namelijk het basisontwerp en het detailontwerp, heb ik er voor gekozen om deze twee fasen samen te voegen tot één. Dit heb ik gedaan zodat de delen van het ontwerp niet onnodig klein zouden worden zodat de samenhang zou vervallen. In het zevende hoofdstuk wordt er beschreven hoe de realisatie van de conversietool is verlopen. Bij elk van deze hoofdstukken wil ik doormiddel van voorbeelden uit de verschillende fasen toelichten hoe ik de fase heb uitgevoerd, waarom ik het traject zo heb doorlopen en welke problemen ik ben tegengekomen.

In het laatste hoofdstuk wordt de evaluatie behandeld. Hier zal ik een schets van mijn persoonlijke evaluatie van de totale procesgang tijdens het afstuderen weergeven. Ook de opgeleverde producten zullen worden geëvalueerd.

In dit rapport zal regelmatig worden gerefereerd naar de (deel)producten opgenomen in de externe bijlagen. Daar waar delen uit de (deel)producten in dit rapport worden gekopieerd wordt aangegeven uit welk (deel)product het deel afkomstig is.

1. Beschrijving ALFA Development

In dit gedeelte wordt de historie van ALFA Development behandeld, vervolgens worden de structuur van de organisatie weergegeven met daarin de plek van de afstudeerder.

1.1. Historie

ALFA Development B.V. is een software ontwikkelaar van bouwkundige CAD Software en is voortgekomen uit ARCADE, bureau voor CAD toepassingen. Met 15 jaar ervaring op het gebied van CAD ontwikkeling is ALFA de leidende bouwkundige CAD applicatie voor AutoCAD, met ca. 2000 gebruikers.
Sinds 1 januari 2003 zijn de ontwikkel en de verkoopactiviteiten van de ALFA software samengevoegd in ALFA Development, waarbij ook het geautoriseerd dealerschap van AutoCAD is verworven.

1.2. Structuur

De ontwikkeling van de software vind plaats in Woerden en de verkoop activiteiten worden verzorgd door het Digitale Huis en is gevestigd in ’s-Hertogenbosch. Het verdient, denk ik, geen uitleg dat ik werkzaam ben in Woerden, daar waar de software wordt ontwikkeld. De afdeling bestaat uit 5 personen, waarbij de heer Pleijsier de manager is.

1.3. Applicaties

Zoals vermeld beheerd ALFA Development de CAD applicatie ALFA. Daarnaast is ALFA Development van plan om het bouwkundige begrotingsprogramma ‘Calkey’ op de markt te brengen. Op dit moment bevind Calkey zich in de laatste fase van ontwikkeling.

Het is mogelijk om Calkey te koppelen aan ALFA zodat als er getekend wordt in ALFA de begroting in Calkey automatisch wordt opgebouwd. Om bedragen te koppelen aan de getekende bouwelementen dient een kostenbestand van de Knaakbaak gebruikt te worden. De Knaakbaak is een ander bouwkundig begrotingsprogramma, in het beheer van het Digitale Huis.

2. Beschrijving het Digitale huis

In dit gedeelte wordt beschreven wat het Digitale Huis is. Vervolgens wordt ook hier weer ingegaan op de structuur van het Digitale Huis.

2.1. Wat is het Digitale Huis

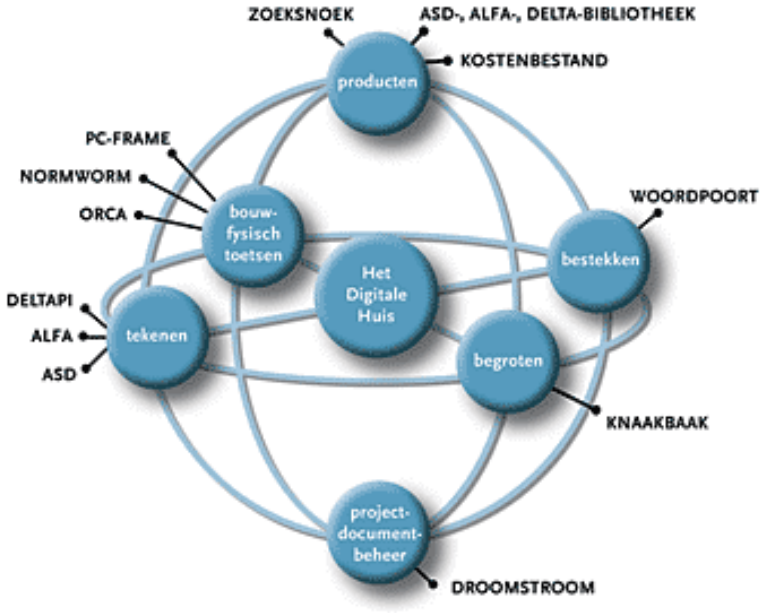
Het digitale huis is een samenwerkingsverband van ontwikkelaars van bouwkundige software. Het doel is de ontwikkeling van een centraal digitaal gebouwmodel. Dit gebouwmodel maakt communicatie tussen de Digitale Huis software mogelijk. Hierdoor kunnen ontwerpers, tekenaars, bestekschrijvers, kostendeskundigen en bouwfysisch toetsers met hun eigen software samenwerken aan één project en onderling gegevens uitwisselen. Door gebruik van een gezamenlijke projectdatabase delen projectmedewerkers altijd dezelfde, meest actuele informatie met elkaar. Optimale communicatie, consistente informatie, minder fouten en een enorme snelheidswinst zijn het gevolg.

Het initiatief van het Digitale Huis wordt gesteund door het ministerie van Economische Zaken (Senter) en de BNA (Bond van de Nederlandse Architecten). Ook wordt nauw samengewerkt met de TU Eindhoven en TNO. Het Digitale Huis is door Senter aangemerkt als ICT-doorbraakproject en is verkozen tot Europees Voorbeeldproject op het gebied van technologische innovatie.

2.2. De structuur van het Digitale Huis

De structuur van het digitale huis is dusdanig opgezet dat alle bewerkingen op een digitaal gebouwmodel mogelijk zijn. Zo wordt er begonnen bij het architectenbureau. Dit is namelijk de eerste partij die in aanmerking komt om het gebouwmodel te vullen. Hier ontstaat immers het ontwerp van een digitaal gebouw en wordt het technisch uitgewerkt. In eerste instantie concentreert het Digitale Huis zich dan ook op de wereld van het architectenbureau en dus op de soorten informatie en applicaties die daar gebruikt worden. Als op het architectenbureau eenmaal een samenhangend model beschikbaar is, kan de architect op allerlei manieren informatie uitwisselen met de andere partijen die zich met een bouwproject bezighouden. Voorbeelden hiervan zijn opdrachtgevers, aannemers, de overheid, constructeurs, installatieadviseurs, enzovoort.

Het Digitale Huis heeft in 2001 een vliegende start gemaakt en wordt door steeds meer partijen gesteund. Naast ALFA Development, DeltaPi Systems en de Twee Snoeken Automatisering dewelke CADapplicaties op de markt brengen dragen ook Mile 17 en de Vabi bij aan de ontwikkeling van Digitale Huis software. Mile 17 houdt zich bezig met de bouwkosten en Vabi is een overkoepelend orgaan van de installatie branche. In het onderstaande plaatje is te zien welke vakgebieden (dit zijn de bollen) er op het Digitale Huis mogelijk zijn, en welke applicaties welk gedeelte voor hun rekening nemen.



afbeelding 1. structuur van het Digitale Huis

3. Omschrijving opdracht

3.1. Probleemstelling

De huidige rekenkernel die wordt gebruikt door de Knaakbaak is te traag.
De datastructuur en werkwijze van beide programma's zijn duidelijk verschillend. Binnen Calkey wordt gebruik gemaakt van een referentie model waar in kosten voor een bepaald bouwdeel, zoals een bepaald type wand, éénmalig vastgelegd in een definitie van het bouwdeel. Bij elke instantie van dat bouwdeel in de calculatie wordt gerefereerd naar die definitie. Binnen de Knaakbaak kunnen bouwdelen met dezelfde codering en omschrijving verschillend afgeprijsd worden. Bouwdelen kunnen in de begroting gekopieerd worden zonder controle op een unieke definitie van de prijsopbouw. Er kan dus niet klakkeloos overgeschakeld worden naar de nieuwe en snellere rekenkernel Calkey.

3.2. Doelstelling van de opdracht

Het is de bedoeling het Knaakbaak basis bestand met bouwkosten definities te converteren naar Calkey bestandsstructuur. Bij een conversie zal een controle gedaan moeten worden op verschillende kostenonderbouwingen van bouwdelen met dezelfde codering en omschrijving. Hierbij moet de gehele boomstructuur van de calculatie doorlopen worden. Bij discrepanties moet een signalering volgen in de vorm van een rapportage.

3.3. Uitgangssituatie

De Knaakbaak kostenbestanden bestaan uit een MS Access database waar in de verschillende tabellen en sommige onderlinge relaties zijn vastgelegd. Calkey is een Delphi applicatie waarbij de records, waar de kosten in worden vastgelegd, in een eigen bestandsstructuur worden opgeslagen.

4. Aanpak opdracht

In dit gedeelte wordt de aanpak van de opdracht beschreven. Hierbij wordt ingegaan op hoe de opdracht gaat worden uitgevoerd. Het plan van aanpak is opgesteld om voor mijzelf inzicht te verschaffen in het uit te voeren project en om een leidraad te creëren voor het uitvoeren van het project. Ook kunnen de begeleiders en het management door middel van dit document geïnformeerd worden over de aanpak van het afstudeerproject. Het inzicht is verschaft door na te denken over de te gebruiken middelen. De leidraad is gecreëerd door het opstellen van een planning. Zowel de gebruikte middelen als de planning worden hieronder weergegeven.

4.1. De gebruikte middelen

Tijdens het project zijn er van een aantal middelen gebruik gemaakt om de benodigde activiteiten te realiseren. De keuze om dit op te nemen in het afstudeerverslag is om de lezer inzicht te verschaffen in de context waarbinnen dit project heeft plaatsgevonden.
De middelen die in dit hoofdstuk behandeld gaan worden staan hieronder opgesomd:

- PC
- Printer
- Server voor de Back-up
- Ontwikkelomgeving: Delphi (5.0)
- Ontwikkelmethodiek: SDM (System Development Methodology)
- Ontwikkeltechniek: UML (Unified Modelling Language)

De keuze van de ontwikkelomgeving, ontwikkelmethodiek en de ontwikkeltechniek wil ik graag even toelichten omdat er hier te kiezen is uit velen.

4.1.1. Ontwikkelomgeving

Voor ontwikkelomgeving is er gekozen voor Delphi. Hiervoor is gekozen omdat het nieuwe begrotingsprogramma van ALFA Development namelijk Calkey ook is geschreven in Delphi. Er liggen plannen voor in de toekomst om de conversietool te gaan gebruiken binnen het programma Calkey. Aangezien het niet handig de sourcecode te gaan vertalen als de conversietool in Calkey geïntegreerd wordt is er voor gekozen om de conversietool ook in Delphi te ontwikkelen. Een bijkomend voordeel is dat ik tijdens mijn stage ook al met Delphi heb gewerkt. Daar had ik geleerd dat de Access database tabellen waarin de gegevens van de Knaakbaak stonden te openen waren. Hierdoor was nog niet het probleem van het uitpluizen van de structuur van de Knaakbaak achter de rug maar de gegevens waren hierdoor wel toegankelijk geworden. Delphi is een objectgeoriënteerd taal gebaseerd op Pascal.

4.1.2. Ontwikkelmethodiek

Om tot een structurele aanpak van mijn afstudeerproject te komen heb ik een ontwikkelmethode gebruikt. Vanuit de opleiding ben ik opgevoed met SDM. De methodiek werkt volgens het waterval model. Dit is ook bruikbaar in het mijn afstudeerproject. Het is namelijk mogelijk om verschillende fases te onderscheiden. Aangezien het een relatief klein project is worden sommige fasen overgeslagen of samengevoegd. Ook het informatieplan was al bekend. Hierin worden normaal gesproken de ontwikkelingsplannen beschreven. Deze bestonden reeds, zo ook de plannen om de conversietool later aan Calkey applicatie toe te voegen. De huidige situatie zijn de Knaakbaak en Calkey. Deze begrotingsprogramma's zullen dan ook in dit gedeelte worden onderzocht en gedocumenteerd. Omdat het een klein project is worden de fasen basisontwerp en detailontwerp,zoals eerder vermeld,

samengevoegd tot één ontwerp. Indien dit niet zou gebeuren zouden er onnodige details of te kleine documenten ontstaan waardoor de samenhang zou vervallen. Ook zal het invoeren van de conversietool zal komen te vervallen omdat deze in een later stadium wordt geïmplementeerd in Calkey. Hierdoor is bijvoorbeeld het opleiden van gebruikers niet nodig. Door de watervalmethode van SDM is het mogelijk om de verschillende fases ook als mijlpaalproducten op te nemen in de planning. Elke fase kan formeel worden afgerond, waarna kan overgestapt op de volgende fase. In dit project worden dus de volgende fases onderscheiden:

- Definitiestudie
- Ontwerp (Basis en detailontwerp)
- Realisatie
- Invoering

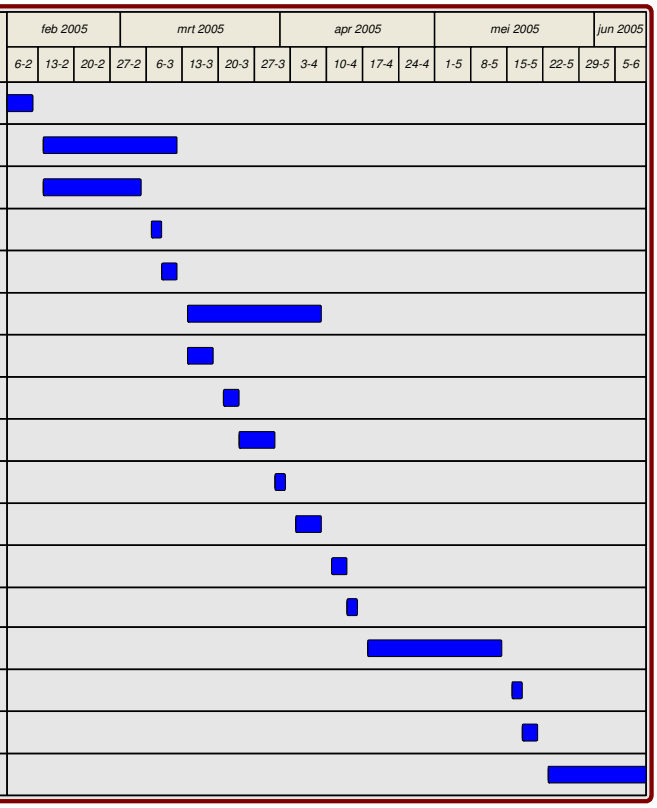
4.1.3. Ontwikkeltechniek

Als modelleringtechniek is er gekozen voor UML (Unified Modelling Language). UML biedt een aanpak voor het ontwerpen van een systeem dat berust op een objectgeoriënteerde modelleringwijze. Deze modelleringwijze stelt dat de toestand van een systeem te omsluiten is in objecten die alleen benaderd kunnen worden door middel van methodes die op deze objecten zijn gedefinieerd. De samenwerkende objecten die een afspiegeling van de entiteiten uit het domein zijn vormen de bouwstenen voor het systeem. Aangezien Calkey ook werkt op basis van objecten en omdat de structuur van Calkey het resultaat van de conversie dient te zijn is er gekozen voor deze modelleringwijze. Ook is UML behandeld op school waardoor ik de voorkeur aan deze taal gaf.

4.2. Planning

Aan de hand van de te behandelen fases kan er een planning opgesteld worden door per fase te benoemen welke werkzaamheden er uitgevoerd dienen te worden en vervolgens in te schatten hoeveel tijd voor die werkzaamheden benodigd is. Hierbij is er voor gezorgd dat er voldoende uitlooptijd was. Het analyseren van de huidige situatie is bijvoorbeeld niet goed in te schatten. Afhankelijk van de complexiteit van de beide structuren zal dit sneller of langzamer verlopen. In de grandchart hieronder afgebeeld staat de planning van het project weergegeven:

ID	Taak	Start	Eind	Duur
1	Opstellen plan van aanpak	7-2-2005	11-2-2005	5d
2	Uitvoeren definitiestudie	14-2-2005	11-3-2005	20d
3	Onderzoeken huidige situatie namelijk de knaakbaak en de calkey	14-2-2005	4-3-2005	15d
4	Opstellen eisen en wensen	7-3-2005	8-3-2005	2d
5	Opstellen systeemconcepten en gevolgen	9-3-2005	11-3-2005	3d
6	Ontwerp conversietool	14-3-2005	8-4-2005	20d
7	Detailleer systeemeisen en informatiebehoeften	14-3-2005	18-3-2005	5d
8	Ontwikkel mens/machine raakvlakken	21-3-2005	23-3-2005	3d
9	Specificeer procedures en formulieren	24-3-2005	30-3-2005	5d
10	Specificeer beeldscherm en lijst indelingen	31-3-2005	1-4-2005	2d
11	Specificeer programmatuur	4-4-2005	8-4-2005	5d
12	Opstellen testplan	11-4-2005	13-4-2005	3d
13	Detailleer plan voor realisatie en invoering	14-4-2005	15-4-2005	2d
14	Realisatie conversietool	18-4-2005	13-5-2005	20d
15	Testen conversietool	16-5-2005	17-5-2005	2d
16	Schrijven korte handleiding	18-5-2005	20-5-2005	3d
17	Schrijven afstudeerverslag	23-5-2005	10-6-2005	15d



afbeelding 2: grandchart uit het plan van aanpak

5. Bouwkostencalculatie

5.1. Inleiding

In dit gedeelte zal in het kort worden beschreven wat een bouwkostencalculatie is. Dit zal gedaan worden omdat beide programma's (de Knaakbaak en de Calkey) een bouwkostencalculatie programma zijn. Om de opslagstructuur van deze programma's beter te kunnen begrijpen wordt er eerst gekeken worden naar wat ze op moeten slaan, daarna wordt er gekeken naar hoe beide programma's deze gegevens opslaan.

5.2. Directe kosten

De kosten die rechtstreeks verbonden zijn aan te bouwen onderdelen van het gebouw vormen de directe kosten. Ze zijn direct aan de productie gerelateerd. De kosten zijn ondergebracht in begrotingsposten, zoals 'storten beton B25', 'metselen halfsteens waalformaat', enz. Aan een begrotingsposten kunnen middelen gekoppeld worden, die nodig zijn voor het realiseren van de post. Deze middelen hebben een prijs per eenheid en een hoeveelheid. Bijvoorbeeld: voor het maken van een bekisting is een aantal uur een timmerman nodig en een aantal m2 bekistinghout.

Een middel is altijd één van de volgende kostensoorten:

- Loon
- Materiaal
- Materieel
- Onderaanneming
- Overige

Alle directe kosten bestaan uit deze kostensoorten

De structuur van een begroting en de ordening van de gegevens kan in diverse situaties verschillen. Zo wordt er onderscheid gemaakt tussen traditionele begroting en een gestructureerde begroting.

5.2.1. Traditionele begroting.

In een traditionele begroting zijn alle begrotingsposten in een lange lijst opgesomd en afgeprijsd. Er is een indeling in hoofdstukken en paragrafen, maar verder is de structuur 'plat'. Een traditionele begroting wordt vaak gebruikt voor voorcalculatie van niet al te grote projecten. De kosten van iedere begrotingspost worden vaak direct bij de post ingevoerd zonder dat ze nader onderbouwd worden.

5.2.2. Gestructureerde begroting.

Een begroting kan heel veel informatie bevatten. Vooral wanneer het een gedetailleerde begroting betreft van een groot bouwwerk, is de hoeveelheid kostenregels zeer groot. Om de kostenstructuur inzichtelijk te maken wordt vaak gewerkt met verschillende detailleringniveaus. Er kan van grof naar fijn worden afgedaald in de begroting. In een begroting zouden de volgende detailleringniveaus kunnen worden onderscheiden:

- Project
- Gebouwdeel
- Bouwdeel
- Begrotingspost
- Middel

In een gestructureerde begroting is er sprake van decompositie; een project bestaat uit gebouwdelen, een gebouwdeel bestaat uit bouwdelen, een bouwdeel bestaat uit begrotingsposten en een begrotingspost bestaat uit middelen. Middelen zij altijd van een bepaalde kostensoort: loon, materiaal, materieel, onderaanneming of overig. Iedere post op het hogere niveau bevat het totaal van de kosten in de onderbouwing op het onderliggende niveau per kostensoort.

5.3. Staartkosten

Op een bouwwerk drukken nog diverse andere kosten, die niet productiegebonden zijn. Zo zijn er diverse voorzieningen benodigd om het bouwterrein in te richten en is er uitvoerend personeel nodig voor de organisatie en coördinatie (uitvoeringskosten). Daarnaast worden de kosten van onder andere het kantoor doorberekend over de bouwprojecten (algemene kosten). Andere opslagen zijn bijvoorbeeld winst en risico. Deze indirecte kosten worden in een apart blad afgedrukt: het staartblad. Meestal worden deze kosten door de aannemer als opslagpercentages over de directe kosten genomen, maar ook worden er wel eens vaste bedragen gehanteerd.

5.4. Stelposten en posten buiten opslagen

Stelposten zijn posten waarvan het nog niet bekend is wat de uiteindelijke kosten zijn, bijv. omdat de opdrachtgever van het project nog geen definitieve keuze heeft gemaakt voor een bepaald onderdeel. Het is gebruikelijk in de staart geen opslagpercentages te nemen over stelposten. Het kan ook voorkomen dat er begrotingsposten zijn, waarover om andere redenen geen opslagen genomen worden door de aannemer, bijvoorbeeld kosten door vergunning of ingenieurskosten. Deze kosten noemen we posten buiten opslagen.

6. Definitiestudie

In dit gedeelte wordt het uitvoeren van de definitiestudie beschreven. Per onderdeel worden de gemaakte keuzes, gebruikte technieken, aangetroffen problemen en resultaten beschreven. Bij het uitvoeren van de definitiestudie is er gekeken naar de huidige situatie, waaruit de eisen en wensen voor de te schrijven conversietool deels uit voortkomen. Tot slot zijn er systeemconcepten opgesteld waar vervolgens een keuze uit gemaakt kon worden.

6.1. Huidige situatie

In dit gedeelte wordt de huidige situatie beschreven. Het beschrijven (en analyseren) van de huidige situatie was nodig om de omvang en diepgang van de conversie in kaart te brengen. Er moet namelijk bekend zijn om een conversie uit te voeren of de gegevensstructuren van beide applicaties de Knaakbaak en Calkey respectievelijk op de juiste manier uit te lezen of in te voeren zijn. Hiervoor heb ik de structuur van de beide begrotingspakketten beschreven.

Omdat Calkey het resultaat van de conversietool diende te zijn is er voor gekozen om eerst Calkey te analyseren, vervolgens de structuur van de Knaakbaak zodat er gekeken kan worden of alle gegeven benodigd in de Calkey structuur ook aanwezig zijn in de Knaakbaak.

6.1.1. Calkey.

Allereerst was het zaak om te onderzoeken hoe Calkey de gegevens van de begroting ordende. Calkey is een applicatie die alle gegevens voor de begroting inleest in het werkgeheugen. Dit heeft als voordeel dat de gegevens snel toegankelijk zijn en mutaties snel zijn doorgevoerd. Het nadeel hiervan is dat de gegevens niet door meerdere gebruikers gedeeld kunnen worden en in het geval van uitval van het systeem de gegevens uit het werkgeheugen verloren zijn gegaan.

Bij het achterhalen van de werking van de structuur heb ik de sourcecode van Calkey gebruikt. Samen met de ontwikkelaar van Calkey hebben we in vogelvlucht de code doorgenomen. Hierdoor werd duidelijk hoe de structuur ten behoeve van het opbouwen van de begroting in elkaar stak.

Calkey is een nieuwe applicatie die door ALFA Development later dit jaar op de markt gebracht wordt. Het begrotingsprogramma is dus ook nog in ontwikkeling. Calkey is gebaseerd op een oude applicatie. Deze applicatie bouwde een begroting op aan de hand van kopregels en receptregels. Deze manier van opbouwen van een begroting wordt in het vervolg receptenmethodiek genoemd.

6.1.2. Receptenmethodiek

In dit gedeelte wordt in het kort beschreven wat de receptenmethodiek inhoudt. Calkey werkt met deze methode en om de structuur van deze applicaties te begrijpen is het zaak om te weten hoe deze methode in elkaar steekt.

De receptenmethodiek houdt in dat een bouwdeel weergegeven wordt door een begrotingsregel die, met hoeveelheden, onderbouwd kan zijn met andere begrotingsregels. De afzonderlijke kosten van deze onderbouwingsregels (receptregels genoemd) worden getotaliseerd tot de totale kosten van het bouwdeel (kopregel).
De verschillende begrotingsregels worden door middel van een unieke code geïdentificeerd. Dit onderbouwen van begrotingsregels kan herhaald plaatsvinden. Zo kan bijvoorbeeld een gevel onder andere bestaan uit een spouwmuur die op zijn beurt weer bestaat uit een binnenblad en buitenblad waarvan het binnenblad weer kan bestaan uit kalkzandsteen, mortel en dergelijke. De gebruiker kan zelf bepalen hoe ver hij met zo'n onderbouwing wil gaan. Vanaf een bepaald niveau van onderbouwing kunnen ook normprijzen gehanteerd worden in plaats van een meer gedetailleerde onderbouwing.

Deze receptenmethodiek kenmerkt zich door de volgende eigenschappen :

- De elementen zijn via een boomstructuur steeds verder onder te verdelen en te onderbouwen. Hierdoor is het mogelijk de kosteninformatie van een project, in niveaus steeds verder te onderbouwen als men in een volgend stadium van het ontwerp komt waar meer gedetailleerde informatie aanwezig is.
- De kosteninformatie uit een hoger niveau, die weer opgebouwd is uit informatie uit een lager niveau, is samengesteld aan de hand van de kosteninformatie op lager gelegen niveaus en kan dus niet gewijzigd worden.
- Indien de bouwdelen door middel van recepten zover onderbouwd worden dat informatie beschikbaar is over alle primaire middelen waaruit de bouwdelen zijn opgebouwd is het mogelijk om een aannemersbegroting op te stellen.

Deze methodiek maakt dus ook onderscheidt zoals tussen projecten, gebouwdelen, bouwdelen, begrotingsposten en middelen. Deze niveaus worden echter niet afgedwongen, er zouden dus ook kostenposten op het bouwdelenniveau ingevuld kunnen worden.

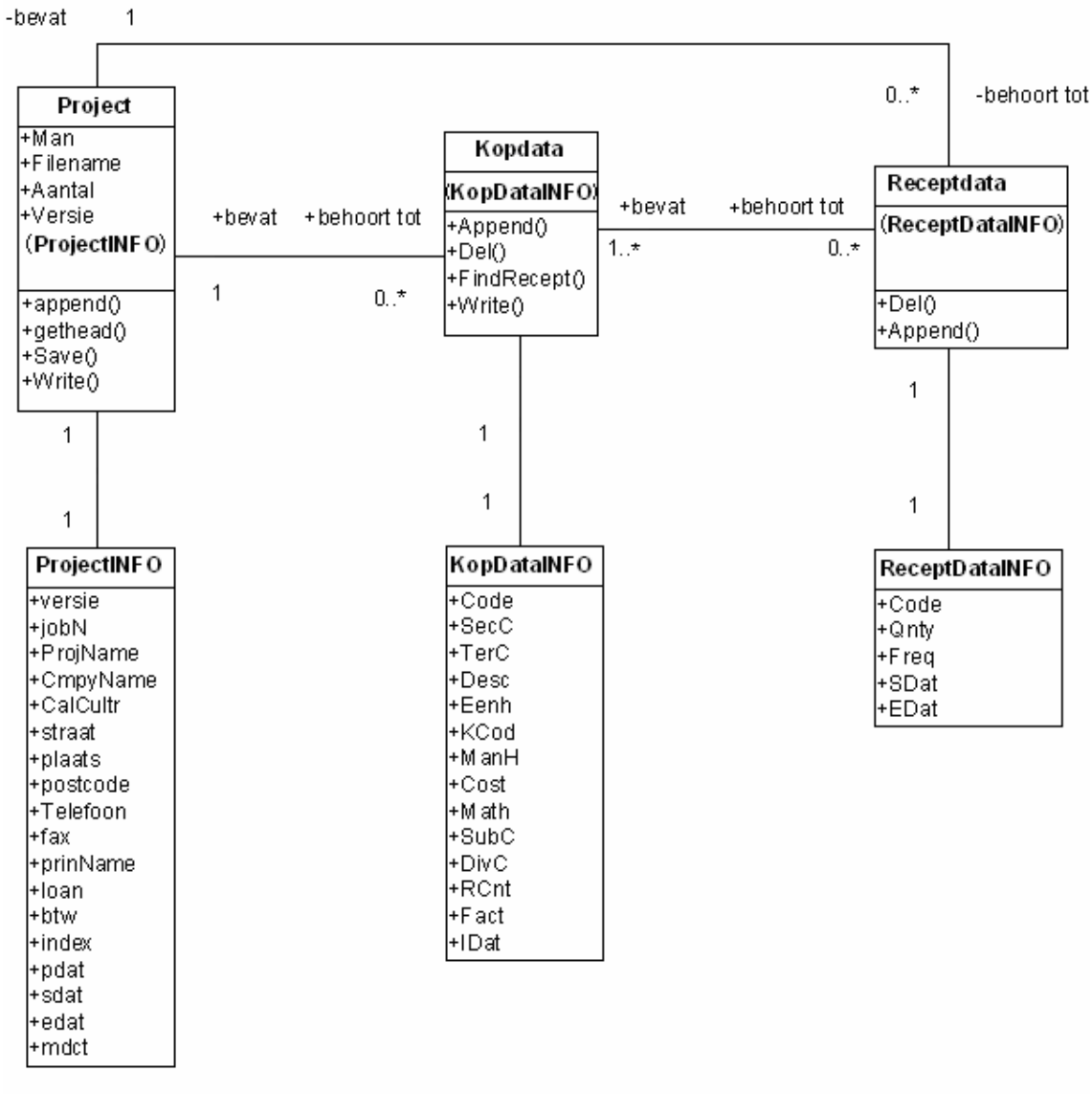
Elk afzonderlijk bouwdeel wordt ongeacht het niveau waarop het voorkomt uniek, dus één maal, beschreven. Dit wordt een kopregel genoemd. Deze kopregel bevat de algemene informatie zoals een unieke code, de omschrijving en de kostenposten. Zo'n kopregel kan onderbouwd zijn met receptregels van bouwdelen op een lager niveau. Omdat zo'n receptregel ook weer een bouwdeel, op een lager niveau voorstelt, houdt dit in dat er voor elke receptregel een kopregel bestaat die dat bouwdeel weer beschrijft.

Voor de weergave van de opslagstructuur is er gebruik gemaakt van een klassendiagram. Hier is voor gekozen omdat Calkey een objectgeoriënteerd applicatie en dus gebruik maakt van klassen. Ik heb alleen de klassen weergegeven die van belang zijn voor de structuur van de begroting.

Een uitgebreide beschrijving van de werking van Calkey en zijn opslagstructuur is terug te vinden in de definitiestudie welke is opgenomen in bijlage A. Ten behoeve van de begrijpbaarheid van dit document is er voor gekozen om enkele facetten van Calkey weer te geven.

6.1.3. Structuur van Calkey

Zoals eerder vermeld is Calkey een objectgeoriënteerd programma. De klassen ten behoeve van het project, de kopregels en de receptregels zijn weergegeven in het hier onderstaande klassendiagram. Aangezien de gegevens van de kopregels, receptregels en de projectgegevens in records staan die onderdeel zijn van de klasse, maar wel apart worden weggeschreven zijn de records apart weergegeven.



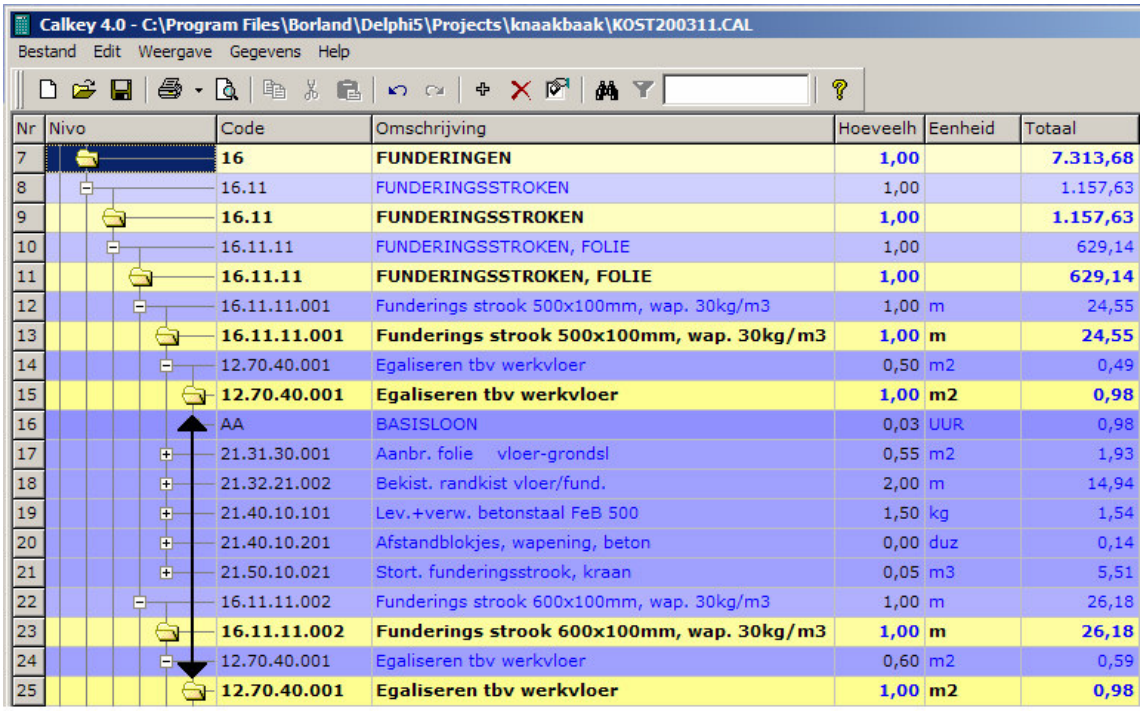
afbeelding 4: klassendiagram van Calkey uit de definitiestudie

Bij de klassen Project, kopdata en receptdata behoren respectievelijk de records ProjectInfo, KopdataInfo en ReceptdataInfo.

In Calkey wordt het attribuut code van het record KopdataInfo gebruikt als koppeling tussen de kopregels en de receptregels. Om er voor te zorgen dat er altijd de juiste kopregel met de juiste code gerefereerd wordt is het van belang dat dit attribuut van KopdataInfo altijd uniek is.

6.1.4. Calkey kopregel structuur

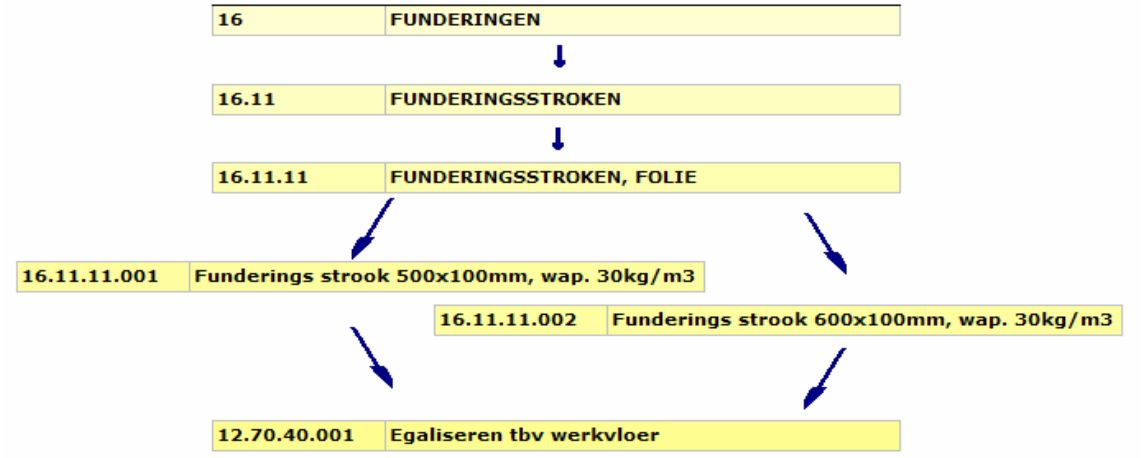
In Calkey is het mogelijk om een kopregel meerdere keren te gebruiken. Dat wil zeggen dat meerdere receptregels naar deze een identieke kopregel kunnen refereren. In de onderstaande afbeelding is te zien dat 'Egaliseren tbv werkvloer' door twee verschillende receptregels gerefereerd wordt.



Nr	Nivo	Code	Omschrijving	Hoeveelh	Eenheid	Totaal
7		**16**	**FUNDERINGEN**	**1,00**		**7.313,68**
8		16.11	FUNDERINGSSTROKEN	1,00		1.157,63
9		**16.11**	**FUNDERINGSSTROKEN**	**1,00**		**1.157,63**
10		16.11.11	FUNDERINGSSTROKEN, FOLIE	1,00		629,14
11		**16.11.11**	**FUNDERINGSSTROKEN, FOLIE**	**1,00**		**629,14**
12		16.11.11.001	Funderings strook 500x100mm, wap. 30kg/m3	1,00	m	24,55
13		**16.11.11.001**	**Funderings strook 500x100mm, wap. 30kg/m3**	**1,00**	**m**	**24,55**
14		12.70.40.001	Egaliseren tbv werkvloer	0,50	m2	0,49
15		**12.70.40.001**	**Egaliseren tbv werkvloer**	**1,00**	**m2**	**0,98**
16		AA	BASISLOON	0,03	UUR	0,98
17		21.31.30.001	Aanbr. folie vloer-grondsl	0,55	m2	1,93
18		21.32.21.002	Bekist. randkist vloer/fund.	2,00	m	14,94
19		21.40.10.101	Lev.+verw. betonstaal FeB 500	1,50	kg	1,54
20		21.40.10.201	Afstandblokjes, wapening, beton	0,00	duz	0,14
21		21.50.10.021	Stort. funderingsstrook, kraan	0,05	m3	5,51
22		16.11.11.002	Funderings strook 600x100mm, wap. 30kg/m3	1,00	m	26,18
23		**16.11.11.002**	**Funderings strook 600x100mm, wap. 30kg/m3**	**1,00**	**m**	**26,18**
24		12.70.40.001	Egaliseren tbv werkvloer	0,60	m2	0,59
25		**12.70.40.001**	**Egaliseren tbv werkvloer**	**1,00**	**m2**	**0,98**

afbeelding 5: Kopregels meerdere malen gerefereerd uit de definitiestudie

Schematisch is de structuur van de kopregels op de volgende manier weer te geven:

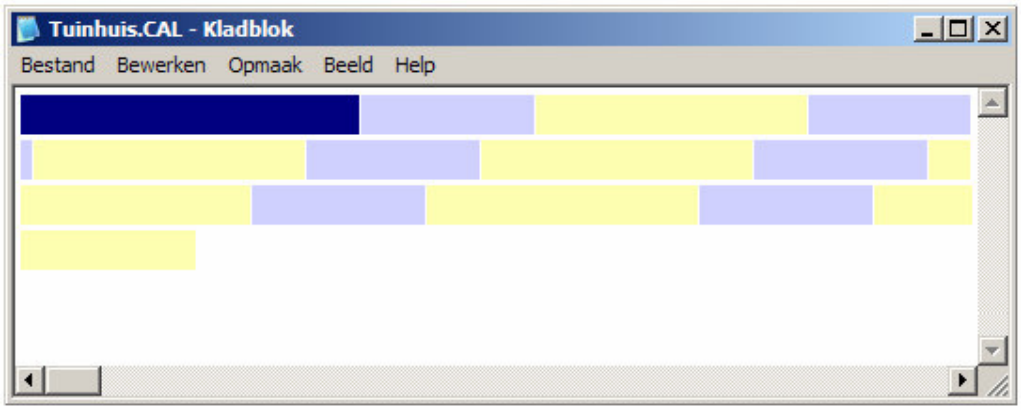


afbeelding 6: Schematische weergave van kopregels meerdere malen gerefereerd.

Op de vorige pagina is schematisch weergegeven hoe twee verschillende receptregels refereren naar één en dezelfde kopregel. Hiermee is Calkey in staat om dynamisch gegevens te veranderen. Als er namelijk in Calkey op één plek een kopregel veranderd dan veranderd deze overal waar naar deze kopregel wordt gerefereerd. Op deze manier kan bijvoorbeeld het loon van een bouwvakker of de prijs van beton per kilogram eenvoudig veranderd worden.

6.1.5. Bestandsstructuur Calkey

De kop- en receptregels staan Calkey in records. Door deze records als blokken weg te schrijven naar een bestand slaat de Calkey de gegevens op. Op deze manier worden ook de projectgegevens, staartbladgegevens, instellingen enzovoorts opgeslagen.



= Projectgegevens = Kopgegevens = Receptgegevens

afbeelding 3: schematische weergave van de opslagstructuur van Calkey uit de definitiestudie

Doordat er bekent is hoe groot dat de in te lezen (dus ook de weggeschreven) blokken zijn kunnen de gegevens weer uit het bestand worden uitgelezen. Om deze reden was het belangrijk dat de sourcecode ook bekend was bij het ontwikkelen van de conversietool. Het wegschrijven van de blokken moest op dezelfde manier gebeuren als bij Calkey.

6.1.6. Aangetroffen problemen

Bij het analyseren van Calkey was het lastig om de structuur van kopregels en receptregels te achterhalen. Doordat er geen documentatie van Calkey aanwezig was heeft het een tijdje geduurd voordat die structuur helder werd. Wel was mijn begeleider, tevens de ontwikkelaar van Calkey, in de buurt om vragen ten behoeve van sourcecode van Calkey te beantwoorden.

6.1.7. De Knaakbaak

Vervolgens moest de structuur van de Knaakbaak beschreven worden. De Knaakbaak maakt gebruik van een tweetal bestanden namelijk projectbestanden en kostenbestanden (ook wel referentiebestanden genoemd). In deze kostenbestanden staan de normprijzen per bouwelement. Deze bestanden zijn ook de voornaamste reden om de conversie uit te voeren.

De kostenbestanden waren, zoals vermeld, Access bestanden. De relaties tussen de tabellen in de Access bestanden waren niet vastgelegd in Access, deze werden programmatechnisch bijgehouden. Hierdoor was niet in één oogopslag duidelijk welke tabellen aan welke tabellen gekoppeld waren. Aangezien de Knaakbaak niet gedocumenteerd was en de ontwikkelaars

niet binnen mijn bereik waren was het niet mogelijk om vragen of de structuur na te zoeken in de documentatie.

Het analyseren van de database ging dan ook niet gemakkelijk omdat deze een tamelijk ingewikkelde structuur had. Ook bevatten de tabellen, voor zover ik weet, veel overbodige velden. Deze velden waren niet nodig voor het opbouwen van de begrotingsstructuur.

Ik heb de kostenbestanden geopend in de Knaakbaak bibliotheek, de applicatie om de kostenbestanden apart te bekijken, en een kopie van dat bestand geopend in MS Access. Hierdoor was het mogelijk om bij elkaar te zoeken welke bouwelementen bij welke bouwelementen hoorden. Door hier verbanden tussen te leggen (welk element hoort bij welk element) was ik in staat om de relaties tussen de verschillende tabellen te definiëren. Met behulp van deze relaties was ik aan het eind van de analyse in staat om alleen met behulp van de database de begroting op te bouwen.

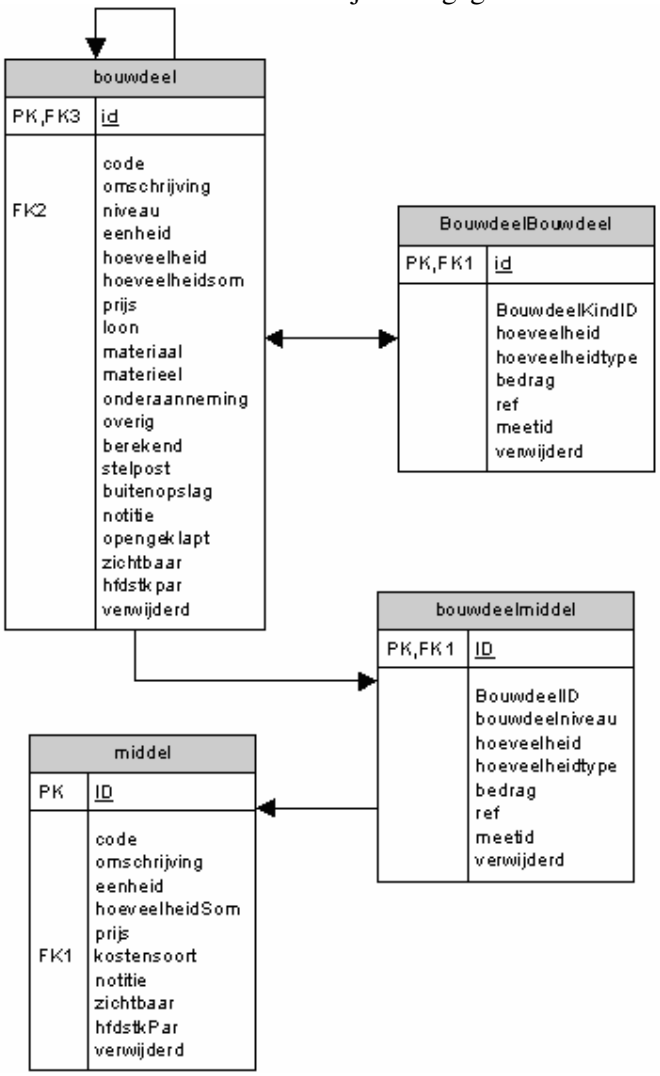
Hierdoor was ik in staat om de structuur weer te geven in een overzicht, dat wil in dit geval dus zeggen, het weergeven van de gebruikte tabellen met de relaties daartussen. De tabellen uit de database die niet benodigd zijn voor het opbouwen van de structuur worden niet in dit overzicht weergegeven.

De gegevens in de tabellen die niet worden weergegeven in dit overzicht zullen verloren gaan tijdens een conversie.

6.1.8. De structuur van de Knaakbaak

Hier wil ik in de structuur van de Knaakbaak weergeven, de bron van de conversie. De Knaakbaak opslagstructuur is voor de kostenbestanden uit gegaan van een gestructureerde begroting. Zo'n begroting ging uit van gebouwdelen, bouwdelen, begrotingsposten en middelen. Hierbij is er voor elk apart project een apart bestand aangemaakt.

In het onderstaande overzicht is de structuur van de Knaakbaak weergegeven. Hier zijn de tabellen waar de gegevens in staan om de begrotingsbestanden uit op te bouwen in opgenomen en de relaties tussen deze tabellen zijn weergegeven:



afbeelding 7: Structuur van de Knaakbaak uit de definitiestudie

In het bovenstaande diagram staan alle velden uit de tabellen weergegeven, hoewel niet alle velden zijn gebruikt. Naast deze vier tabellen waren er nog negen tabellen in de bestanden opgenomen. Deze tabellen werden niet gebruikt voor het opbouwen van de structuur en zijn dus weggelaten.
Dit overzicht stelt geen ERD en geen klassendiagram voor, alleen een schematische weergave van de database. Dit is ook niet nodig omdat de database reeds bestaat en er is alleen van belang in welke verhouding deze gegevens met die van Calkey staan.

6.1.9. Aangetroffen problemen

Omdat de relaties van de kostenbestanden niet in de database waren vastgelegd, moesten de relaties opnieuw worden vastgelegd. Dit proces werd gecompliceerd omdat er veel tabellen en velden waren die niet voor het opstellen van de structuur benodigd waren en omdat de tabellen veel records bevatten.

6.2. Eisen en wensen

De systeemeisen vormen de eisen waaraan het systeem moet voldoen en de eisen vormen de basis waarop het testplan wordt gebaseerd. Een eigenschap van de watervalmethode gebruikt door SDM is dat je de eisen en wensen vooraf al helemaal duidelijk moet hebben. Het is namelijk niet gebruikelijk om terug te stappen in een al afgeronde fase. Hierdoor was het zaak om de eisen en wensen goed in kaart te brengen zodat dit geen problemen op zou leveren. Hierdoor was het ook zaak dat de huidige situatie goed werd beschreven.

De eisen en wensen hangen nauw samen met de doelstelling van de opdracht. In de doelstelling stond namelijk:
Het is de bedoeling het Knaakbaak basis bestand met bouwkosten definities te converteren naar Calkey bestandsstructuur. Bij een conversie zal een controle gedaan moeten worden op verschillende kostenonderbouwingen van bouwdelen met dezelfde codering en omschrijving. Hierbij moet de gehele boomstructuur van de calculatie doorlopen worden. Bij discrepanties moet een signalering volgen in de vorm van een rapportage.

Door middel van de doelstelling was het mogelijk om een aantal eisen op te stellen namelijk:

- De conversietool moet het Knaakbaak basis bestand met bouwkostendefinities kunnen converteren naar Calkey bestandsstructuur.
- Er moet een controle worden uitgevoerd zodat er geen bouwelementen met dezelfde code worden ingevoerd.
- Discrepanties moeten worden weergegeven.

Door de Knaakbaak en Calkey naast elkaar te houden werd het duidelijk dat een volledige conversie niet altijd handig te zijn. De gegevens van de Knaakbaak moeten namelijk zo worden uitgediept dat op het laagste niveau maar één kostensoort per bouwelement mag bevatten namelijk het middelen niveau. In Calkey is het mogelijk om dit tot één niveau hoger te realiseren. Wel treedt er op deze manier gegevensverlies op want de codes en omschrijvingen van de bouwelementen op het laagste niveau in de Knaakbaak worden dat niet opgenomen in Calkey. Hierdoor is er voor gekozen om het samenvoegen van de gegevens op het laagste niveau optioneel te maken.

Dit soort eisen en wensen zijn moeilijk van tevoren te achterhalen. Omdat niemand de beide applicaties vergelijkend bekeken had was het goed dat de huidige situatie dus de Knaakbaak en Calkey beschreven zijn zodat het mogelijk was de applicaties met elkaar te vergelijken. Aangezien Calkey nog niet op de markt uitgebracht was, was het ondervragen van toekomstige gebruikers niet mogelijk. Omdat het gegevensverlies zou veroorzaken maar wel overzichtelijkheid zou bieden zitten er aan het wel en niet samenvoegen zowel voor als nadelen. Hieruit volgde dus de eis:

- De samenvoegen moet de bouwelementen op het laagste niveau, indien aangegeven, comprimeren.

Voor het opstellen van de eisen en wensen zijn er gesprekken gevoerd met de heer Woolderink, de ontwikkelaar van Calkey, en hier kwamen deze, relatief kleine, lijst met eisen en wensen uit. Omdat de conversietool in een later stadium wordt geïntegreerd in Calkey waren er geen eisen wat betreft de interface van de conversietool en omdat elk bestand maar één keer geconverteerd hoeft te worden krijgt de performance ook geen hoge prioriteit.

6.3. Opstellen van de systeemconcepten

Het systeemconcept geeft het te ontwikkelen systeem conceptueel weer zodat het eenduidig genoeg is voor ontwikkelaars en toch begrijpelijk genoeg is voor de opdrachtgever. Door het opstellen van de systeemconcepten wordt er getracht om logische voor de hand liggende keuzes niet over het hoofd te zien. Voor het weergeven van de systeemconcepten is alleen een korte beschrijving gebruikt. De te informeren partijen waren namelijk zelf ook goed op de hoogte van de stand van zaken.

Hieronder staan vier alternatieven uitgewerkt:

Alternatief 1

Het eerste alternatief wat wordt aangedragen is een conversietool die de gegevens converteert door middel van een 'tussenstructuur'. Hier worden de gegevens eerst overgezet van de Knaakbaak naar een aparte structuur, vervolgens kunnen in deze structuur wijzigingen worden aangebracht. Vanuit deze structuur worden de gegeven in de Calkey structuur gezet.

Het voordeel van deze aanpak is dat indien de structuur van de Knaakbaak of Calkey veranderd dat alleen de vertaling naar de tussenstructuur herschreven dient te worden. Indien de structuur in de toekomst ook nog overgezet moet worden naar een derde applicatie kan er overgezet worden vanuit deze tussenstructuur.

Het nadeel is dat er in dit geval twee verschillende conversies uitgevoerd moeten worden.

Alternatief 2

Het tweede alternatief is het direct converteren van de Knaakbaakstructuur naar de Calkeystructuur. Hier worden de gegevens direct vanuit de Knaakbaakstructuur in de Calkeystructuur gezet. Eventuele wijzigingen in de structuur kunnen alleen tijdens dit proces uitgevoerd worden.

Het voordeel is dat er maar één conversie benodigd is bij het overzetten van de gegevens. Mocht echter één structuur veranderen dient de gehele conversietool aangepast te worden.

Alternatief 3

Het derde alternatief is het aanpassen van de structuur van Calkey. Aangezien de structuur van Calkey nog niet vast ligt is het mogelijk om de structuur van Calkey aan te passen op de structuur van de Knaakbaak. Hierdoor zou een conversie overbodig worden en dus de kostenbestanden van de Knaakbaak altijd in Calkey in te lezen zijn. Het nadeel hiervan is dat de eigenschappen van de structuur van de Knaakbaak ook worden meegenomen in Calkey. Dit zou Calkey dus vertragen en hierdoor zou ook de ingewikkelde structuur meegenomen worden in Calkey.

Alternatief 4

Het vierde en laatste alternatief is het handmatig overtypen van de gegevens. Hier wordt er een tijdwinst geboekt omdat er dan geen applicatie ontwikkeld hoeft te worden en er kan ingespeeld worden op een wijziging van de structuur. Het nadeel hiervan is dat de persoon die de gegevens overzet de kennis moet hebben van de structuur van de Knaakbaak en de structuur van Calkey. Daar komt bij dat de kostenbestanden duizenden bouwelementen bevatten, het overzetten kost dus veel tijd en is foutgevoelig.

6.3.1. Keuze

Om het half jaar komt er een update van het kostenbestand uit. Een automatisch oplossing is dus gewenst zodat er niet elk half jaar veel tijd besteed hoeft te worden aan de conversie. De structuren van beide begrotingsprogramma's zijn statisch en er zijn zover het er nu naar uit ziet geen conversiebehoeften naar derde applicaties. De Calkey is ontwikkeld omdat de Knaakbaak te traag werkte. Hierdoor is het geen optie om de structuur van Calkey aan te passen op die van de Knaakbaak. Hierdoor zou ook Calkey traag worden. Om deze redenen is er gekozen voor het schrijven van een conversietool die geen gebruik maakt van een tussenstructuur. Deze conversietool krijgt de naam 'de Knalkey', een samenvoeging van de Knaakbaak en Calkey.

7. Ontwerpen van de Knalkey

In dit hoofdstuk wordt de gehanteerde werkwijze beschreven met betrekking tot het realiseren van het systeemontwerp van de conversietool de Knalkey. Dit is gedaan om het systeemconcept verkozen in de definitiestudie tot op een niveau uit te werken zodat het realiseren van de Knalkey mogelijk was. Hiervoor is, zoals eerder vermeld, gebruik gemaakt van elementen van UML. Niet alle onderdelen van de taal worden gebruikt, alleen de voor dit project bruikbare elementen. Zo worden er bijvoorbeeld geen toestandsdiagrammen beschreven omdat deze diagrammen niets toevoegen aan dit ontwerp. Er zal blijken dat er namelijk geen klassen zijn die veel toestanden aannemen. Als er onderdelen op een andere manier worden ingevuld zal dat per onderdeel worden weergegeven. Dit is gedaan zodat alleen de onderwerpen die echt bruikbaar waren gebruikt werden waardoor het ontwerp compacter maar nog steeds volledig kon zijn.

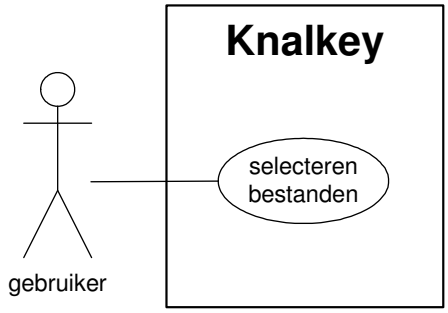
7.1. Ontwikkel mens en machine raakvlakken

Hoewel het gebruikelijk is met UML om een brainstormsessie te houden zodat er kandidaat-klassen gevormd kunnen worden is dit niet gedaan aan het begin van dit ontwerp. In een zekere zin is die brainstormsessie reeds gehouden bij het opstellen van de eisen en wensen en bij het vergelijken van de beide applicaties. Om deze reden heb ik de functionele eisen vermeld in de definitiestudie gebruikt als basis voor de Use Cases.

Een use case geeft de functionele eisen van het systeem weer. Bij een use case wordt van de gebruiker uit gegaan. Hierdoor wordt er een goed beeld gevormd van wat de raakvlakken van de gebruiker zijn met het systeem. Aangezien de use cases worden beschreven in natuurlijke taal is het een informele techniek. Het voordeel hiervan is dat alle te beschrijven eisen en wensen gemakkelijk in de use cases kunnen worden beschreven.

Voor het gebruik van de use case is gekozen om nog een laatste keer naar het eisen en wensen pakket te kijken. Door een frisse blik en een andere weergave zouden er, hoewel we ons in de volgende fase bevinden, weer nieuwe eisen en wensen naar voren kunnen komen. Ook de mens- machine-interface kan op deze manier weergegeven en geeft een duidelijk beeld van wat het systeem volgens een gebruiker moet kunnen.

Vertaald naar de Knalkey stelt de use case niet veel voor. De gebruiker hoeft namelijk niet veel te doen bij het converteren. Het enige wat de gebruiker dient te doen is het selecteren van de te converteren bestanden. Hierdoor is de use case heel eenvoudig namelijk als volgt:



Afbeelding 8: use case uit het ontwerp

De weergegeven use case beschreven in het onderstaande template:

Naam	Selecteren bestanden
Aannamen	De gebruiker heeft toegang tot de te converteren bestanden
Beschrijving	De gebruiker geeft aan of het laagste niveau van de knaakbaak samengevoegd dient te worden. Vervolgens selecteert de gebruiker een Knaakbaakbestand. Dit zijn bestanden met de extensie *.bkb (een project bestand) of *.kkb (een referentie- / kostenbestand).
Uitzonderingen	Het geselecteerde bestand heeft de juiste extensie maar heeft niet de juiste structuur.
Resultaat	De bestanden zijn geconverteerd en, indien aangegeven, het laagste niveau gecomprimeerd. De discrepanties worden door middel van een rapportage weergegeven.

Aangezien het aantal van de raakvlakken tussen de gebruiker en het systeem blijft steken op één, wordt het raakvlak niet verder gespecificeerd. Hier wordt bijvoorbeeld niet de criteria vastgelegd waaraan de toekomstige beeldschermen en lijstindelingen zouden moeten voldoen. Omdat de Knalkey een beperkte interface heeft is de kans op een inconsistente interface gering.

7.2. Specificeren procedures en formulieren

Bij het specificeren van de procedures en formulieren is er gebruik gemaakt van sequence-diagrammen. Een sequence diagram geeft de interactie van de objecten in het systeem weer om het juiste systeemgedrag te verwezenlijken. Hiervoor dient er bekend te zijn welke objecten in het systeem aanwezig zijn. Hiervoor zijn de klassen van Calkey en de Knaakbaak gebruikt. Om de klassen van Calkey en de tabellen van de Knaakbaak weergegeven als klassen in één diagram te plaatsen was het zaak om er achter te komen tussen welke tabellen van de Knaakbaak en welke klassen van Calkey relaties zouden bestaan.

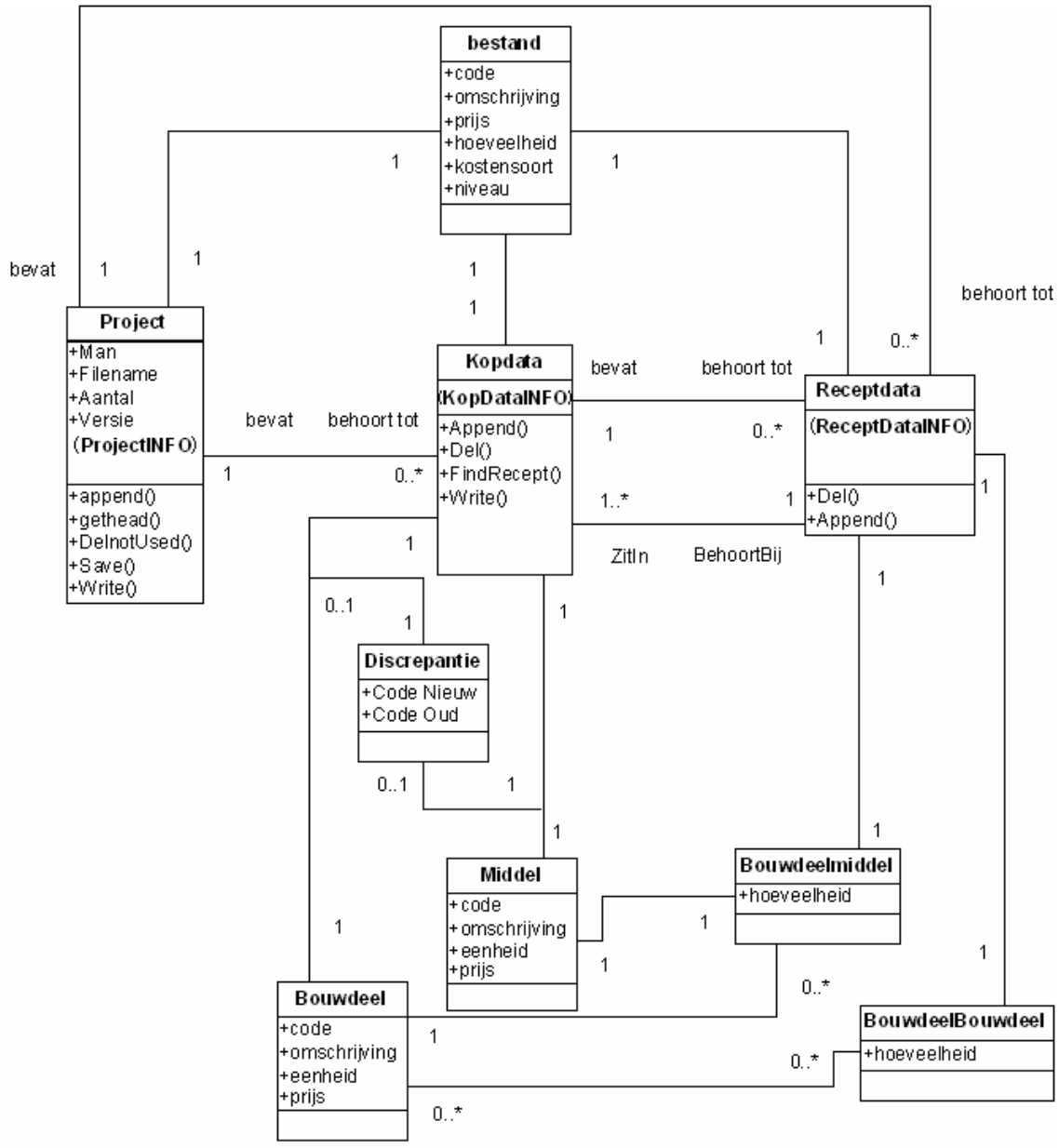
Er is van uit gegaan dat als een gegeven in Calkey klasse benodigd is en als dat zelfde gegeven in een Knaakbaak tabel staat dat er dan een relatie tussen deze twee bestaat. Op deze manier was het mogelijk om lijsten te maken zodat er per attribuut van een Calkey klasse aangegeven staat in welke tabel en welk veld de in te vullen waarde staat. In het volgende voorbeeld een weergave van KopdataInfo.

Het record KopdataInfo:

Attribuut	Tabel	Veld	Voorwaarden
Code	Bouwdeel	Code	
	Middel	Code	
Desc	Bouwdeel	Omschrijving	
	Middel	Omschrijving	
Eenh	Bouwdeel	Eenheid	
	Middel	Eenheid	
Kcod	Middel	Prijs *1	Kostensoort = 1
ManH	BouwdeelMiddel	Hoeveelheid	
Cost	Middel	Prijs	Kostensoort = 2
MatC	Middel	Prijs	Kostensoort = 3
SubC	Middel	Prijs	Kostensoort = 4
DivC	Middel	Prijs	Kostensoort = 5

Door alle benodigde gegevens in Calkey op deze manier te koppelen met de tabellen van de Knaakbaak worden de gegevens die niet in Calkey benodigd zijn overgeslagen.

Door het koppelen van de klassen en de tabellen werd het mogelijk om de beide diagrammen samen te voegen tot één diagram. In de onderstaande afbeelding is het klassendiagram te zien dat gevormd is door het samen voegen van het diagram van Calkey en het diagram van de Knaakbaak. Bij het samenvoegen is ook het diagram van de projectbestanden meegenomen.



afbeelding 9: klassendiagram van de Calkey en de Knaakbaak samengevoegd afkomstig uit het ontwerp.

Ook de klasse discrepantie is toegevoegd aan het klassendiagram om bij te houden welke codes er dubbel in de structuur aanwezig zijn in de Knaakbaak.
De klasse ‘Bestand’ is afkomstig van het inlezen van de projectgegevens van de Knaakbaak. Deze staan namelijk allemaal in één tabel.

7.3. Splitsen sequence diagram

In een sequence diagram doet een gebruiker een serviceaanvraag. In het geval van de Knalkey zou er maar één sequence diagram komen omdat er ook maar één use case was. Wel zou dit een hele grote worden omdat het systeem bij deze service aanvraag veel acties moet uitvoeren. Ten behoeve van de overzichtelijkheid is dit grote diagram opgesplitst in meerdere gedeeltes te weten:

- Het uitlezen van de kostenbestanden van de Knaakbaak
- Het uitlezen van de projectbestanden van de Knaakbaak
- Het invoeren van de elementen in Calkey
- Het samenvoegen van het laagste niveau
- Het wegschrijven van Calkeystructuur

Voor deze scheiding heb ik gekozen omdat het mogelijk was om één enkel element in de structuur van Calkey in te voeren. Hierdoor was het mogelijk om de structuur van de Knaakbaak te doorlopen, zonder dat er rekening gehouden wordt met de structuur van Calkey. Het enige wat bekend moest zijn was het element zelf en de referenties naar de onderliggende elementen.

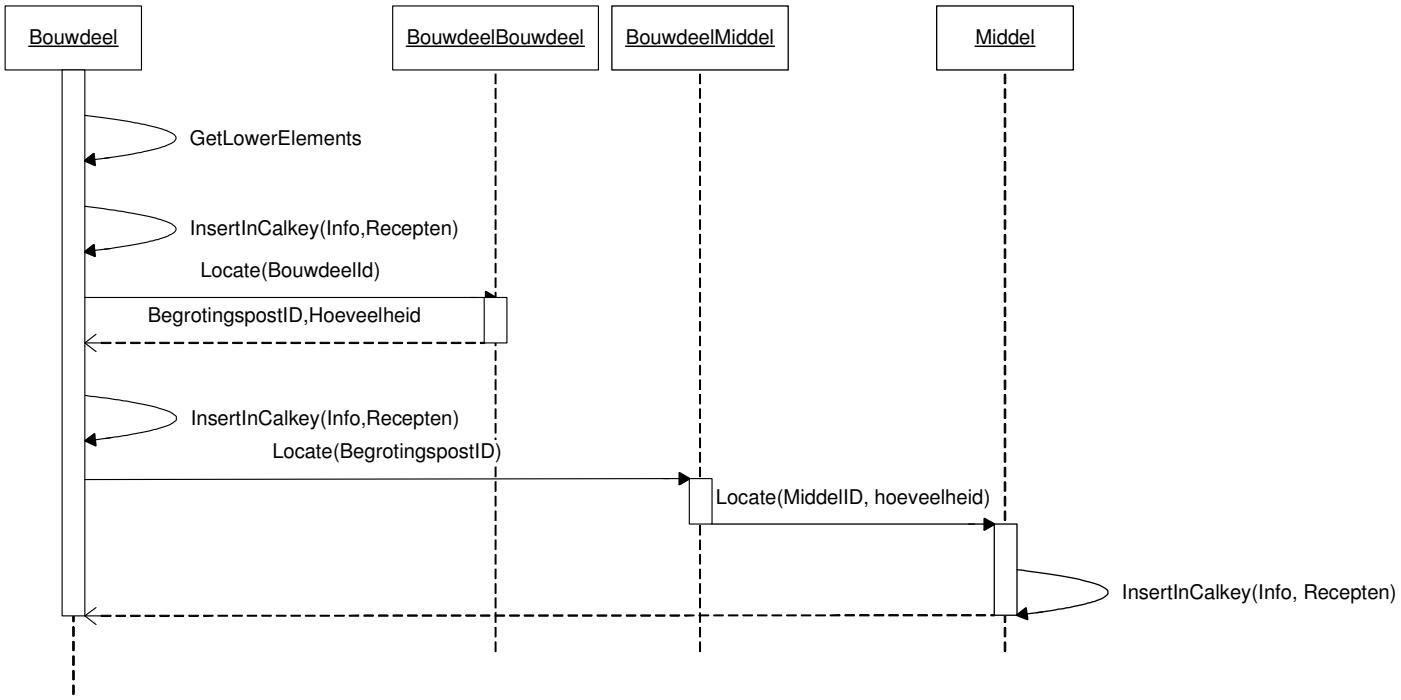
7.4. Sequence diagram

Zoals vermeld wordt het grote sequence diagram dat zou voortkomen uit de use case opgesplitst in meerdere, kleinere sequence diagrammen. De sequence diagrammen worden hierdoor beter begrepen, wel is het zo dat niet elk sequence diagram een actor (gebruiker) heeft. Hierdoor wordt afgeweken van de standaard, wat tot verwarring zou kunnen leiden. Dit is toch gedaan omdat de verwarring, indien het niet zou gebeuren, zeker een feit zou zijn omdat het diagram dan onoverzichtelijk zou zijn.

Bij het opstellen van het diagram voor het inlezen van het kostenbestand kwam ik vaak in de verwarring doordat de structuur van de Knaakbaak te verwarrend was. Hierdoor heb ik de definitiestudie met daarin de beschrijving van de structuur van de Knaakbaak meerdere malen moeten opzoeken. Hierin komt dus het belang naar voren van een goede documentatie van het werk, in dit geval voor het analyseren van de Knaakbaak.

Doordat ik er voor heb gekozen om de elementen van de Knaakbaak apart in Calkey in te voeren dienen de elementen van beneden naar boven in de structuur geplaatst te worden. Indien dit niet gebeurt worden er referenties aan Calkey structuur doorgegeven van kopregels die nog niet in Calkey structuur bestaan en mogelijker wijs ook niet worden ingevoerd.

In het onderstaande diagram staat weergegeven hoe de elementen uit de Knaakbaak structuur worden uitgelezen.



afbeelding 10: uitlezen Knaakbaak kostenbestand afkomstig uit het ontwerp.

7.5. Sourcecode van Calkey

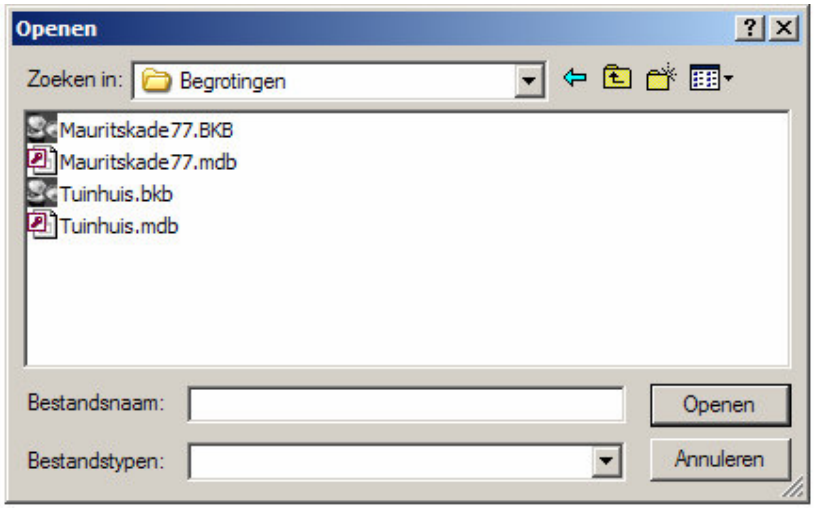
Om Calkey structuur met de sourcecode van Calkey weg te schrijven was het zaak om de sourcecode van Calkey te begrijpen. Doordat mijn begeleider ook programmeur van de applicatie was, was ik in de gelegenheid om veel vragen over de code te stellen. Vaak was het wel een gepuzzel omdat de heer Woolderink veel elementen heeft geoptimaliseerd waardoor de code voor een leek minder begrijpbaar was.

Hierdoor was het wel mogelijk om de sourcecode van Calkey te gebruiken. Helaas heeft dit naar alle waarschijnlijkheid geen tijdswinst opgeleverd omdat het uitzoeken net zoveel tijd heeft gekost als het opnieuw schrijven van de sourcecode.

Wel ligt hierin een groot voordeel. Indien de structuur van Calkey ooit mocht veranderen dan wordt de oude code vervangen door de nieuwe code (dezelfde als van Calkey). Hierdoor wordt ingesprongen op mogelijke veranderingen zoals het veranderen van de recordstructuur die wordt weggeschreven. Omdat alle records als blokken worden weggeschreven is er ook van belang hoeveel ruimte voor een attribuut wordt gereserveerd. Een verandering van de grootte zou dus fataal kunnen zijn.

7.6. Specificeer beeldscherm en lijstindelingen

Het doel van deze activiteit is het vervaardigen van de fysieke ontwerpen van beeldscherm en lijstindelingen. Als basis hiervoor werd de use case gebruikt. De use case gaf immers de interactie met de gebruiker weer. In het geval van de Knalkey is er, zoals eerder vermeld, weinig interactie met de gebruiker. De gebruiker hoeft alleen maar de bestanden te selecteren die geconverteerd moeten worden. Voor het selecteren van bestanden heeft Delphi een speciaal component. Dit component is te gebruiken met filter zodat alleen de bestanden met een bepaalde extensie getoond worden, bestanden navigatie en het component ziet er als volgt uit:



afbeelding 12: Scherm openen bestanden

Aangezien er maar twee schermen benodigd zijn namelijk het hoofdscherm (waarin de discrepanties worden weergegeven) en het bovenstaande scherm ben ik niet verder ingegaan op de interactie tussen de schermen. Ik ben er van uit gegaan dat het aannemelijk is dat als er op de knop openen wordt gedrukt dat dit scherm te voorschijn komt.

7.7. Specificeren programmatuur

Door deze activiteit uit te voeren heb ik getracht om de fysieke structuur van de gehele programmatuur meer in kaart te brengen. Om dat de sequence diagrammen de interactie tussen de objecten weergeeft is het voldoende om de alleen een beschrijving te geven van de gebruikte operaties. Ook heb ik nog een keer weergegeven welke velden er uit welke tabellen van de Knaakbaak gebruikt worden, dit maal met toevoeging van de typen van de velden.

Per operatie moet de functie bekend zijn en de in- en uitvoer zal specifiek worden beschreven. Dit wordt gedaan om, van te voren, het realiseren van de Knalkey te versnellen. Hieronder heb ik een beschrijving weergegeven van gebruikte procedure in de Knalkey:

Naam: IsDiscrepantie.

Preconditie: Er bestaat een project.

Postconditie: Er is duidelijk of de code van een kopregel al bestaat, als dit het geval is, is er een nieuwe, unieke code voor de kopregel gegenereerd.

Omschrijving: Zoekt een kopregel met dezelfde code. Als deze niet bestaat is het geen discrepantie. Als deze wel bestaat moet er een unieke code worden samengesteld.

8. Vervaardigen testplan

Bij het vervaardigen van het testplan was moest er een manier worden beschreven hoe er, in een later stadia getest dient te worden. Aangezien het kostenbestand in de Knaakbaak en het geconverteerde bestand in Calkey eenvoudig te testen zijn (deze dienen namelijk, in het geval van geen samenvoegen van de structuur, hetzelfde te zijn) wordt er gebruik gemaakt van black-box testing. Hierbij wordt alleen de in- en uitvoer van de Knalkey gecontroleerd en de interne werking van de Knalkey wordt buiten beschouwing gelaten.

De structuur van het kostenbestand was hierdoor al vanaf het begin testbaar, de originele en de nieuwe moesten namelijk gelijk zijn aan elkaar. De geconverteerde bedragen gekoppeld aan de bouwelementen op het laagste niveau (de middelen in de Knaakbaak) moesten ook vergeleken worden. Om voorkomen te worden dat alle 10.000 middelen gecontroleerd moeten worden, wordt er voor het testen het kostenbestand in de Knaakbaak ingelezen (bedenk dat de kostenbestanden normaal in de bibliotheek geopend worden). Hierdoor worden alle kosten door de Knaakbaak opgeteld. Aangezien de kosten in Calkey ook altijd opgeteld worden kunnen de twee totalen met elkaar vergeleken kunnen worden. Deze totalen behoren hetzelfde te zijn.

9. Realisatie

In dit hoofdstuk wordt beschreven hoe de realisatie van de Knalkey is verlopen en welke problemen hierbij zijn opgetreden. Ook het realiseren van de Knalkey heb ik gedaan op de in plan van aanpak beschreven en ook beschikbaar gekregen PC. Hier was ook de ontwikkelomgeving Delphi aanwezig. Hierbij is niet voor de laatste versie gekozen maar voor de versie waarin Calkey ook is ontwikkeld zodat er geen complicaties optreden als de code van de Knalkey later in Calkey zal worden gebruikt.

Het gebruik van Delphi had wel weer enige opfris tijd nodig, het was twee jaar geleden dat ik voor het laatst met Delphi had geprogrammeerd. Dit duurde twee dagen tot alles weer opgehelderd was.

9.1. Basis Knalkey

Bij het realiseren van de Knalkey moest eerst de basis worden gelegd voor de applicatie. Dit is het klassendiagram. In het geval van de Knalkey was het een uitzonderlijke situatie. Het grootste gedeelte van het klassendiagram bestond al, de helft was namelijk de structuur van Calkey en de andere helft was van de structuur was van de Knaakbaak.

Aangezien Calkey een apart data-unit had, waar alle klassen definities in stonden, was mogelijk om de structuur van Calkey ook te gebruiken in de Knalkey. Wel werden er op deze manier veel klassen in de structuur meegenomen die niet voor de Knalkey relevant zijn. Hierdoor wordt de structuur van Knalkey onnodig groot terwijl maar een klein deel daarvan gebruikt wordt. Toch heb ik voor deze aanpak gekozen. Het voordeel van deze aanpak is dat als de Calkeystructuur eenmaal mocht veranderen, het vervangen van de data-unit de structuur van de Knalkey gelijk afstemt op de Calkeystructuur.

In het geval van de structuur van de Knaakbaak lag het iets gecompliceerder. De gegevens van de Knaakbaak stonden in een Access database. In het begin dacht ik hier weinig problemen mee te hebben, ik had het dit immers ook al tijdens mijn stage gedaan. Na enig onderzoek bleek dat ik tijdens mijn stage gebruik had gemaakt van componenten die door het stagebedrijf ontwikkeld waren. Delphi is namelijk een ontwikkelomgeving waarin gemakkelijk componenten ontwikkeld kunnen worden. Met deze componenten was het mogelijk om run-time MS Access databases te koppelen en deze gegevens toegankelijk te maken. Helaas had ik dat component niet binnen mijn bereik en had ik geen tijd om het component zelf te ontwikkelen.

Hierdoor ben ik op zoek gegaan naar alternatieven zoals bijvoorbeeld free-ware componenten op het internet. Gelukkig heb ik een component kunnen vinden en heb het component dan ook gebruikt. Dit zoeken heeft me wel tijd gekost, terwijl ik dat niet had verwacht. Door de marge die ik heb ingebouwd in de planning ben ik niet uitgelopen. Dit voorval geeft wel aan dat er bij het opstellen van een planning rekening gehouden dient te worden met onvoorziene omstandigheden.

Toen de tabellen van de Knaakbaak toegankelijk gemaakt waren, werd kwam het volgende probleem aan het ligt. De tabellen van de Knaakbaak zijn als klassen in het klassendiagram weergegeven terwijl dit tabellen zijn. In het ontwerp zijn wel procedures aan de klassen toegevoegd terwijl dit niet eens kan. Het is namelijk niet mogelijk om in de tabellen van Access operaties toe te voegen en deze vervolgens te gebruiken vanuit Delphi.

Nu is het zo dat Delphi met de databases communiceert met behulp van klassen, bij dit project door middel van de klassen van het zojuist vermeldde free-ware component. Hierdoor kom ik niet in de problemen doordat ik de tabellen als klassen heb weergegeven. Eventueel toegevoegde operaties zouden aan deze klassen toegevoegd kunnen worden. De klassen binnen de ontwikkelomgeving die de tabellen benaderen en de tabellen van de Knaakbaak worden schematisch dan ook als één geheel gezien.

Door deze samenvoeging te maken als basis voor het systeem was het mogelijk ten eerste een redelijke weergave te geven in welk stadia het realiseren was aanbeland. Ook werd het realiseren hierdoor een stuk overzichtelijker. Hierdoor was het mogelijk om per sequence diagram de realisatie door te voeren. Doordat er per sequence diagram werd gerealiseerd kon ook hier de voortgang van de realisatie aan worden afgelezen.

Hierdoor moest er wel een bepaalde volgorde komen voor het realiseren van de verschillende delen van de sequence diagrammen omdat de deelsystemen dan niet getest zouden kunnen worden. Hiervoor is logischer wijze voor de volgende volgorde gekozen

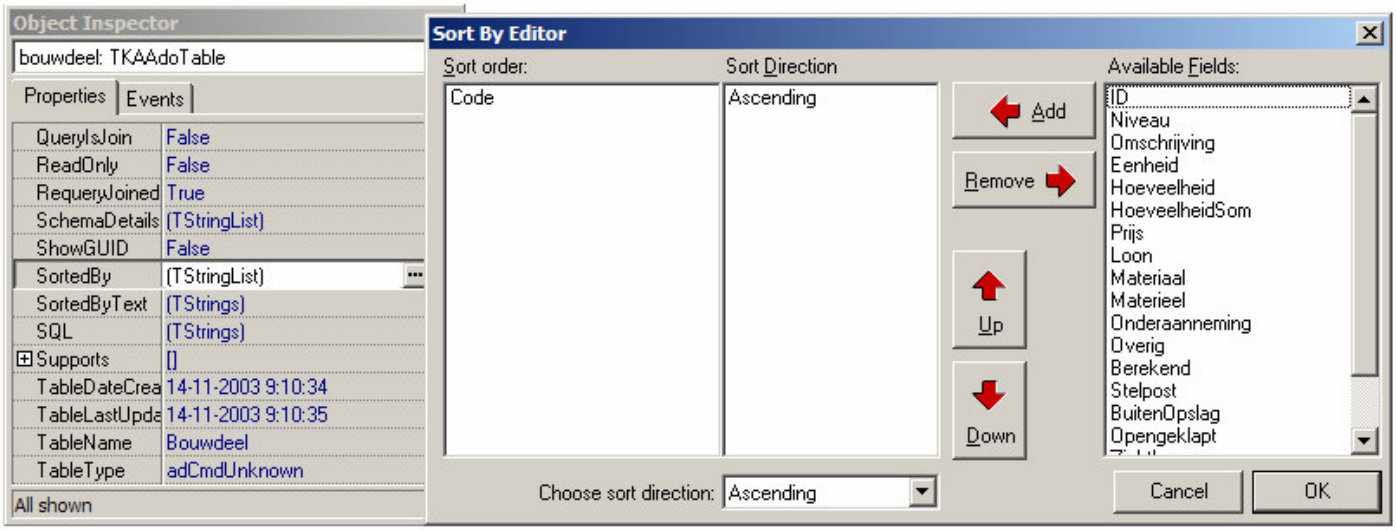
- Inlezen
- Converteren
- Wegschrijven
- Comprimeren

Het was een optie om het comprimeren direct bij het converteren uit te voeren. Het nadeel was dan dat er aanzienlijk meer functies geschreven zouden moeten worden en dat daarvan de performance winst miniem zou zijn. De winst zou zich alleen voordoen als er gecomprimeerd moest worden.

9.2. Inlezen van Knaakbaak kostenbestand

Bij het realiseren van het sequence diagram “Inlezen van het Knaakbaak kostenbestand” traden een aantal problemen op. Bij het beschrijven van de huidige situatie is er van uit gegaan dat de bouwdelen gekoppeld zijn door middel van de code. Helaas was de tabel niet gesorteerd op de code. Ik ben bekend met sql-queries, ik was in het begin ook van plan om deze te gebruiken. Hiermee kon ik de gegevens van de Knaakbaak sorteren, filteren, groeperen en koppelen. Hiermee kon ik ook al bouwdelen uitlezen. Dit leidde pas tot conflicten toen ik runtime de database waaraan de applicatie gekoppeld was, wilde wijzigen. Dit bleek niet mogelijk te zijn. Hierdoor ben ik gaan zoeken naar een andere oplossing.

Door middel van gebruik van het free-ware component was ik in staat om de bouwdelen te sorteren op code.



Hiervan heb ik ook gebruik gemaakt om de tabellen op het veld ‘code’ te sorteren. Omdat er aan de hand van de code van de bouwdelen gekeken kan worden welke bouwdelen bij elkaar horen stonden de bouwdelen goed onder elkaar. Op deze manier wist ik dat als ik een bouwdeel met code ’12’ bereikte dat alle bouwdelen waar het bouwdeel met de code ‘11’ naar refereerde al behandeld waren.

Ook was het mogelijk om door middel van het component de gegevens te filteren. Hier heb ik van gebruik gemaakt en gefilterd op het veld ‘niveau’. De bouwdelen hadden namelijk waarde ‘2’ bij niveau en de begrotingsposten hadden de waarde ‘3’.

Voor het uitlezen van de bouwdelen heb ik gebruik gemaakt van een recursieve functie. Hoewel het schrijven van een recursieve functie vaak meer tijd kost en minder begrijpbaar is heb ik er toch voor gekozen. Op deze manier kon ik bouwdelen tot op een oneindig niveau behandelen. Ook het bij elkaar zoeken van de elementen die bij het bouwdeel horen zijn hierdoor gemakkelijker op te sporen. Nog een bijkomend voordeel hiervan is dat de tabel maar één keer doorlopen dient te worden, hetwelk aanzienlijk in performance scheelt.

```
function Tform1.GetLowerElements(kopcode:string; Id:integer):Tlist;
var
  Recept : TreceptInfo;
  Kopdata : TkopInfo;
  Identifier : integer;
  Record1 : Trecord;
  Recepten : tlist;
begin
  result := Tlist.create;
  while not datamodule1.bouwdeel.Eof do begin
    Record1 := Trecord.create();
    Record1.Code := datamodule1.bouwdeel.fieldvalues['code'];
    Record1.Id := datamodule1.bouwdeel.fieldvalues['id'];
    Record1.Omschrijving := datamodule1.bouwdeel.fieldvalues['omschrijving'];
    Record1.Niveau := 3;
    Record1.Eenheid := datamodule1.bouwdeel.fieldvalues['Eenheid'];
    if (kopcode = '*') or (pos(trim(kopcode),datamodule1.bouwdeel.fieldvalues['code'])=1) then begin
      Identifier := datamodule1.bouwdeel.fieldvalues['ID'];
      Recept := TreceptInfo.create();
      Recept.code := datamodule1.bouwdeel.fieldvalues['Code'];
      Recept.Hoeveelheid := 1;//datamodule1.bouwdeel.fieldvalues['Hoeveelheid'];
      result.Add(Recept);
      kopdata := TkopInfo.create();
      kopdata.code := datamodule1.bouwdeel.fieldvalues['Code'];
      datamodule1.bouwdeel.Next;
      recepten := GetlowerElements(Kopdata.code,Identifier);
      record1.code := InsertInCalkey(record1,recepten);
      recept.code := record1.code;
      Bouwelementen.add(Record1);
    end else begin
      exit;
    end;
  end;
end;
```

Zoals te zien in de bovenstaande functie wordt hier reeds gebruik gemaakt van InsertInCalkey. Met deze procedure worden de elementen, zoals ook weergegeven in het ontwerp, in Calkey structuur gezet.

Vaak wilde mijn begeleider weten hoe ver ik was met het uitvoeren van de conversie en ook ik wilde weten of het uitlezen van de Knaakbaak kostenbestanden gelukt zou zijn. Hierdoor heb ik er voor gekozen om het uitlezen van de projectbestanden te laten voor wat het was. Het uitlezen van de projectbestanden zou op een later tijdstip gerealiseerd worden. Ik ben overgegaan op het overzetten van de gegevens naar de structuur van Calkey zodat de gegevens eerder weggeschreven zouden kunnen worden. Hierdoor kunnen de gegevens in Calkey ingelezen worden. Aangezien het uitlezen van de projectbestanden een apart sequence diagram was het niet waarschijnlijk dat dit conflicten op zou leveren, dit is dan ook niet gebeurd.

9.3. InsertInCalkey

Bij het realiseren van het sequence diagram ‘InsertInCalkey’ heb ik lang gezocht naar de structuur van Calkey, terwijl het in de huidige situatie goed beschreven was. Calkey gebruikte namelijk veel pointers om de gegevens run-time snel beschikbaar te hebben zodat de bewerkingen op de gegevens snel uitgevoerd kunnen worden. Aangezien het bij de conversie niet nodig is om de Calkeystructuur te doorlopen heb ik die pointers ook niet nodig. Ook bij het wegschrijven van de structuur worden de pointers niet gebruikt. Hiervoor heb ik de pointers weggelaten waardoor de structuur van Calkey een stuk eenvoudiger werd.

Bij het testen van dit onderdeel ben ik keer op keer tegen dezelfde bug opgelopen. Maar ik heb hem niet kunnen ontdekken in mijn eigen code. Het heeft lang geduurd voordat ik me bedacht dat Calkey ook nog in ontwikkeling was en dat deze applicatie ook nog bugs zou kunnen bevatten. Dit bleek het geval te zijn. Gelukkig ben ik hierdoor niet in tijdnood gekomen, wel was mijn voorsprong weer ‘netjes’ weggewerkt. Door dit voorval ben ik achterdochtig geworden ten behoeve van Calkey, omdat het me best veel tijd had gekost. Dit bleek later niet nodig zijn geweest, dat was namelijk de enige bug die in Calkey zat ten behoeve van de conversietool.

9.4. Samenvoegen Calkeystructuur

Het realiseren van 'het samenvoegen van de Calkeystructuur' verliep moeizaam. Ik liep namelijk tegen het probleem aan dat ik niet wist wanneer kopregels samengevoegd dienden te worden. Van te voren stond vast dat het ging over de elementen op het laagste niveau en het niveau daarboven. Wat minder duidelijk was of er ook samengevoegd kon worden indien er naar elementen werd gerefereerd met dezelfde kostensoorten.

De gedachte achter het samenvoegen was, zo is me verteld, dat de bedragen in de kopregels gevuld zouden zijn. Per kopregel was het mogelijk om verschillende kostensoorten op te nemen. Als er meerdere kostensoorten in één veld zouden komen te staan dan zouden de kostensoorten bij elkaar opgeteld moeten worden. Dit zou een verkeerd beeld geven van de kopregel. Hierdoor is er voor gekozen om alleen samen te voegen als er gerefereerd naar elementen met stuk voor stuk aparte kostensoorten.

De performance bij het comprimeren werd een probleem. Omdat het niet mogelijk om per kopregel te kijken of naar het element gerefereerd werd moet er voor het verwijderen van een kopregel alle receptregels worden nagelopen op de verwijderde code. Hiervoor het ik een lijst bijgehouden welke kopregels verwijderd werden en de recepten aan het eind van dit proces uitgevoerd. Hierdoor was het mogelijk om de lange lijst recepten maar 1 keer te doorlopen. Per recept wordt er dan gekeken of het element waar naar gerefereerd wordt is verwijderd. Hierdoor werden er aanzienlijk minder vergelijkingen uitgevoerd en verbeterde de performance aanzienlijk.

9.5. Wegschrijven Calkeystructuur

Bij het wegschrijven van Calkeystructuur is er, zoals eerder vermeld, gebruik gemaakt van de sourcecode van Calkey. Ik heb hiervoor eerst de code van Calkey moeten bestuderen. Hier kwam ik er achter dat er gerefereerd werd naar een pointer die ik niet gebruikt heb. Deze pointer werd in Calkey gebruikt om er zeker van te zijn dat de juiste code bij de recepten werd gezet. Deze beveiliging heb ik eruit gehaald, ik had hem ook niet nodig omdat ik geen bewerkingen doe op de elementen dus kan de code ook niet (gewild of ongewild) veranderen. Hierdoor heb ik niet geheel de zelfde code van Calkey kunnen gebruiken, dit is dan ook duidelijk weergegeven in de documentatie.

9.6. Afronden Knalkey

Na het realiseren van de in het ontwerp opgenomen sequence diagrammen waren alle functionaliteiten aanwezig in de Knalkey. Wel bevatte de Knalkey er nog een paar negatieve eigenschappen:

- Tijdens het converteren leek het net of de Knalkey was vast gelopen omdat de Knalkey dan niet reageerde op acties van de gebruiker.
- Het was nog niet mogelijk om meerdere bestanden te selecteren.
- De Knalkey was langzaam.

Door middel van het toevoegen een progressbar was het mogelijk om aan te tonen dat de Knalkey nog bezig was met het converteren / comprimeren van de bestanden.
De toevoeging van dit element koste wel iets van de performance die al slecht was, maar over het algemeen geldt dat gebruikers willen weten hoe lang ze moeten wachten omdat ze anders het proces afbreken.

Door het bijhouden van de bestanden (de directory en de bestandsnaam) was het mogelijk om de gebruiker meerdere bestanden te kunnen laten selecteren. Hierbij werd het proces meerdere keren herhaald waardoor de gebruikersvriendelijkheid van de Knalkey omhoog is gegaan.

9.7. Optimaliseren

Het optimaliseren van de Knalkey was niet in de planning opgenomen. Wel was het nodig om enige optimalisatie uit te voeren omdat het converteren van de kostenbestanden vrij lang duurde, ongeveer een halve minuut. De kostenbestanden bevatten duizenden elementen waardoor, met inefficiënte benadering van deze gegevens veel tijd verloren gaat.

De optimalisatie staat niet beschreven in het plan van aanpak en was daardoor ook niet verplicht. Door mijn streven naar een goede conversietool en aangezien ik vooruit liep op de planning heb ik er voor gekozen om de Knalkey te optimaliseren.

Bij het zoeken naar de manier om te optimaliseren was het zaak om te kijken welke operaties de meeste tijd kosten. Hieruit bleek dat de operaties die de gegevens uit de database van de Knaakbaak haalden het meeste tijd kosten. Hier kon natuurlijk niets aan veranderd worden. Wel werden sommige elementen meerdere keren benaderd.. Zo kan een bepaald element, ook in de Knaakbaak, meerdere malen worden gerefereerd. Hierdoor werden er onnodig veel operaties uitgevoerd die gegevens opvroegen uit de database.

Er is voor gekozen om de elementen in één lijst te zetten. Hierdoor werd elk element in uit de database maar één keer benaderd. Het nadeel hiervan is dat de elementen waar niet naar wordt gerefereerd ook uit de database ingelezen worden.

Delphi heeft een klasse genaamd TList waarbij het mogelijk was om objecten in te zetten. Het is ook mogelijk om de elementen door middel van het quicksort algoritme de elementen te sorteren. Hiervoor diende een procedure meegegeven te worden en Tlist handelde de rest af. Aangezien dit sorteeralgoritme het snelste is, is hier ook van gebruik gemaakt.

Wat ik wel miste was een BubbleSort algoritme in de klasse voor het zoeken van de objecten. Om dit probleem op te lossen heb ik een procedure geschreven die de elementen op basis van de code, waar de elementen op gesorteerd staan, opzoeken door middel van het algoritme.

Door het toepassen van deze algoritmes is de performance van de conversietool ongeveer verdubbeld.

9.8. Testen van de Knalkey

Bij het testen is het opgestelde testplan uitgevoerd. Bij het uitvoeren van de test zijn er ook een aantal bugs uit de Knalkey naar voren gekomen. Dit waren voornamelijk bugs gerelateerd aan de structuur.

Ik ben in de mogelijkheid geweest om de bugs op te lossen en zodoende er voor te zorgen dat de conversie juist wordt uitgevoerd. Verder hebben zich bij het testen van de Knalkey geen problemen voorgedaan.

10. De evaluatie

In de voorgaande hoofdstukken heb ik de activiteiten, keuzes, technieken en methodes die tijdens mijn afstudeerperiode naar voren zijn gekomen behandeld. In deze evaluatie wordt teruggeblikt op de afstudeerperiode. Dit zal gedaan worden aan de hand van het proces en de producten welke ik tijdens deze periode heb doorlopen en opgeleverd. Tenslotte wordt de evaluatie beëindigd met de geboekte leerwinst en een conclusie.

10.1. Proces

10.1.1. Opdrachtgever, begeleiding en samenwerking.

Tijdens mijn afstudeerperiode heeft de heer Woolderink mij begeleid. Hij was samen met de heer Pleijsier mijn opdrachtgever voor de opdracht. Voornamelijk de informele sfeer en het altijd aanspreekbaar zijn is mij goed bevallen. Dit heeft dan ook zijn vruchten afgeworpen bij bijvoorbeeld het onderzoeken van de Calkeystructuur en het opstellen van de eisen en wensen van de Knalkey.

De samenwerking met mijn collega's is mij goed bevallen. De overige collega's waren niet bij dit project betrokken. Dit is niet ten koste gegaan van de sfeer op het kantoor. Ik heb dan ook een gezellige en leerzame tijd gehad bij ALFA Development.

10.1.2. De planning

Het plannen van een project heb ik altijd als lastig ervaren, zo ook deze keer. Deze keer werd het nog een keer gecompliceerd omdat het analyseren van de Knaakbaakstructuur een onzekere factor was. Om dit probleem op te lossen heb ik het zekere voor het onzekere genomen en voldoende tijd ingepland voor de definitiestudie.

Aangezien het analyseren van de Knaakbaakstructuur sneller was verlopen dan gepland liep ik al vanaf het begin van het project voor op de planning. Hierdoor heb ik meer tijd gehad voor bijvoorbeeld het optimaliseren van de Knalkey en het schrijven van het afstudeerverslag. Toch ben ik niet geheel tevreden, ook het inlopen op een planning toont aan dat er verkeerd is gepland. Wel kan ik zeggen dat het plannen mij steeds beter afgaat, dit was namelijk de beste planning die ik gevolgd heb voor het uitvoeren van een opdracht van 20 weken.

10.1.3. Methoden en technieken

Bij het project heb ik gebruik gemaakt van SDM en UML. SDM heeft mij geholpen bij het beheersbaar maken van het project. Vooral de scheiding van de fases zijn goed bevallen. Hierdoor was het mogelijk om na elke fase even afstand te nemen van het project en hierdoor goed overzicht te houden over het te volgen traject.

Ook heeft het geen problemen opgeleverd dat de er per fase gewerkt moet worden en dat het niet wenselijk is dat er terug wordt gestapt in een al afgehandelde fase. Zo was bijvoorbeeld mogelijk om de eisen en wensen van tevoren op te stellen en is ben ik bij het realiseren niet tegen problemen aangelopen die zonder het aanpassen van het ontwerp onoplosbaar waren.

Doordat het een relatief klein traject was heb ik geen last van ontwikkelmoeheid. Dit zou kunnen optreden als er na lange tijd van inspanning geen tastbare resultaten opgeleverd worden. Hierdoor heb ik geen moment getwijfeld aan een succesvolle afloop.

Door middel van de use cases, klassendiagrammen en sequencediagrammen uit de taal UML was ik in staat het gewenste systeem te modelleren. Doordat Calkey ook object georiënteerd was en ik delen van Calkey over zou nemen, heb ik er voor gekozen om voor de Knalkey te ontwikkelen in UML. Hierdoor ben ik niet in problemen gekomen door het gebruik van de sourcecode van Calkey. Ook omdat ik vanuit mijn studie al bekent was met de taal had ik geen tijd nodig om de taal te leren. Mede omdat het traject succesvol is gelopen was het plezierig om met UML te werken.

Het realiseren van de Knalkey is uitgevoerd in de ontwikkelomgeving Delphi 5.0. Hoewel dit een oude versie is, werkte het prettig en zeker de help functie is goed opgebouwd. Ik heb de help functie ook met zekere regelmaat geraadpleegd en heb met plezier de Knalkey in Delphi 5.0 ontwikkeld.

10.2. Product

10.2.1. Plan van aanpak

Het opstellen van een plan van aanpak heb ik nooit leuk gevonden. Het liefst ga ik direct aan de slag en laat ik dat soort zaken links liggen. Wel heb ik dit project ondervonden dat het plan van aanpak duidelijkheid schept in hoe het project wordt aangepakt en daardoor het project beter beheersbaar maakt. Het plan van aanpak is voor grote delen gebaseerd op de opdrachtomschrijving, welke al voor het begin van de afstudeerperiode bekend was. Hierdoor was het opstellen van het plan van aanpak niet veel moeite maar heeft daardoor niet minder bijgedragen aan de werking daarvan.

10.2.2. Definitiestudie

Het doel van de definitiestudie was om te kijken of het ontwikkelen en realiseren van de conversietool haalbaar en zinvol zou zijn. Hiervoor heb ik de Knaakbaakstructuur en de Calkeystructuur moet beschrijven. Door het beschrijven van deze structuren werd duidelijk bij mij wat er allemaal voor de conversie nodig was.

10.2.3. Ontwerp (basisontwerp en detailontwerp)

Dit product is samengesteld om het systeemconcept (het realiseren van de conversietool) te verfijnen tot op een niveau waarop het mogelijk wordt om het systeem te realiseren. Ook is er hier vooruit gedacht over het testen van de Knalkey. In de meeste gevallen vind ik deze fase de minst leuke, wel blijkt elke keer weer dat het onmisbaar is. Ook deze keer ben ik weer door het schrijven van het ontwerp verplicht geweest om vooruit te denken, waardoor het realiseren gemakkelijk is verlopen.

10.2.4. Source Code

Voor het opleveren van de sourcecode heb ik de sourcecode doorgenomen met mijn begeleider. Hierdoor was mijn code niet alleen door mijn commentaar begrijpbaar maar heb ik hem waar nodig ook de nodige toelichting gegeven. Hierdoor worden er geen problemen verwacht als de Knalkey in een later stadium geïntegreerd wordt in Calkey.

10.3. Geboekte leerwinst

De opleiding Informatica en Informatiekunde van de Haagse Hogeschool heeft mij degelijke basis kennis verschaft welke ik gebruikt heb voor het uitstippelen van het te doorlopen traject. Ook voor het doorlopen van het traject zelf heb ik gebruik gemaakt van de kennis die ik op school heb opgedaan. Voor het afstuderen had ik niet het idee dat ik op school veel had geleerd, nu weet ik dat ik op school veel, ook in de praktijk, bruikbare vakken heb gevolgd.

Buiten, voor de ICT minder belangrijke dingen als een kostenbegroting heb ook een aantal zaken geleerd die wel voor belangrijk zijn:

- Vergroten van de kennis van het ontwikkelen in de ontwikkelomgeving Delhi 5.0
- Toepassen van een ontwikkelmethode in de praktijk.
-

10.4. Conclusie

Bij ALFA Development heb ik een leuke tijd gehad. Ik heb met plezier de opdracht uitgevoerd. Ook het feit dat de Knalkey later geïntegreerd wordt in Calkey en dat de Calkey later daadwerkelijk op de markt komt vind ik leuk. In het verleden heb ik namelijk alleen interne applicaties geschreven. Ook het ontwikkelen in de praktijk heb ik als prettig ervaren. Hieruit blijkt dat lang niet iedereen applicaties (op de zelfde manier) ontwikkeld. Hierbij moet wel worden uitgekeken dat onmisbare stappen niet worden overgeslagen. De gedachte hierachter is wel goed namelijk het komen tot een goed resultaat en niet de ‘regelgeving’ die bij een ontwikkelmethode hoort.

Ik ben blij dat de conversietool functioneel is en ik ben, samen met mijn opdrachtgevers, tevreden over het gemaakte werk.

Ik vind het jammer dat mijn contract niet verlengt kan worden bij ALFA Development, ik zou er graag willen blijven werken.

11. Geraadpleegde literatuur

Boeken:

- Jos Warmer & Anneke Kleppe, Praktisch UML.
- Misset, Een selectie uit Misset´s bouwkosten voor het bouwkundig onderwijs
- Marco Cantu, Mastering Delphi

Handleiding

- Handleiding van het begrotingsprogramma Calkey
- Handleiding van het begrotingsprogramma de Knaakbaak
- De help functie van Delphi 5.0
- De help functie van Calkey

Website

- http://www.cs.ru.nl/~gerp/B3/Dictaat/

12. Woordenlijst

Bijlage A: De opdrachtomschrijving en het plan van aanpak

Plan van aanpak en opdrachtsomschrijving

Voorwoord

Dit document is bedoeld om de lezer een indruk te geven van het te doorlopen traject bij het uitvoeren van de opdracht ‘Conversie van de Knaakbaak naar de Calkey’. De heer Woolderink is in deze de opdrachtgever en de contactpersoon. Hij zal ook voor de begeleiding vanuit ALFA Development zorgen.

De doelgroep van dit document zijn de personen die betrokken zijn bij de conversie van de begrotingsprogramma’s, en heeft als doel deze te informeren over het te verlopen project.

Woerden,

J. Beukers
Student aan de Haagse Hogeschool.

1. Inleiding

Het Digitale Huis beheert de samenwerking van CAD-applicaties met bestekverwerkers, toetsingsprogrammatuur, documentbeheerssysteem en een calculatieprogramma. Het Digitale Huis heeft als doel om te komen tot een centraal digitaal gebouwmodel. Als bron wordt een centrale database voor bouwdelen en producten gebruikt, die speciaal afgestemd is op én het centrale model
én de verschillende applicaties. Uitgangspunt is niet punt nul, maar 3 CAD-applicaties op AutoCAD
(ALFA, DeltaPI en ARKEY). De organisaties van de in 2002 in Nederland leidende CAD-applicaties
besloten in 2002 tot het ontwikkelen van het Digitale Huis.
Bij ALFA Development is een calculatieprogramma 'de Calkey' ontwikkeld met een zeer snelle rekenkernel. Binnen het samenwerkingsverband Het Digitale Huis waar in ALFA participeert, wordt ook een ouder calculatieprogramma 'de Knaakbaak' gebruikt met een trage (VB) rekenkernel maar met een uitstekend 'moederbestand' voor bouwkundige kostencalculatie.

Dit document bevat de opdrachtomschrijving en het plan van aanpak bij het project 'het conversiesysteem de Knalkey'. In de opdrachtomschrijving staat beschreven wat de opdracht inhoud. Hier wordt ingegaan op wat de 'de Knalkey' moet kunnen. In het plan van aanpak staat beschreven hoe deze opdracht zal worden gaan uitgevoerd. Ook worden hier onderwerpen als planning, risico's en kosten behandeld.

2. Opdrachtomschrijving

2.1. Inleiding

Om een indruk te krijgen van het project wordt eerst de opdracht in detail uitgewerkt. Er wordt gekeken naar het probleem, de eisen en wensen en de op te leveren producten. Wat is de huidige situatie, wat is de te verwachte eindsituatie en hoe tot deze eindsituatie te komen.

2.2. Probleemstelling

De huidige rekenkernel is te traag. De datastructuur en werkwijze van beide programma's zijn duidelijk verschillend. Binnen de Calkey wordt gebruik gemaakt van een referentie model waar in kosten voor een bepaald bouwdeel, zoals een bepaald type wand, éénmalig vastgelegd in een definitie van het bouwdeel. Bij elke instantie van dat bouwdeel in de calculatie wordt gerefereerd naar die definitie. Binnen de Knaakbaak kunnen bouwdelen met dezelfde codering en omschrijving verschillend afgeprijsd worden. Bouwdelen kunnen in de begroting gekopieerd worden zonder controle op een unieke definitie van de prijsopbouw. Er kan dus niet vlekkeloos overgeschakeld worden naar de nieuwe en snellere rekenkernel de Calkey.

2.3. Doelstelling van de opdracht

Het is de bedoeling het Knaakbaak basis bestand met bouwkosten definities te converteren naar de Calkey bestandsstructuur. Bij een conversie zal een controle gedaan moeten worden op verschillende kostenonderbouwingen van bouwdelen met dezelfde codering en omschrijving. Hierbij moet de gehele boomstructuur van de calculatie doorlopen worden. Bij discrepanties moet een signalering volgen in de vorm van een rapportage.

2.4. Uitgangssituatie

De Knaakbaak kostenbestanden bestaan uit een MS Access database waar in de verschillende tabellen en sommige onderlinge relaties zijn vastgelegd. Calkey is een Delphi applicatie waarbij
de records waar in de kosten worden vastgelegd in een eigen bestandsstructuur worden opgeslagen.

2.5. Benodigde software

Voor het uitvoeren van de opdracht is de volgende software benodigd:

- MS Access voor het analyseren van de knaakbaak structuur
- MS Visio en MS Word voor het modelleren van de structuren en uitwerken van diagrammen
- Delphi 5.0 voor de realisatie van de conversietool
- MS Word voor de verslaggeving

2.6. Benodigde hardware

Voor het uitvoeren van de opdracht is de volgende hardware benodigd:

- Een werkstation
- Een Printer
- Server voor de Back-ups

2.7. Beschikbare rapporten

Voor het uitvoeren van de opdracht zijn de volgende rapporten benodigd:

- Documentatie van het calculatieprogramma de Calkey voor het achterhalen van de structuur
- Sourcecode van het calculatieprogramma de Calkey om gebruik te kunnen maken de functies

2.8. Concrete werkzaamheden

Het schrijven van een Delphi applicatie die de MS Access database van de Knaakbaak inleest, controleert en converteert en uiteindelijk het resultaat wegschrijft in het bestandsformaat van Calkey.

2.9. Uit te voeren activiteiten

- Opstellen plan van aanpak
- Uitvoeren definitiestudie.
    - Onderzoeken huidige situatie, namelijk de structuur van de Knaakbaak en de Calkey.
    - Opstellen eisen en de wensen.
    - Opstellen systeemconcept en gevolgen.
- Ontwerp Conversietool
    - Detailleer systeemeisen en informatiebehoeften.
    - Ontwikkel mens- machineraakvlakken.
    - Specificeer procedures en formulieren.
    - Specificeer beeldscherm- en lijstindelingen.
    - Specificeer programmatuur.
    - Opstellen testplan.
    - Detailleer plan voor realisatie en invoering.
- Realisatie Conversietool.
- Testen Conversietool.
- Schrijven korte handleiding
- Schrijven afstudeerverslag

2.10. Resultaten voor de opdrachtgever

Aan het eind van het traject zullen de volgende producten opgeleverd zijn:

- Plan van aanpak
- Definitiestudie
- Ontwerp van conversietool
- Sourcecode van conversietool
- De handleiding van conversietool

3. Plan van aanpak

3.1. Inleiding

Het plan van aanpak kan worden gezien als een handleiding voor uitvoeren van het project. Dit hoofdstuk gaat in op de omgeving van de projectgroep, de fasering, de methodes en technieken, het bouwen, het invoeren en het testen van de applicatie.

3.2. Betrokkenheid

3.2.1. Gebruiker(s) / opdrachtgever

De heer R. Woolderink is een medewerker van ALFA Development en tevens de ontwikkelaar van de Calkey. Hij zal voor dit project de opdrachtgever zijn waar voor vragen over de functionaliteit van de conversietool, structuur van de Calkey, enz. aangeklopt kan worden. Hij zal ook de voornaamste gebruiker zijn van de conversietool.

De communicatie zal voornamelijk plaatsvinden door middel van gesprekken op de werkvloer. Als de heer R. Woolderink niet aanwezig is, is er altijd de mogelijkheid om telefonisch of via de e-mail in contact te komen.

3.2.2. De bedrijfsmentor

De heer R. Woolderink is tevens de bedrijfsmentor. Deze persoon is het eerste contact voor vragen en opmerkingen vanuit de projectgroep. Voor de bedrijfsmentor geldt een informeel contact.

3.2.3. De projectgroep

Gedurende dit project bestaat de projectgroep slechts uit een medewerker, namelijk de heer J. Beukers. Deze persoon is een student aan de Haagse Hogeschool, en voert deze opdracht individueel uit ten behoeven van zijn afstudeerstage. De student is verantwoordelijk voor de kwaliteit en kwantiteit van zijn werk. Voor het project is een vijftiental werkweken uitgetrokken á 38 uur per week. Gedurende deze periode zal hij het project afhandelen. De planning hiervan is opgenomen in hoofdstuk 3.5 met de titel ‘Fasering’.

Het projectlid heeft de juiste vakkennis om een groot aantal beslissingen zelf te maken, waar een beslissing grote veranderingen met zich teweeg brengt kan deze terugvallen op de mentor / opdrachtgever.

3.3. risicofactoren

Bij het uitvoeren van het project dient er rekening gehouden te worden met de volgende risicofactoren:

- De structuur van de database van de Knaakbaak is niet te achterhalen.

  In dit geval zouden de ontwikkelaars van de Knaakbaak opgespoord moeten worden om de structuur van de Knaakbaak door middel van hun kennis te achterhalen. Dit zou van beide partijen veel tijd kosten.

  Tegenmaatregel:

  Voor het achterhalen van de structuur moet er voldoende tijd worden gereserveerd zodat het project niet zal uitlopen.

- Het project zou meer tijd in beslag nemen dan gepland / beschikbaar is.

Tegenmaatregel:

Door het inbouwen van een tijdmarge in de planning voor eventuele speling is dit op te lossen. Wanneer toch de grens van vijftien weken overschreden wordt, zal er naar een passende oplossing moeten worden gezocht.

- Het eindproduct niet naar wens van de opdrachtgever.

  Tegenmaatregel:

  Er moet zorg worden gedragen dat er duidelijkheid is richting de opdrachtgever. Dit begint al bij het goedkeuren van de opdrachtomschrijving. Hierin staat wat de daadwerkelijke opdracht is. De opdrachtgever mag hier alleen in overleg met de projectgroep van afwijken. De projectgroep mag daarop de planning aanpassen en zo ook de op te leveren producten.

  Door het tussentijds opleveren van deelproducten kunnen misverstanden vroegtijdig geconstateerd worden.

3.4. Standaards, procedures en richtlijnen

Voor de verslaggeving wordt gebruik gemaakt van Microsoft Office, en schema's en diagrammen worden in Microsoft Visio vervaardigd. Voor de opmaak wordt er gebruik gemaakt van dezelfde lay-out als in dit document.

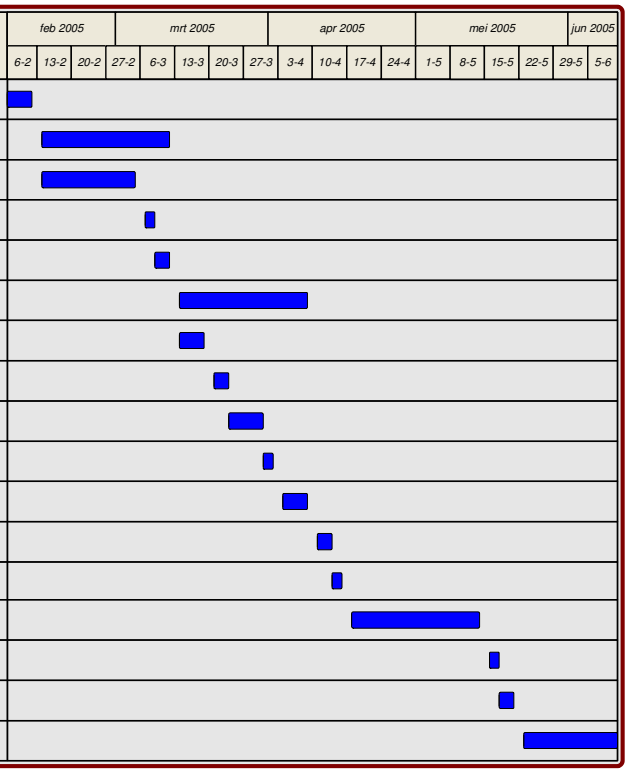
De sourcecode moet voldoen aan de volgende richtlijnen:

- Voorzien van voldoende commentaar.
- Er is gebruik gemaakt van duidelijke benamingen.

3.5. Fasering

In onderstaande tabel staan de fase die doorlopen moeten worden. Deze planning geeft aan welke stappen elkaar opvolgen, en hoeveel tijd er is ingepland voor elke stap.

ID	Taak	Start	Eind	Duur
1	Opstellen plan van aanpak	7-2-2005	11-2-2005	5d
2	Uitvoeren definitiestudie	14-2-2005	11-3-2005	20d
3	Onderzoeken huidige situatie namelijk de knaakbaak en de calkey	14-2-2005	4-3-2005	15d
4	Opstellen eisen en wensen	7-3-2005	8-3-2005	2d
5	Opstellen systeemconcepten en gevolgen	9-3-2005	11-3-2005	3d
6	Ontwerp conversietool	14-3-2005	8-4-2005	20d
7	Detailleer systeemeisen en informatiebehoeften	14-3-2005	18-3-2005	5d
8	Ontwikkel mens/machine raakvlakken	21-3-2005	23-3-2005	3d
9	Specificeer procedures en formulieren	24-3-2005	30-3-2005	5d
10	Specificeer beeldscherm en lijst indelingen	31-3-2005	1-4-2005	2d
11	Specificeer programmatuur	4-4-2005	8-4-2005	5d
12	Opstellen testplan	11-4-2005	13-4-2005	3d
13	Detailleer plan voor realisatie en invoering	14-4-2005	15-4-2005	2d
14	Realisatie conversietool	18-4-2005	13-5-2005	20d
15	Testen conversietool	16-5-2005	17-5-2005	2d
16	Schrijven korte handleiding	18-5-2005	20-5-2005	3d
17	Schrijven afstudeerverslag	23-5-2005	10-6-2005	15d



3.6. Aanpak definitiestudie

Tijdens de definitiestudie wordt de huidige situatie in kaart gebracht namelijk de structuur van de knaakbaak database. Vervolgens moeten de eisen en wensen van conversietool worden opgesteld. Met deze eisen en wensen kan uiteindelijk het systeemconcept met de gevolgen daarvan worden opgesteld.

3.7. Ontwerp van conversietool

De ontwerpfase gaat uit van het beschrijven van wat het systeem moet kunnen, hierin wordt nog niet ingegaan op hoe het gebeuren zal. Het is een vertaling van de opdrachtomschrijving naar overzichtelijke diagrammen. Voor het vertalen van de opdrachtomschrijving wordt de taal UML gebruikt. Ook het opstellen van het testplan is een onderdeel van de ontwerpfase.

3.8. Realisatie en testen

Het ontwerp zal daarna vertaald worden naar het daadwerkelijke programma. Dit zal gebeuren in Delphi 5.0. Na de realisatie zal de conversietool getest moeten worden aan de hand van het testplan beschreven in de ontwerpfase. Er zal gebruik worden gemaakt van ‘black box’ testen. Hierbij wordt de input vergeleken met de output om met die gegevens de werking van de conversietool te testen.

3.9. Oplevering

Uiteindelijk zal de conversietool opgeleverd moeten worden. Dit zal beginnen met het schrijven van de handleiding bij de conversietool en zal eindigen bij het daadwerkelijk overdragen van de conversietool met de daarbijbehorende documentatie.

3.10. Kwaliteitscontrole

Kwaliteit van het product zal gecontroleerd worden aan de hand van het ontwerp en de opdrachtomschrijving. Het ontwerp behoort alle functie eisen te omvatten die samengesteld zijn aan de hand van de opdrachtomschrijving en in overleg met de opdrachtgever. Het programma zal in eerste instantie moeten voldoen aan deze documenten.
Daarnaast is er de tevredenheid van de opdrachtgever en de uiteindelijke functionaliteit van het programma in de praktijk wat het programma succesvol maakt.

3.11. Kosten en baten analyse

De gebruiker zal van het programma alleen baat ondervinden als blijkt dat de gegevens zonder al te veel problemen over gezet kunnen worden naar het formaat van de Calkey. Hierdoor zouden dus de kostenbestanden worden gebruikt in de Calkey. Wanneer er teveel dubbele benamingen zijn zou het invoeren vertragen en dus de efficiëntie verlagen.

De kosten zijn 400 euro per maand voor een periode van 5 maanden hetwelk resulteert in een bedrag van 2000 euro. Hier zijn de uren die R. Woolderink aan het begeleiden en informatie verschaffen over de functionaliteiten van de Knalkey niet meegerekend. Hoeveel dit zal zijn is niet van tevoren te zeggen.

Bijlage B: Definitiestudie

Definitiestudie

1. Inleiding

Er zijn twee bouwkundige begrotingsprogramma's. Het ene programma heet de Knaakbaak en het andere programma heet Calkey. Dit laatste programma is nog in ontwikkeling en moet nog worden verkocht. Beide programma's gebruiken verschillende bestandsformaten om de gegevens op te slaan.

De Knaakbaak is inmiddels al een jaar op de markt en wordt uitgeleverd met een kostenbestand met duizenden prijzen van materialen, materieel, uurlonen en andere gegevens.

Het doel van deze definitiestudie is om te beoordelen of het ontwikkelen van een conversiesysteem mogelijk en zinvol is. Of het gewenste systeem in technische zin haalbaar is. Hiervoor zal eerst de huidige situatie beschreven moeten worden. Goed beschouwd is deze huidige situatie er nog niet. Er is namelijk nog geen conversie tussen de Knaakbaak en de Calkey. Hiervoor zullen de twee opslagstructuren van de begrotingsprogramma's als huidige situatie dienen zodat de complexiteit van de situatie aan het licht komt. Vervolgens worden er veranderingsbehoeften en eisen en wensen opgesteld zodat er alternatieven bedacht kunnen worden. Tot slot volgt dan de keuze van deze alternatieven.

2. Huidige situatie

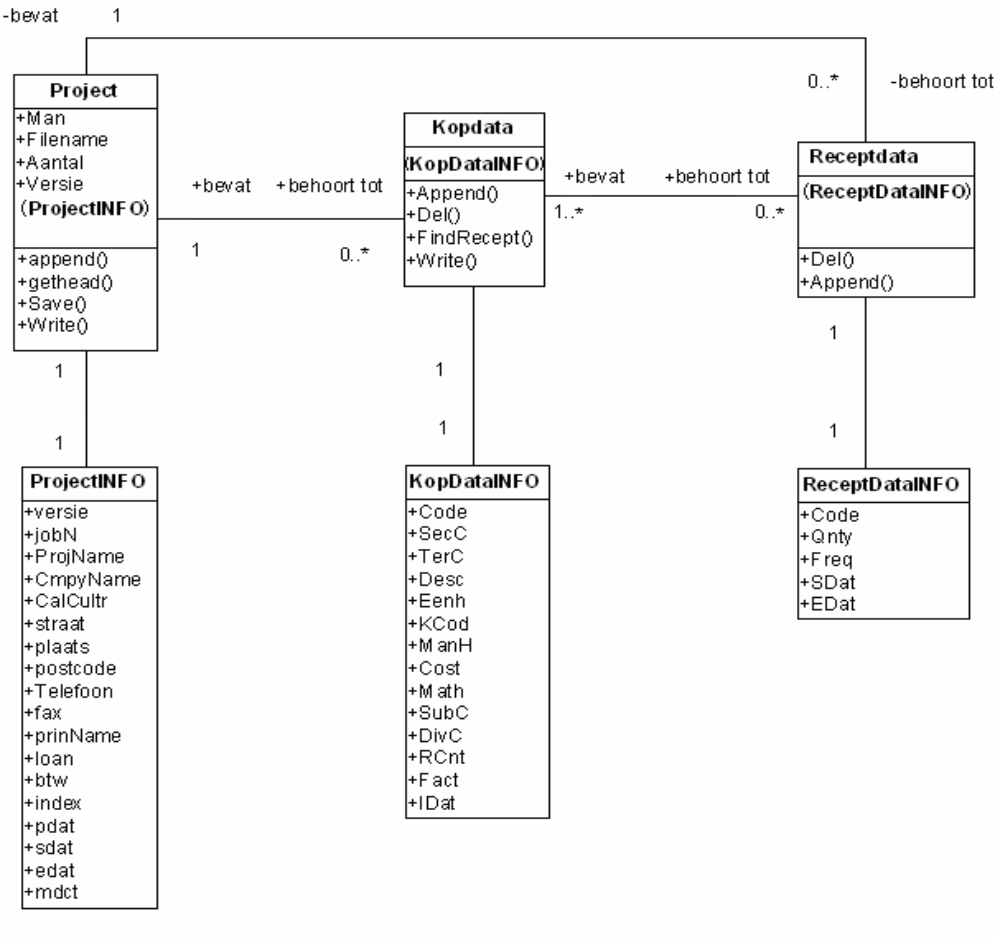
2.1. Inleiding

Allereerst zal de huidige situatie in kaart moeten worden gebracht. *De Knaakbaak* heeft een trage rekenkernel. Om dit probleem op te lossen is er al een nieuwe applicatie gemaakt met een snelle rekenkernel. Dit programma heet *Calkey*. Deze nieuwe rekenkernel gebruikt een andere bestandsstructuur voor het opslaan van de gegevens. Hierdoor kunnen de bestanden van de Knaakbaak niet direct in Calkey worden ingelezen.

De huidige situatie is dus eigenlijk dat de Knaakbaak gegevens niet zonder conversie in het nieuwere Calkey begrotingsprogramma kunnen worden ingelezen. Om de complexiteit van een eventuele oplossing in kaart te kunnen brengen is het zaak om de werking van de Knaakbaak en de Calkey enigszins toe te lichten.

2.2. Structuur van Calkey

Calkey is een objectgeoriënteerd programma. Het maakt dus gebruik van klassen en is in Delphi 5.0 ontwikkeld. De klassen die worden gebruikt voor het maken van de begroting worden in het diagram in afbeelding 1 weergegeven:

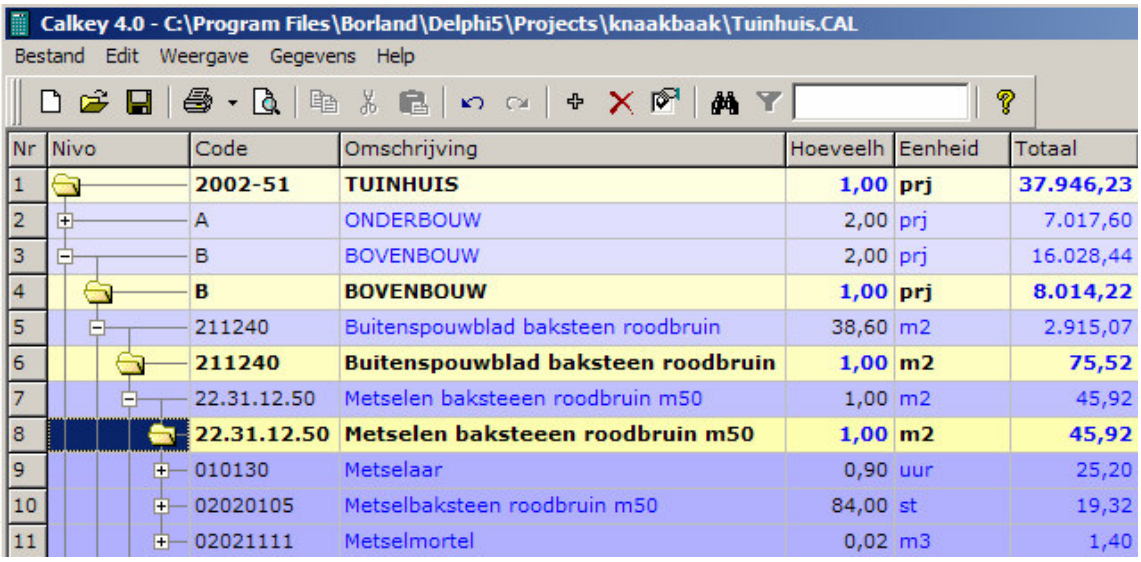


afbeelding 1

Bij de klassen *Project*, *Kopdata* en *Receptdata* behoren respectievelijk de records *ProjectInfo*, *KopdataInfo* en *ReceptdataInfo*.

In de Calkey wordt het attribuut *Code* van het record *KopdataInfo* gebruikt als primary key bij het wegschrijven van de gegevens. Bij het inlezen van de gegevens wordt de attribuut *Code* weer gebruikt om de *ReceptdataInfo* aan de juiste *KopdataInfo* te kunnen koppelen. Om deze reden dient *Code* uniek te zijn.

Met behulp van het diagram in afbeelding 1 is het mogelijk om de volgende structuur op te bouwen:



Nr	Nivo	Code	Omschrijving	Hoeveelh	Eenheid	Totaal
1		**2002-51**	**TUINHUIS**	**1,00**	**prj**	**37.946,23**
2		A	ONDERBOUW	2,00	prj	7.017,60
3		B	BOVENBOUW	2,00	prj	16.028,44
4		**B**	**BOVENBOUW**	**1,00**	**prj**	**8.014,22**
5		211240	Buitenspouwblad baksteen roodbruin	38,60	m2	2.915,07
6		**211240**	**Buitenspouwblad baksteen roodbruin**	**1,00**	**m2**	**75,52**
7		22.31.12.50	Metselen baksteeen roodbruin m50	1,00	m2	45,92
8		**22.31.12.50**	**Metselen baksteeen roodbruin m50**	**1,00**	**m2**	**45,92**
9		010130	Metselaar	0,90	uur	25,20
10		02020105	Metselbaksteen roodbruin m50	84,00	st	19,32
11		02021111	Metselmortel	0,02	m3	1,40

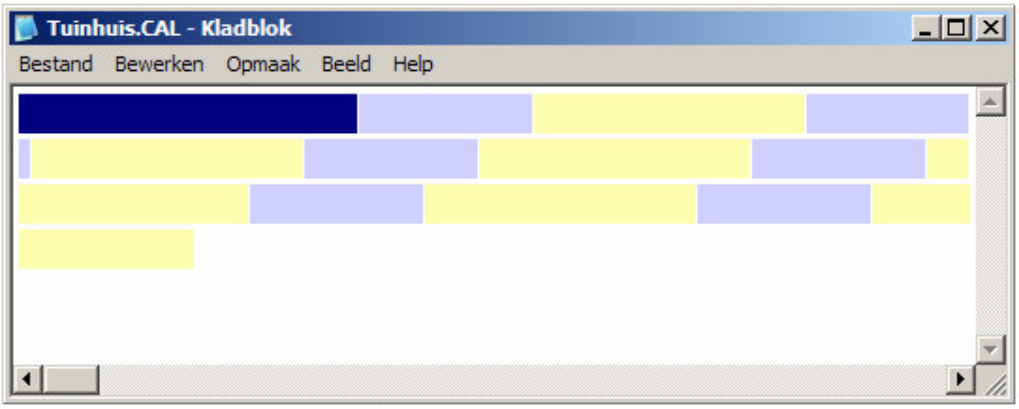
afbeelding 2

Het nieuwe programma Calkey maakt gebruik van Recept en Kopregels. Kopregels zijn recordregels die de prijs per eenheid is vastgelegd. De hoeveelheid van deze regels is altijd 1. Receptregels worden gebruikt om de hoeveelheden aan te kunnen geven.
In afbeelding 2 worden de kopregels met gele en receptregels met een blauwe kleur weergegeven. In het project *Tuinhuis* wordt de regel *Bovenbouw* twee keer weergegeven. De bovenste regel is een receptregel en de onderste een kopregel.

2.3. Opslagstructuur Calkey

Calkey gebruikt een eigen opslagstructuur voor het wegschrijven van de projectgegevens. Dit zijn bestanden met de extensie .cal. Dit zijn de eerste drie letters van de naam Calkey.

Calkey schrijft de gegevens per record weg naar dit type bestand. Bij het uitlezen van het bestand wordt er per record van uitgegaan dat de blokstructuur onveranderd is. Als dit niet het geval is dan kunnen de gegevens niet op de juiste manier uit het bestand worden uitgelezen. Op de volgende pagina wordt de opslagstructuur van de Calkey schematisch weergegeven.

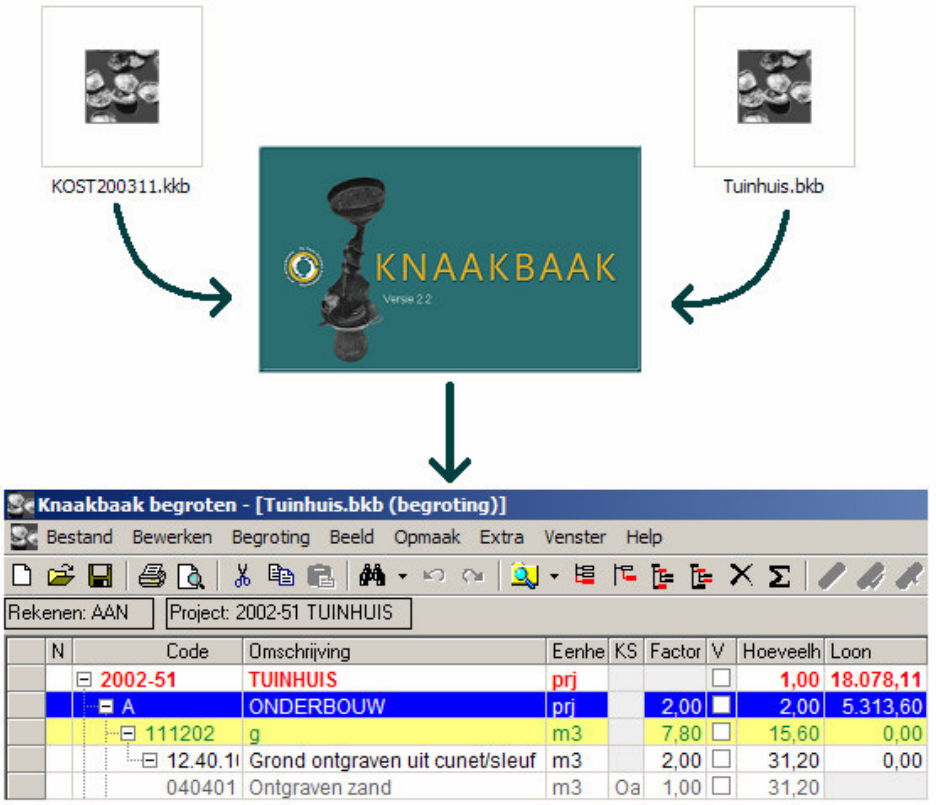


= Projectgegevens = Kopgegevens = Receptgegevens

Hierdoor is het dus zaak dat de sourcecode van de Calkey beschikbaar is.

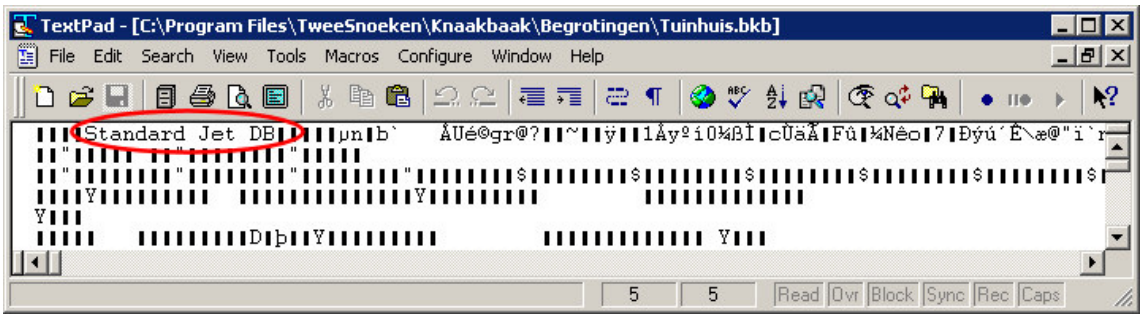
2.4. Werking van de Knaakbaak

De Knaakbaak slaat de projectgegevens op in een bestand met de extensie .bkb en gebruikt voor het kostenbestand de extensie .kkb. De Knaakbaak bevat functies om gegevens uit het kostenbestand naar het projectbestand over te hevelen. Hierdoor kan door selectie van een project, gebouwdeel, bouwdeel, begrotingspost of een middel, inclusief de onderliggende items worden overgeheveld. Dit wordt in het schema van afbeelding 4 schematisch weergegeven:



2.5. Databases van de knaakbaak

De Knaakbaak maakt, zoals hierboven beschreven, gebruik van twee verschillende bestanden namelijk het projectbestand en het kostenbestand. Om erachter te komen wat voor bestandsformaat de Knaakbaak gebruikt, zijn de knaakbaak bestanden in een ACSII tekstverwerker geopend.



Afbeelding 5

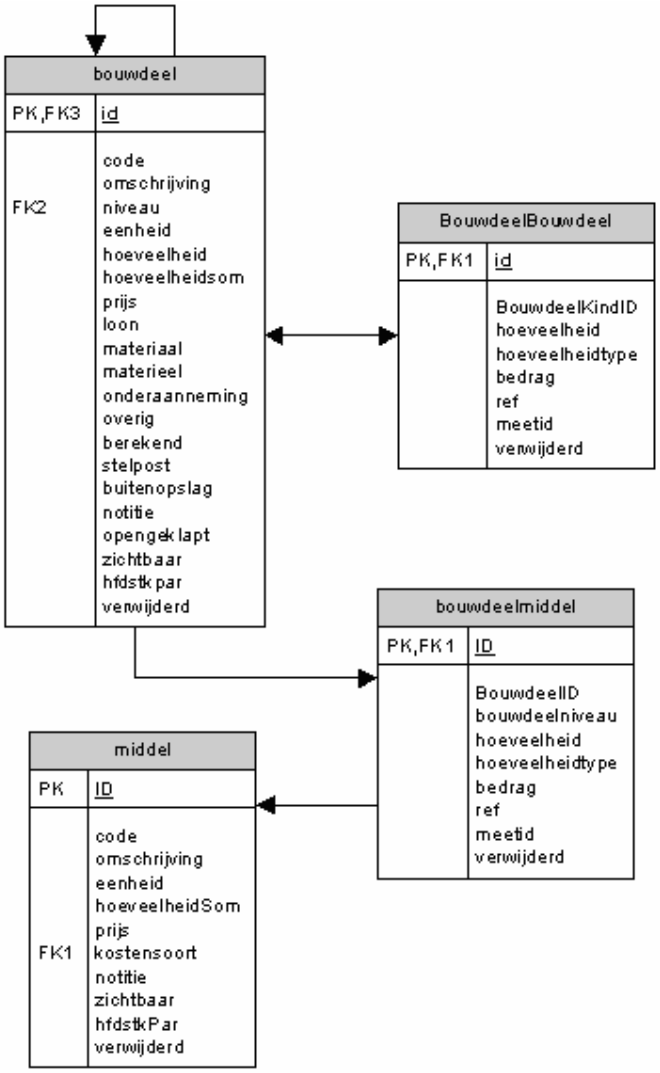
De bestanden zijn aangemaakt met de Standaard Jet DB (zie afbeelding 5). Uit nader onderzoek bleek dat de bestanden MS Access bestanden zijn. Door de extensie van de knaakbaak bestanden te veranderen naar de extensie van MS Access, namelijk *.mdb. bleek het mogelijk te zijn deze bestanden in MS Access te openen.

2.6. Structuur van het projectbestand

De gegevens van de projectbestanden van de Knaakbaak staan in de database allemaal in één tabel namelijk ‘Bestand’. De gegevens worden aan elkaar gekoppeld door middel van het veld ‘Id’ en ‘VaderId’.

2.7. Structuur van het kostenbestand

Na analyse van het Knaakbaak kostenbestand, worden in afbeelding 6 zijn de belangrijkste tabellen weergegeven. De overige tabellen zijn weggelaten omdat deze of niet gevuld zijn of omdat deze tabellen niet relevant zijn voor de conversie naar Calkey. Een voorbeeld hiervan zijn de opmaakgegevens van de knaakbaak.



afbeelding 6

De relatie bouwdeel -> bouwdeel bestaat niet uit een relatie met sleutels maar uit de opbouw van de code. Het bouwdeel met de code *1* is de vader van het bouwdeel met de code *1.10*. Het bouwdeel met de code *1.10* is dus ook weer de vader van het bouwdeel met de code *1.10.111* maar ook van het bouwdeel met de code *1.10.123*. Op deze manier kan er dus een boom opgebouwd bestaand uit bouwdelen.

De bouwdeel typen zijn onder te verdelen in projecten, gebouwdelen, bouwdelen en begrotingsposten.

Bij bouwdelen heeft het attribuut *niveau* de waarde *2* en de bij begrotingsposten heeft het attribuut de waarde *3*. De begrotingsposten zijn doormiddel van de tabel *bouwdeelbouwdeel* gekoppeld aan de bouwdelen. In de tabel *bouwdeelbouwdeel* staat ook de hoeveelheid vermeld van de betreffende begrotingspost die bij een bouwdeel hoort.

De middelen, behorend op laatste niveau, zijn gekoppeld door de tabel *BouwdeelMiddel* met het veld *BouwdeelID*. Het veld *MiddelID* geeft aan welk middel bij de begrotingspost hoort. Ook de hoeveelheid van het te gebruiken middel is in deze tabel terug te vinden.

3. Veranderingsbehoeften en systeemeisen

3.1. Inleiding

Om tot een duidelijke lijst van systeem eisen te komen is het belangrijk om naar de veranderingsbehoeften van de huidige situatie te kijken. De vragen die hierbij kunnen worden gesteld zijn: *Waarom is de huidige situatie niet meer afdoende?* en *Wat moet er in de nieuwe situatie wel kunnen?*.

3.2. Veranderingsbehoeften

Op de vraag: *Waarom is de huidige situatie niet meer afdoende?* is het volgende antwoord te geven: De gegevens van de Knaakbaak zijn niet te gebruiken in de Calkey. Dit neemt de volgende gevolgen met zich mee:

- De kostenbestanden zijn niet te gebruiken in de Calkey waardoor het niet mogelijk is om bouwelementen gebruikt in de Knaakbaak te gebruiken in de Calkey.
- De projectbestanden van de Knaakbaak zijn niet te openen met de Calkey.

3.3. Systeemeisen

Om ervoor te zorgen dat de veranderingsbehoeften vervuld worden, dus dat de gegevens in de Calkey gebruikt kunnen worden zijn de volgende systeemeisen opgesteld:

- Het systeem moet de gegevens van het Knaakbaak formaat converteren naar het Calkey formaat.
- Het systeem moet een dusdanige performance hebben zodat het converteren van de gegevens van de trage rekenkernel naar de snelle rekenkernel niet drempelgevoelig is.
- Het systeem moet discrepanties weer kunnen geven in de vorm van een rapportage.

4. Opstellen systeemconcept en gevolgen

4.1. Inleiding

In dit gedeelte worden er een aantal systeemconcepten met gevolgen weergegeven zodat uit deze systeemconcepten een keuze kan worden gemaakt. Hierdoor is de kans kleiner dat er voor een ingewikkelde oplossing wordt gekozen, terwijl er een minder ingewikkelde voor handen ligt.

4.2. Alternatief 1

Het eerste alternatief wat wordt aangedragen is een conversietool die de gegevens converteert door middel van een 'tussenstructuur'. Hier worden de gegevens eerst overgezet van de Knaakbaak naar een losse structuur, vervolgens kunnen in deze structuur wijzigingen worden aangebracht. Vanuit deze structuur worden de gegeven in de Calkey structuur gezet.

Het voordeel van deze aanpak is dat indien de structuur van de Knaakbaak of Calkey veranderd dat alleen de vertaling naar de tussenstructuur herschreven dient te worden. Indien de structuur in de toekomst ook nog overgezet moet worden naar een derde applicatie kan er overgezet worden vanuit deze tussenstructuur.

Het nadeel is dat er in dit geval twee verschillende conversies uitgevoerd moeten worden.

4.3. Alternatief 2

Het tweede alternatief is het direct converteren van de Knaakbaakstructuur naar de Calkeystructuur. Hier worden de gegevens direct vanuit de Knaakbaakstructuur in de Calkeystructuur gezet. Eventuele wijzigingen in de structuur kunnen alleen tijdens dit proces uitgevoerd worden.

Het voordeel is dat er maar één conversie benodigd is bij het overzetten van de gegevens. Mocht echter 1 structuur veranderen dient de gehele conversietool aangepast te worden.

4.4. Alternatief 3

Het derde alternatief is het aanpassen van de structuur van Calkey. Aangezien de structuur van Calkey nog niet vast ligt is het mogelijk om de structuur van Calkey aan te passen op de structuur van de Knaakbaak. Hierdoor zou een conversie overbodig worden en dus de kostenbestanden van de Knaakbaak altijd in Calkey in te lezen zijn. Het nadeel hiervan is dat de eigenschappen van de structuur van de Knaakbaak ook worden meegenomen in Calkey. Dit zou Calkey dus vertragen en hierdoor zou ook de ingewikkelde structuur meegenomen worden in Calkey.

4.5. Alternatief 4

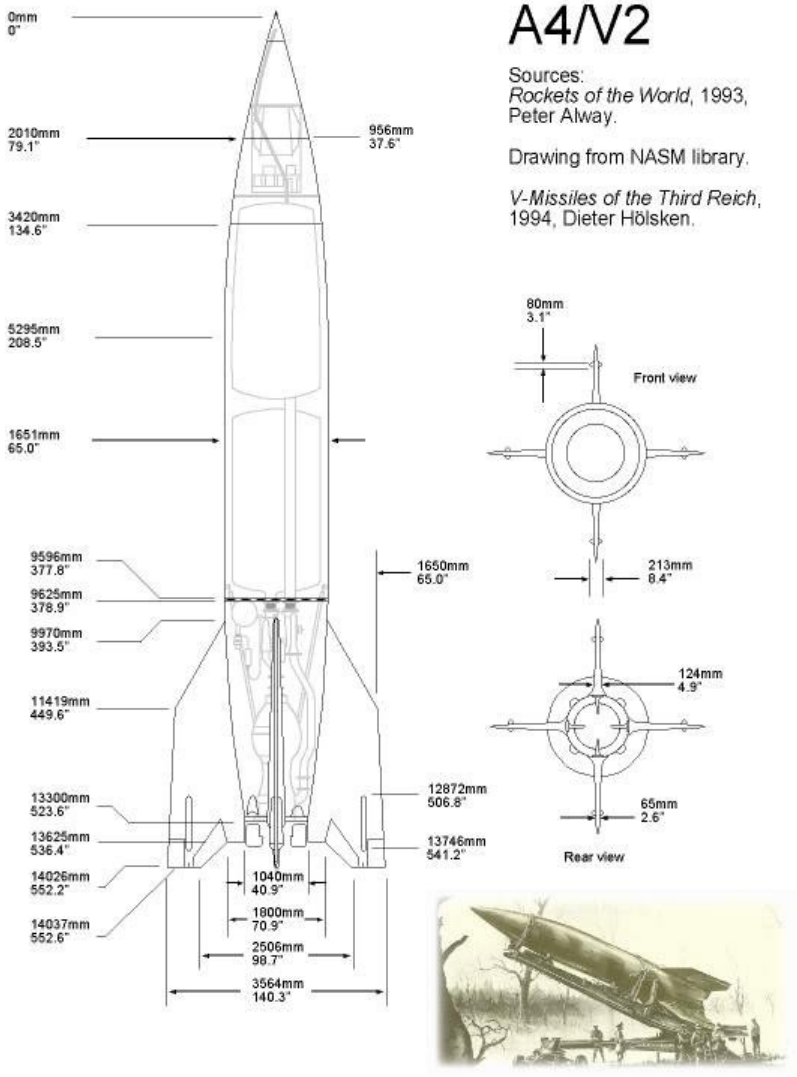
Het vierde en laatste alternatief is het handmatig overtypen van de gegevens. Hier wordt er een tijdwinst geboekt omdat er dan geen applicatie ontwikkeld hoeft te worden en er kan ingespeeld worden op een wijziging van de structuur. Het nadeel hiervan is dat de persoon die de gegevens overzet de kennis moet hebben van de structuur van de Knaakbaak en de structuur van Calkey. Daar komt bij dat de kostenbestanden duizenden bouwelementen bevatten, het overzetten kost dus veel tijd.

4.6. Keuze

Om het half jaar komt er een update van het kostenbestand uit. Een automatisch oplossing is dus gewenst zodat er niet elk half jaar veel tijd besteed hoeft te worden aan de conversie. De structuren van beide begrotingsprogramma's zijn statisch en er zijn zover het er nu naar uit ziet geen conversiebehoeften naar derde applicaties. De Calkey is ontwikkeld omdat de Knaakbaak te traag werkte. Hierdoor is het geen optie om de structuur van Calkey aan te passen op die van de Knaakbaak. Hierdoor zou ook Calkey traag worden. Om deze redenen is er gekozen voor het schrijven van een conversietool die geen gebruik maakt van een tussenstructuur. Deze conversietool krijgt de naam 'de Knalkey', een samenvoeging van de Knaakbaak en Calkey.

Bijlage C: Ontwerp

Ontwerp KnalKey



Inhoudsopgave

1 Inleiding

Dit document zal het ontwerp van de Knalkey beschrijven. In dit ontwerp zal naar voren komen wat de Knalkey precies moet kunnen en hoe dit gerealiseerd gaat worden. Door middel van use cases, klassendiagrammen en sequence diagrammen zal er aan het licht komen wat de gedetailleerde systeem eisen zijn, de mens en machineraakvlakken worden weergegeven. Tot slot worden de beeldschermen en lijstindelingen weergegeven en wordt de programmatuur gespecificeerd.

2 Use Cases

2.1 Inleiding

In dit gedeelte worden de use cases beschreven. De systeemeisen eerder genoemd in de definitiestudie zullen hier verder worden uitgewerkt. Er zal hierbij onderscheid gemaakt worden tussen functionele en niet functionele eisen. De functionele eisen zullen worden weergegeven in het use case diagram en in de use case templates zullen de use cases worden beschreven.

2.2 Systeemeisen

Hier zullen de systeemeisen verder worden uitgewerkt. Dit wordt gedaan om meer duidelijkheid te verkrijgen over wat het systeem dient te kunnen. Gegeven zijn de systeemeisen zoals eerder vermeld in de definitiestudie:

- Het systeem moet de gegevens van het Knaakbaak formaat converteren naar het Calkey formaat.
- Het systeem moet een dusdanige performance hebben zodat het converteren van de gegevens van de trage rekenkernel naar de snelle rekenkernel niet drempelgevoelig is.
- Het systeem moet discrepanties signaleren en weer kunnen geven in de vorm van een rapportage.

Deze eisen onder te verdelen in functionele en niet functionele eisen. Hieronder worden functionele eisen gedetailleerd weergegeven:

- Het systeem moet de gegevens van het Knaakbaak formaat converteren naar het Calkey formaat.
    - De gebruiker moet met het systeem Knaakbaak bestanden kunnen selecteren.
    - Het systeem moet de structuur van het geselecteerde bestand converteren naar de Calkey structuur.
    - Het systeem moet de structuur van de Calkey wegschrijven naar een bestand.
- Het systeem moet discrepanties weer kunnen geven in de vorm van een rapportage
    - Het systeem moet discrepanties kunnen signaleren
    - Het systeem moet de gesignaleerde discrepanties weer kunnen geven in de vorm van een rapportage.

De niet functionele eisen:

- Het systeem moet een dusdanige performance hebben zodat het converteren van de gegevens van de trage rekenkernel naar de snelle rekenkernel niet drempel gevoelig is.

2.2.1 Converteren

Het converteren van de Knaakbaak kan op een tweetal manieren. De structuur van de Calkey is namelijk zo dat er in het laagste niveau kosten van het materiaal, materieel, loon,

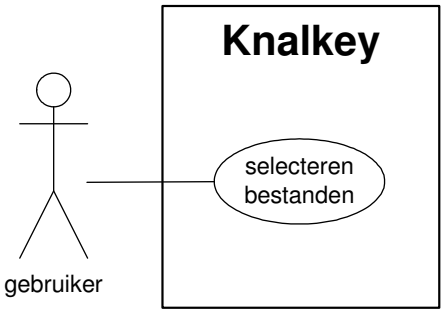
de kosten van de onderaanneming en de overige kosten kan worden opgenomen. Dit is in de Knaakbaak niet het geval. In de Knaakbaak staat het loon altijd apart vernoemd. De volgende manieren komen hieruit voort:

- De structuur van de Knaakbaak wordt aangehouden.
- Het laagste niveau van de structuur van de Knaakbaak wordt samengevoegd en in het niveau erboven gezet.

Het comprimeren van het laagste niveau heeft als voordeel dat het project (of kostenbestand) een plattere structuur krijgt. Het gevolg is echter wel dat de codes en de omschrijvingen die aan het laagste niveau waren gekoppeld verdwijnen.

2.3 Use Cases

Hier zullen de use cases van de Knalkey weer worden gegeven. In feite doet de gebruiker van de Knalkey maar één ding namelijk het selecteren van een bestand(en). Let wel dat er dan in een eerder stadia is aangegeven of het laagste niveau gecomprimeerd dient te worden. Dit is opgenomen en hieronder weergegeven in één use case.



De weergegeven use case beschreven in het onderstaande template:

Naam	Selecteren bestanden
Aannamen	De gebruiker heeft toegang tot de te converteren bestanden
Beschrijving	De gebruiker geeft aan of het laagste niveau van de knaakbaak gecomprimeerd dient te worden. Vervolgens selecteert de gebruiker één of meerdere Knaakbaakbestanden. Dit zijn bestanden met de extensie *.bkb (een project bestand) of *.kkb (een referentie- / kostenbestand).
Uitzonderingen	
Resultaat	De bestanden zijn geconverteerd en, indien aangegeven, het laagste niveau gecomprimeerd. De discrepanties worden door middel van een rapportage weergegeven.

3 Klassendiagram

3.1 Inleiding

In dit gedeelte wordt het klassendiagram van de Knalkey beschreven. De gebruikelijke methode is om het diagram op te stellen aan de hand van brainstormsessies, probleembeschrijvingen, enz. Deze methode is in deze situatie niet bruikbaar. Er bestaan namelijk al twee diagrammen waar rekening mee gehouden dient te worden. Dit zijn de tabellen van de Knaakbaak en het klassendiagram van de Calkey. Deze diagrammen zullen dan ook als uitgangspunt van het klassendiagram genomen worden. Dit diagram zal aangevuld worden aan de hand van de functionele systeemeisen, welke op hun beurt weer voorkomen uit de probleembeschrijving en brainstormsessies.

3.2 Knaakbaak en de Calkey

Om te kijken welke relaties er tussen deze diagrammen bestaan is het eerst zaak welke informatie er in de Calkey benodigd is. Vervolgens wordt er gekeken welke informatie er in de Knaakbaak aanwezig is. Indien de informatie aanwezig is dient de relatie gelegd te worden tussen de juiste Calkey klassen en de juiste Knaakbaak tabellen.

Aangezien het referentie bestand van de Knaakbaak een andere structuur heeft als het project bestand worden deze bestanden apart behandeld.

3.2.1 Structuur van de Calkey

De Calkey zet de projectgegevens runtime in het werkgeheugen. Hierbij worden er pointers gezet tussen de projecten, kopregels en receptregels. Dit is gedaan ten behoeve van de snelheid. In het geval van de Knalkey is dat niet nodig. Bij het wegschrijven van de structuur wordt namelijk alleen de code van de kopregels en receptregels en de volgorde (kopregels > receptregels) gebruikt voor het bij elkaar zoeken van de juiste kopregels en de juiste receptregels. In de Knalkey worden er geen bewerkingen gedaan op de kopregels en receptregels buiten het aanmaken, verwijderen en wegschrijven om. Hierdoor zal er geen tijdswinst maar tijdverlies worden geboekt indien er gebruik wordt gemaakt van pointers. Om deze reden worden de pointers van de recepten verwijzend naar de kopregels niet gebruikt.

Hieronder wordt de benodigde informatie van de Calkey per klasse of record, met een omschrijving weergegeven. Hierbij worden alleen de public attributen weergegeven, o.a. de niet gebruikte pointers worden weggelaten:

De klasse Kopdata:

Attribuut	Omschrijving
RPTR	Pointer naar de recepten die bij deze kopregel horen
Info	Pointer naar de informatie van de kopregel (KopdataInfo)

Het record KopdataInfo:

Attribuut	Omschrijving
Code	De code van het bouwdeel
SecC	De CADcode
TerC	Bestekscode
Desc	De omschrijving van een bouwdeel / middel
Eenh	De eenheid
Kcod	De code van een type manuur zodat het tarief bekend is
ManH	Het aantal manuren benodigd voor het bouwdeel / middel
Cost	De kosten benodigd voor het materiaal bij het bouwdeel / middel
MatC	De kosten benodigd voor het materieel bij het bouwdeel / middel
SubC	De kosten verbonden aan de onderaannemer
DivC	De kosten bestemd voor overige posten

Fact	De factor waarmee het geheel wordt vermenigvuldigd
Idat	De datum van invoering

De klasse Receptdata:

Attribuut	Omschrijving
Fnext	Pointer naar de volgende receptregel die bij dezelfde kopregel behoord
Info	Pointer naar de informatie van de receptregel (ReceptdataInfo)

Het record ReceptdataInfo:

Attribuut	Omschrijving
Code	De code van het bouwdeel
Qnty	De hoeveelheid benodigd voor een bouwdeel van een onderliggend item
Freq	De frequentie van het onderhoud
Sdat	De startdatum van een bouwdeel
Edat	De einddatum van een bouwdeel

De klasse Project:

Attribuut	Omschrijving
Frecept	Verwijzing naar receptdata
Fkop	Lijst met kopregels
Man	Lijst met manuurgegevens
Filename	Naam van het bestand
Aantal	Aantal kopregels in het project
Versie	Versienummer
Info	Pointer naar de informatie van het project (ProjectInfo)

Het record ProjectInfo:

Attribuut	Omschrijving
Versie	De versie van Calkey
Jobn	Werknummer
ProjName	Projectnaam
CmpyName	Bedrijfsnaam
Calcultr	Naam van de calculator
Straat	Straatnaam van het project
Plaats	Plaats van het project
Telefoon	Telefoonnummer van het project
Fax	Faxnummer van het project
Prinname	Opdrachtgever
Loan	Uurloon
BTW	BTW percentage
Index	Index op loonkosten
Pdat	Project startdatum, gewone calculatie
Sdat	Startdatum planmatig onderhoud
Edat	Einddatum planmatig onderhoud
MFct	Lijst afwijkende uurloonfactoren

De Calkey werkt volgens hetzelfde principe als primaire sleutels. De code van een kopregel is hierbij de primaire sleutel. Als receptregels de code van een kopregel in Receptdata zetten dan verwijzen ze naar de kopregel. Een kopdata weet aan de hand van het attribuut RPTR welke receptregels hij bezit. Het project houdt ook een lijst met pointers die verwijzen naar de kopregels.

3.2.2 De structuur van het Knaakbaak kostenbestand

Hier wordt bekeken of het vullen van de Calkey structuur door middel van het Knaakbaak referentiebestand mogelijk is.

Het doorlopen van de structuur van de Knaakbaak is mogelijk (zie definitiestudie). Er is namelijk bekend dat:

- De bouwdelen met de gebouwdelen gekoppeld zijn door middel van het veld Code.
- De begrotingsposten met de bouwdelen gekoppeld zijn door de tabel bouwdeelbouwdeel
- De middelen met de begrotingsposten gekoppeld zijn door de tabel bouwdeelmiddel

Het is dus mogelijk om de klassen Project, Kopdata en Receptdata in de Calkey structuur te vullen, waarbij de originele structuur van de Knaakbaak behouden blijft. Vervolgens moet er gekeken worden hoe de Inforecords gevuld dienen te worden. Hieronder worden de Inforecords nogmaals weergegeven en per attribuut wordt er weergegeven welke tabel en welk veld daarbij vergelijkbaar is in het Knaakbaakbestand. De attributen die niet in het referentiebestand zijn vertegenwoordigd worden weggelaten.

Het record KopdataInfo:

Attribuut	Tabel	Veld	Voorwaarden
Code	Bouwdeel	Code	
	Middel	Code	
Desc	Bouwdeel	Omschrijving	
	Middel	Omschrijving	
Eenh	Bouwdeel	Eenheid	
	Middel	Eenheid	
Kcod	Middel	Prijs *1	Kostensoort = 1
ManH	BouwdeelMiddel	Hoeveelheid	
Cost	Middel	Prijs	Kostensoort = 2
MatC	Middel	Prijs	Kostensoort = 3
SubC	Middel	Prijs	Kostensoort = 4
DivC	Middel	Prijs	Kostensoort = 5

*1 = het attribuut Kcod krijgt de code van het tarief (middel.veld). Als dit tarief niet aanwezig is dient het tarief toegevoegd te worden en krijgt Kcod de nieuwe code.

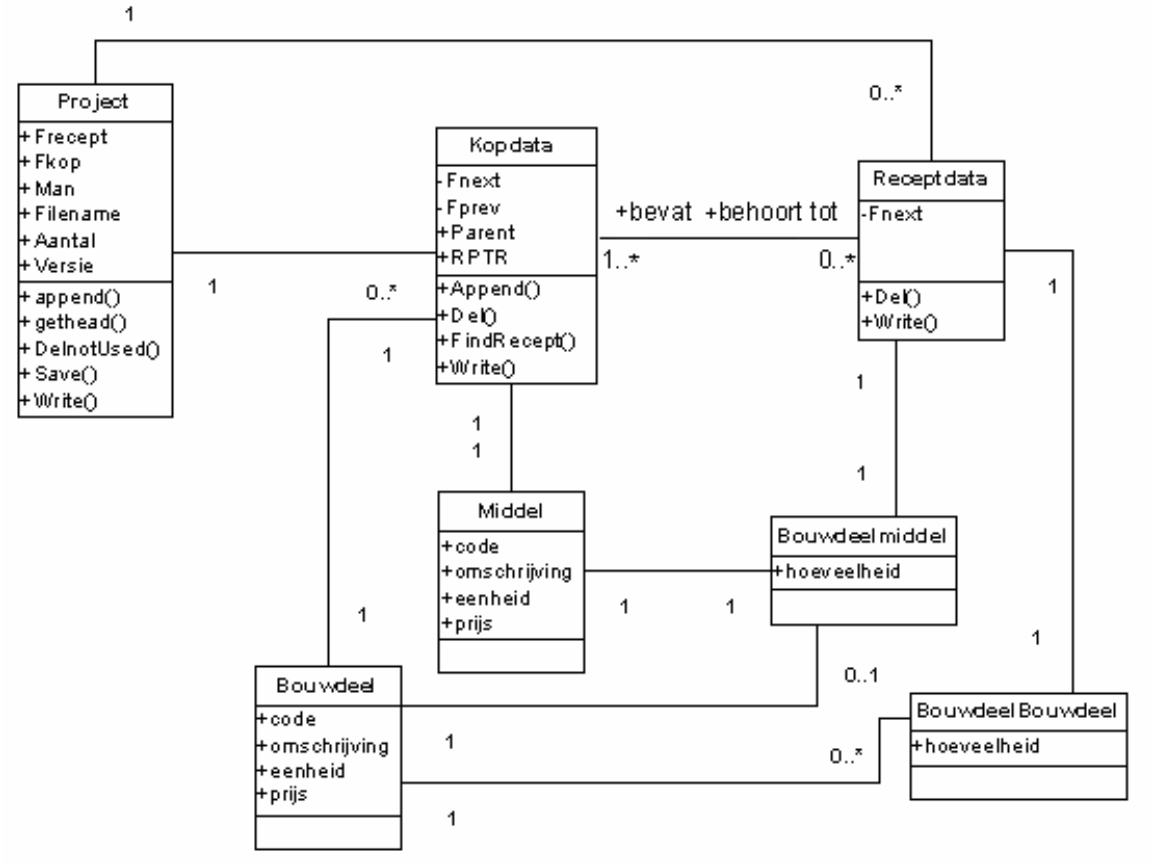
Het record ReceptdataInfo:

Attribuut	Tabel	Veld	Voorwaarden
Code	Bouwdeel	Code	
	Middel	Code	
Qnty	BouwdeelBouwdeel	Hoeveelheid	
	BouwdeelMiddel	Hoeveelheid	

In het referentiebestand van de Knaakbaak is er geen projectinfo beschikbaar zoals die in de Calkey benodigd is. Voorbeelden hiervan zijn het telefoonnummer, adres, plaats, enz. Hierom wordt de bestandsnaam van het referentiebestand bij het record ProjectInfo als projectnaam gebruikt.

3.2.3 Klassendiagram Knalkey kostenbestand

In afbeelding 1 is het klassendiagram van de Knalkey met betrekking tot het inlezen van het referentiebestand weergegeven. De tabbellen van de Knaakbaak zijn hierbij als klassen weergegeven. Alleen de velden die benodigd zijn voor de Calkey structuur zijn opgenomen in het diagram.



afbeelding 1

De Inforecords van de Calkey zijn weggelaten. Deze records worden namelijk altijd via de klassen *Project*, *Kopdata* en *Receptdata* benaderd en de relatie tussen deze klassen en de records is één op één. Het attribuut *prijs* bij de klassen Bouwdeel en Middel is het totaal van de velden Materiaal, Materieel, Loon, Onderaanneming en Overige. Alleen op het laagste niveau wordt gebruikt gemaakt van deze velden, in de andere gevallen worden deze velden samengesteld uit onderliggende waarden.

3.2.4 De structuur van het Knaakbaak projectbestand

Hier wordt bekeken of het vullen van de Calkey structuur door middel van het Knaakbaak projectbestand mogelijk is. Het doorlopen van de structuur van het Knaakbaak projectbestand is mogelijk (zie definitiestudie). Er is namelijk bekend dat de gegevens in de tabel bestand staan en zijn gekoppeld door de velden vaderID en ID. Vervolgens dient er gekeken te worden of mogelijk is om de Inforecords te vullen. Om deze reden zijn de Inforecords hier nog een keer weergegeven en is er aangegeven welke tabel en veld van het projectbestand per attribuut vergelijkbaar is.

Het Inforecord KopdataInfo:

Attribuut	Tabel	Veld	Voorwaarden
Code	Bestand	Code	
Desc	Bestand	Omschrijving	
Eenh	Bestand	Eenheid	
Kcod	Bestand	Prijs *1	Kostensoort = 1
ManH	Bestand	Hoeveelheid	
Cost	Bestand	Prijs	Kostensoort = 2
MatC	Bestand	Prijs	Kostensoort = 3
SubC	Bestand	Prijs	Kostensoort = 4
DivC	Bestand	Prijs	Kostensoort = 5

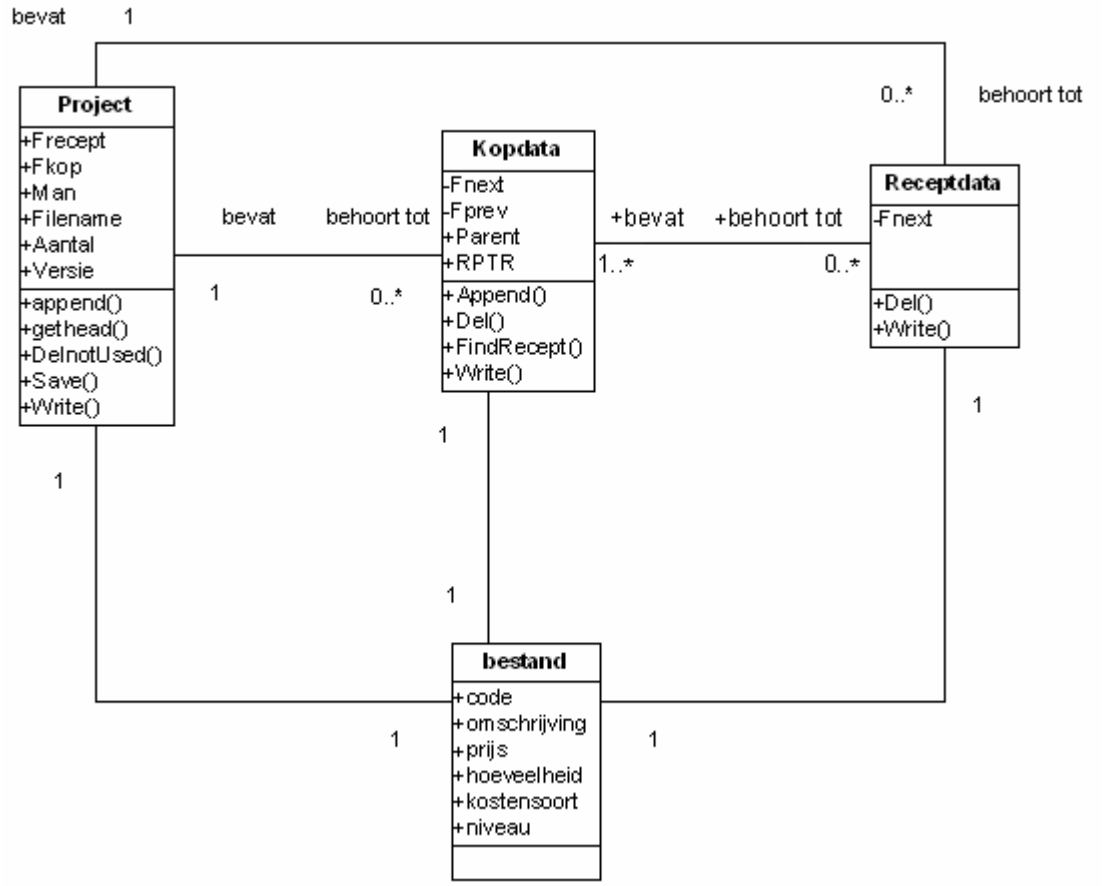
Het Inforecord Receptdata:

Attribuut	Tabel	Veld	Voorwaarden
Code	Bestand	Code	
Qnty	Bestand	Hoeveelheid	

De gegevens ten behoeve van de projecten die door de Knaakbaak bestanden worden opgeslagen zijn niet zo gedetailleerd als de projectgegevens in Calkey. Zo houdt de Knaakbaak van een project alleen de code (naam) en een omschrijving bij. Hierdoor blijven de projectgegevens in Calkey zoals telefoonnummer, adres, plaats, enz. niet ingevuld. De Knaakbaak projectgegevens staan ook in de tabel bestand. Dit record heeft bij het veld niveau de waarde 5.

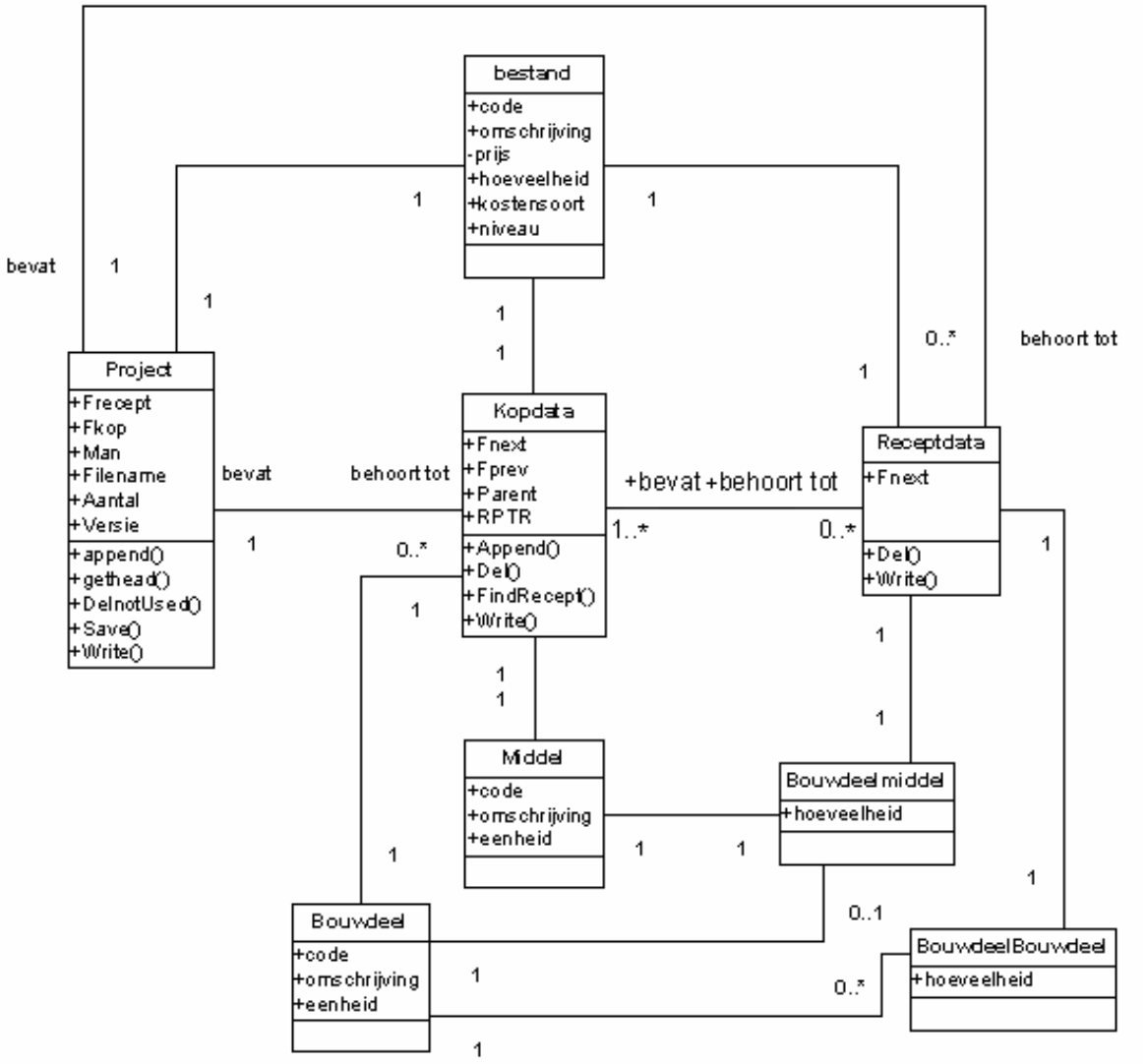
3.2.5 Klassendiagram Knalkey projectbestand

In afbeelding 1 is het klassendiagram van de Knalkey met betrekking tot het inlezen van het projectbestand weergegeven. De tabel *Bestand* van de Knaakbaak is hierbij als klasse weergegeven. Alleen de velden die benodigd zijn voor de Calkey structuur zijn opgenomen in het diagram.



3.2.6 Klassendiagram projectbestanden en kostenbestanden

In afbeelding 3 is het klassendiagram weergegeven dat voortgekomen is uit de samenvoeging van de twee voorgaande diagrammen. Ook hier zijn de tabellen van de Knaakbaak als klassen weergegeven.



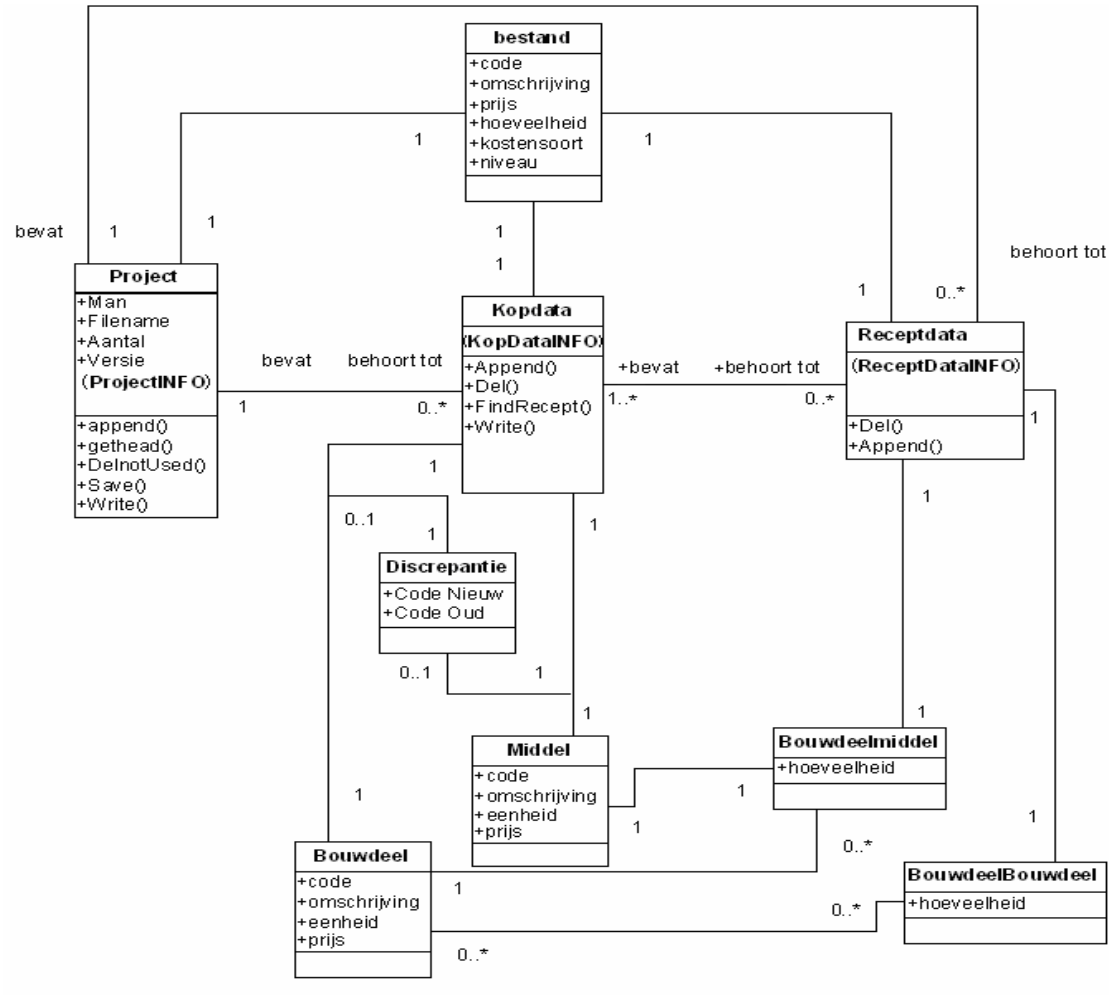
afbeelding 3: Klassendiagram kostenbestand en projectbestand

3.3 Eisen en wensen

In dit gedeelte worden de eisen en wensen geanalyseerd en mogelijkerwijs moeten de klassen die voortkomen uit de eisen en wensen toe worden gevoegd aan het klassendiagram. Dit zal alleen de functionele eisen en wensen betreffen, welke hieronder nog een keer worden weergegeven.

- Het systeem moet de gegevens van het Knaakbaak formaat converteren naar het Calkey formaat.
  - De gebruiker moet met het systeem Knaakbaak bestanden kunnen selecteren.
  - Het systeem moet de structuur van het geselecteerde bestand converteren naar de Calkey structuur.
  - Het systeem moet de structuur van de Calkey wegschrijven naar een bestand.
- Het systeem moet discrepanties weer kunnen geven in de vorm van een rapportage
  - Het systeem moet discrepanties kunnen signaleren.
  - Het systeem moet de gesignaleerde discrepanties weer kunnen geven in de vorm van een rapportage.

Als de gegevens van de Knaakbaak direct in de Calkey structuur worden gezet is er geen structuur benodigd voor het overhevelen van de gegevens. Wel moet er dan gekeken worden bij het inlezen of er discrepanties zijn. Als deze er zijn moet er een nieuwe code gegenereerd worden en in de structuur worden gezet. Tevens moet de oude en de nieuwe code worden opgeslagen zodat de gebruiker kan zien welke codes bouwelementen zijn gewijzigd. Om deze reden dient er dus nog een klasse bij te komen namelijk de klasse Discrepantie. Een Discrepantie kan dus optreden bij het overzetten van de middelen/ bouwdelen van de Knaakbaak naar de Kopdata (eigenlijk KopdataInfo) van de Calkey. De klasse behoort dus bij de relatie tussen middel en kopdata en tussen bouwmiddel en kopdata. Dit is als volgt weergegeven:



4 Sequence diagram

4.1 Inleiding

In dit gedeelte worden de sequence diagrammen weergegeven. Dit wordt gedaan om aan te geven welke interactie tussen de objecten onderling is. Hier worden operaties van deze objecten aangeroepen om het gewenste resultaat te bereiken. Dit is overzichtelijk weergegeven in het sequence diagram.

De use case is onder te verdelen in 3 processen namelijk:

- het inlezen en overzetten van de Knaakbaakbestanden onderverdeeld in:
  - Het inlezen en overzetten van de Kostenbestanden
  - Het inlezen en overzetten van de Projectbestanden
- Het wegschrijven van de Calkey structuur.

Het overzetten van de bouwelementen uit de Knaakbaak structuur naar de Calkey structuur kan per bouwelement worden uitgevoerd. Hierbij is het wel van belang dat de recepten behorend bij het bouwelement bekend zijn bij het invoeren.

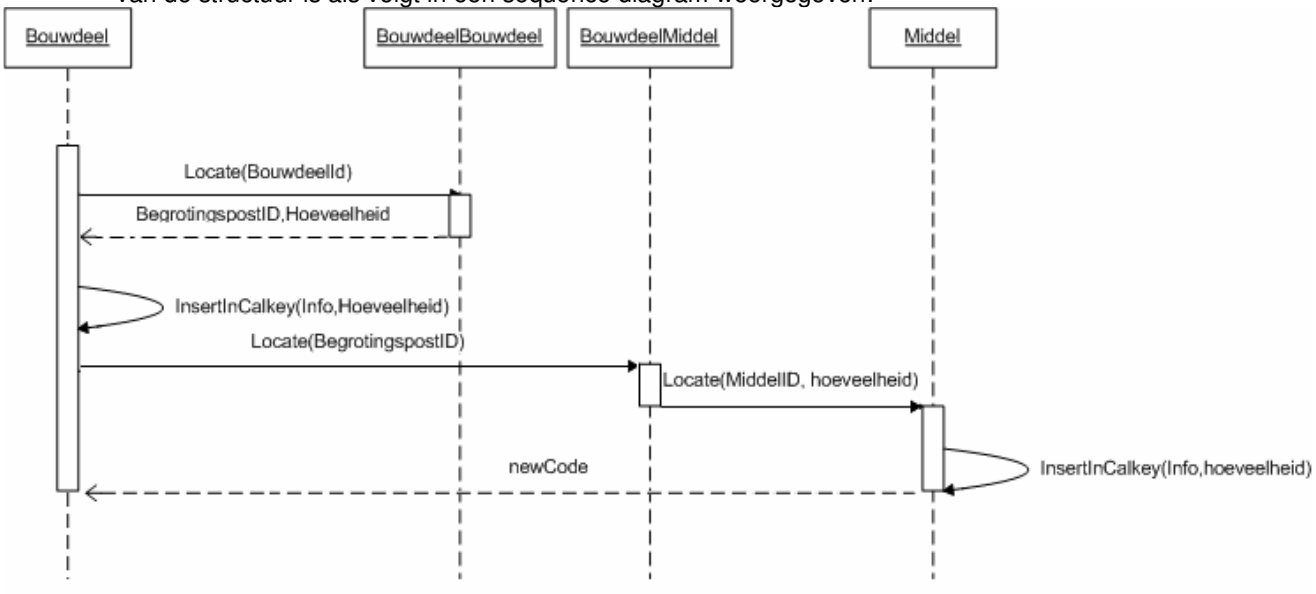
Hier is onderscheid gemaakt in twee gedeeltes namelijk de Knaakbaak en Calkey. Het toevoegen van de kopregels met de daarbij behorende receptregels (AppendKopData en AppendReceptData) in de Calkeystructuur wordt voor elk bouwelement dat in de Knaakbaak structuur aanwezig is uitgevoerd. Elk van de volgende sequence diagrammen beschrijft een deel van dit algemene sequence diagram. De volgende delen zijn te onderscheiden:

- Het inlezen van de kostenbestanden van de Knaakbaak
- Het inlezen van de projectbestanden van de Knaakbaak
- Het invoeren van de elementen in de Calkey
- Het comprimeren van het laagste niveau
- Het wegschrijven van de Calkeystructuur

Het toevoegen van een element (appendKopData en appendReceptData) wordt elke keer uitgevoerd als er een element wordt overgezet van de Knaakbaakstructuur naar de Calkeystructuur. In de komende diagrammen wordt dit als de operatie InsertInCalkey weergegeven.

4.2 Inlezen Kostenbestanden

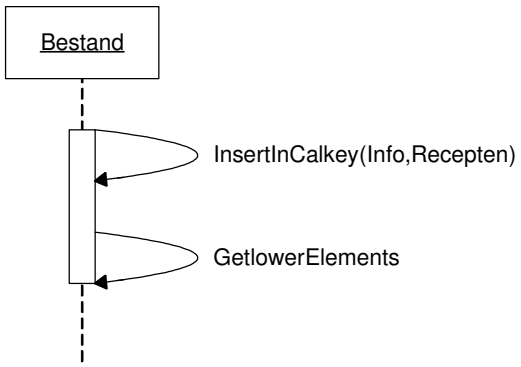
Het is voor de structuur van de Calkey niet belangrijk of het in te voeren bouwelement een project, gebouwdeel, bouwdeel, begrotingspost of een middel betreft. Het enige wat van belang is dat er bij het toevoegen van een element bekend is welke elementen bij dat element behoren. Het doorlopen van de structuur is als volgt in een sequence diagram weergegeven:



De bouwdelen zelf waren gekoppeld door middel van de Code. Hier was bijvoorbeeld Code *1* het bouwelement wat boven het element met de Code *1.10.* Deze elementen worden bij elkaar gezocht door GetLowerElements. De operatie GetLowerElements herhaalt zich totdat alle bouwdelen zijn doorlopen. Per bouwdeel wordt er gekeken of een bouwdeel of begrotingspost onderliggende elementen bevat. Als dit het geval is dan worden deze in de Calkey structuur geplaatst.

4.3 Inlezen Projectbestanden

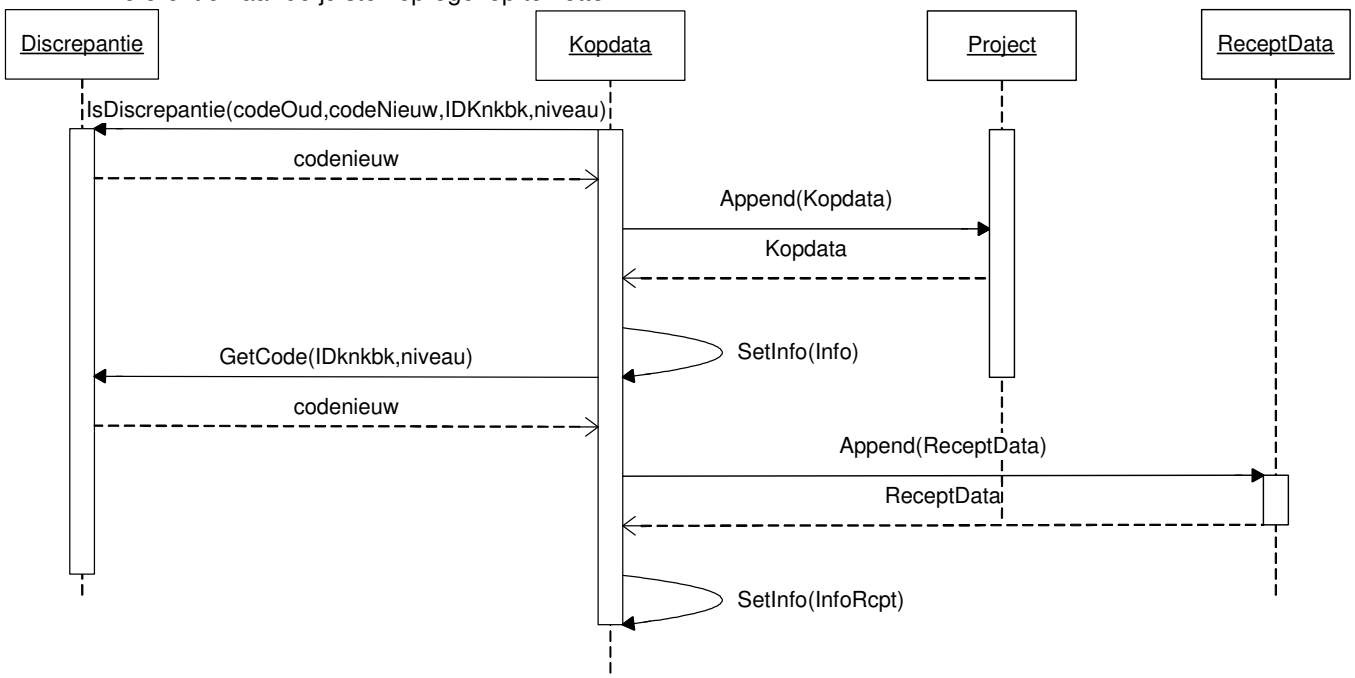
Bij het inlezen van de Projectbestanden wordt er maar gebruik gemaakt van één tabel. Dit is de tabel Bestand die gekoppeld is doormiddel van *Id* en *VaderID.* De operatie GetLowerElement in het onderstaande sequence diagram zoekt de onderliggende elementen op. Vervolgens wordt het element met de recepten van de onderliggende elementen in de Calkeystructuur geplaatst. Dit is weergegeven in het volgende sequence diagram:



Ook hier kan het invoeren van de elementen in de Calkeystructuur weer worden weergegeven zoals is weergegeven in afbeelding 6.

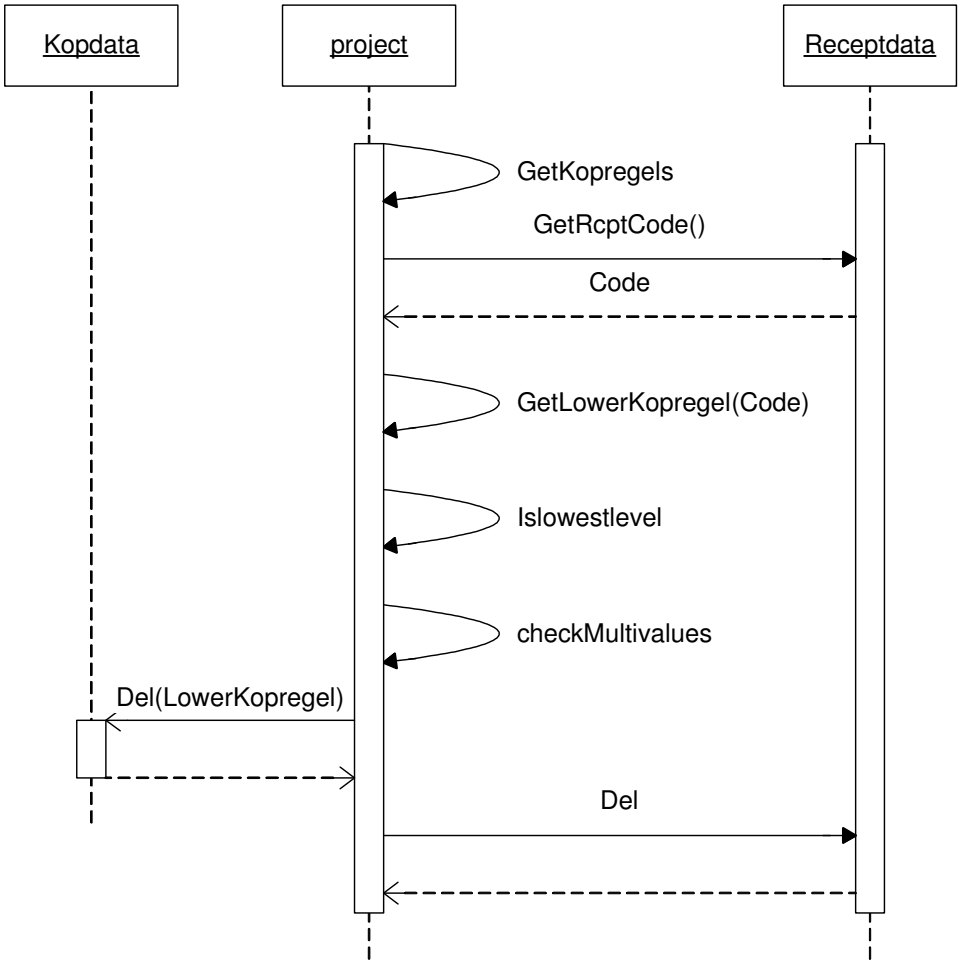
4.4 Invoeren in Calkey

Elke keer als InsertInCalkey wordt aangeroepen wordt het bouwelement inclusief de receptregels (refererend naar de onderliggende items) in de Calkey structuur gezet. IsDiscrepantie controleert of de code van de kopregel al bestaat. Als dit het geval is wordt er een nieuwe code aangemaakt en wordt deze gebruikt en bewaard in Discrepantie. Ter identificatie dient IDKnkbk en niveau. Om deze reden dient er ook van het laagste niveau ingevoerd te worden. Bij het invoeren van de recepten wordt er gecontroleerd of er voor de in te voeren code een nieuwe code is gegenereerd om de referentie naar de juiste kopregel op te zetten.



4.5 Comprimeer laagste niveau

Bij het comprimeren van het laagste niveau wordt de lijst met kopregels doorlopen. Per kopregel wordt, via de receptregels, de onderliggende kopregel opgezocht. Indien dit een kopregel op het laagste niveau is en de prijzen van de kopregels allemaal voor verschillende kostenposten zijn worden de waarden in de bovenliggende kopregel gezet en wordt de kopregel verwijderd. Vervolgens worden de receptregels verwijderd.



4.6 Wegschrijven Calkeystructuur

De procedures voor het wegschrijven van de Calkeystructuur zijn er al. Deze komen uit de sourcecode van de Calkey zelf. Als alles op de juiste manier in de Calkeystructuur wordt gezet, verloopt dit proces naar behoren en kan het weggeschreven bestand ingelezen worden in Calkey.

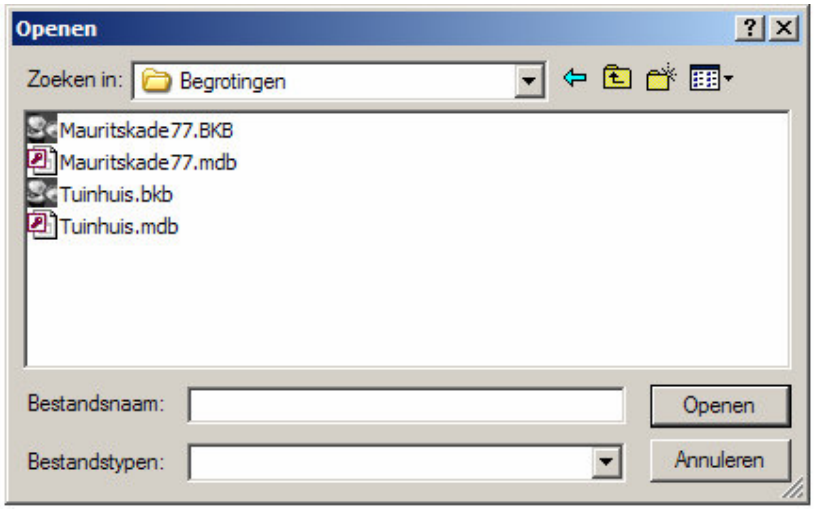
5 Specificeer beeldscherm en lijstindelingen

5.1 Inleiding

In dit gedeelte worden de schermen en lijstindelingen die gebruikt worden voor de interactie van de gebruiker weergegeven. De use cases gaven de interactie aan dus per use case dient er een scherm gemaakt te worden. Daarnaast zal er nog een hoofdscherm komen waarvan uit onder andere het scherm om de bestanden te selecteren te openen is. Ook zullen er in dit scherm zaken als status van het programma, tijd, een toolbar, maar ook de discrepanties weergegeven worden.

5.2 Selecteren bestanden

Het scherm voor het selecteren van de bestanden ziet er als volgt uit:



Hierbij moeten er bij bestandstypen een drietal bestandstypen te selecteren zijn namelijk de knaakbaak kostenbestanden en projectbestanden en database bestanden. Hierbij dient de gebruiker te kunnen navigeren door de bestanden op het systeem van de gebruiker.

5.3 Hoofdscherm

In het Hoofdscherm dienen een aantal zaken geregeld te worden namelijk:

- De gebruiker moet kunnen aangeven dat hij bestanden wil selecteren
- De gebruiker moet na het proces de discrepanties kunnen lezen
- De gebruiker moet aan kunnen geven op welke manier de conversie moet worden uitgevoerd.

Om de gebruikersvriendelijkheid te vergroten wordt:

- De tijd weergegeven
- De status van de applicatie weergegeven

Het hoofdscherm zal er zo uit komen te zien:

6 Specificeer programmatuur

6.1 Inleiding

In dit gedeelte zal de programmatuur worden gespecificeerd. De klassen van de Calkey en de tabbellen van de Knaakbaak bestaan al en worden hieronder gedocumenteerd weergegeven. Hierbij worden alleen de klassen, attributen, operaties, tabbellen en velden weergegeven die gebruikt worden door de Knalkey. Ook zullen de nieuwe operaties en de nieuwe klasse discrepantie gespecificeerd moeten worden.

Per operatie, gebruikt in de sequence diagrammen zal er beschreven worden wat de pre- en postcondities zijn en er zal in een korte omschrijving weergegeven worden wat de operatie doet.

6.2 Knaakbaak tabbellen en velden

Hier worden de tabbellen, velden en de typen van de velden weergegeven. Hierbij worden alleen de tabbellen en velden weergegeven die gebruikt gaan worden door de Knalkey:
Allereerst de tabbellen van het Knaakbaak referentiebestand:

Tabel Bouwdeel:

Veld	Type
BouwdeelID	Integer
Code	String
Omschrijving	String
Hoeveelheid	Double
Eenheid	String

Tabel BouwdeelBouwdeel:

Veld	Type
BouwdeelKindID	Integer
Hoeveelheid	double

Tabel BouwdeelMiddel:

Veld	Type
BouwdeelMiddelID	Integer
MiddelId	Integer
Hoeveelheid	double

Tabel Middel:

Veld	Type
ID	Integer
Code	String
Omschrijving	String
Eenheid	String
Prijs	Double
Kostensoort	Integer

De tabel Bestand van het Knaakbaak projectbestand:

Veld	Type
Id	Integer
VaderID	Integer
Code	String
Omschrijving	String
Hoeveelheid	Double
Materiaal	Double
Materieel	Double
Loon	Double
Onderaanneming	Double

6.3 Calkey klassen, attributen en operaties

Hier worden de klassen, attributen en operaties van de Calkey weergegeven. Let wel dat ook hier alleen de delen die gebruikt worden door de Knalkey, weergegeven worden. Als eerste worden de attributen met hun type per klasse weergegeven, vervolgens de gebruikte operaties.

Record KopdataInfo:

Attribuut	Type
Code	String (20)
Desc	String (60)
Eenh	String
Kcod	Char
ManH	Double
Cost	Double
MatC	Double
SubC	Double
DivC	Double

Record ReceptDataInfo:

Attribuut	Type
Code	String (60)
Qnty	Double

Record ProjectInfo:

Attribuut	Type
Versie	String
ProjName	String
Loan	Double
BTW	Double

6.3.1 Operaties van de klasse Projectdata

Naam: Append (kopdata)

Preconditie: Er bestaat een project en de code van de in te voeren kopregel bestaat nog niet.

PostConditie: De nieuwe kopregel is aan de lijst met kopregels toegevoegd en het aantal kopregels is met één verhoogd.

Omschrijving: De operatie voegt een kopregel toe aan het project.

Naam: GetHead (code)

Preconditie: Er bestaat een project.

Postconditie: De kopregel met de met de meegegeven code of, als de kopregel met die code niet bestaat, een nil pointer is het wordt geretourneerd door de operatie.

Omschrijving: De lijst met kopregels wordt doorlopen tot de kopregel met de code gevonden is. Vervolgens wordt de kopregel als resultaat van de operatie gezet. Als de kopregel met de desbetreffende code niet bestaat wordt het resultaat ‘nil’

Naam: Save

Preconditie: Er bestaat een project

Postconditie: Het project is naar bestand weggeschreven.

Omschrijving: Zorgt er voor dat het project weg wordt geschreven naar een bestand.

Naam: Write

Preconditie: Er is een bestand geopend om te schrijven

Postconditie: Het record is weggeschreven naar het bestand.

Omschrijving: Schrijft het ProjectInfo record als blok weg naar het geopende bestand.

6.3.2 Operaties van de klasse Kopdata

Naam: Append

Preconditie: -

Postconditie: Er is een receptregel toegevoegd aan de kopregel en het aantal receptregels is met één opgehoogd.

Omschrijving: Append voegt een receptregel toe aan de kopregel, zorgt dat de pointers goed komen te staan en verhoogd het aantal receptregels met één.

Naam: Del

Preconditie: Er bestaat een project met daarin de te verwijderen kopregel

Postconditie: De kopregel is verwijderd.

Omschrijving: Doorloopt de lijst van de kopregels van het project, als de juiste kopregel gevonden is wordt de kopregel uit de lijst verwijderd.

Naam: Findrecept

Preconditie: Er bestaat een project.

Postconditie: Het eerste recept dat naar de meegegeven kopregel verwijst worden geretourneerd.

Omschrijving: Doorloopt de lijst met kopregels en kijkt of de kopregels een recept bevatten die naar de meegegeven kopregel refereert. Als er een recept gevonden is wordt er gestopt met zoeken en wordt het recept als resultaat meegegeven.

Naam: Write

Preconditie: Er is een bestand geopend om te schrijven

Postconditie: Het KopdataInfo record is weggeschreven naar het geopende bestand.

Omschrijving: Schrijft het inforecord als blok weg in het geopende bestand.

6.3.3 Operaties van de klasse ReceptData

Naam: Del
Preconditie: De kopregel waar het recept toe behoord bestaat.
Postconditie: De receptregel is verwijderd en uit de lijst receptregels gehaald. Ook het aantal is met één verlaagd.
Omschrijving: De lijst met receptregels behorend bij de kopregel wordt doorlopen. Als de receptregel gevonden is wordt deze verwijderd en het aantal wordt met een verlaagd.

Naam: Write
Preconditie: Er is een bestand geopend om te schrijven
Postconditie: Het ReceptdataInfo record is weggeschreven naar het geopende bestand.
Omschrijving: Schrijft het inforecord als blok weg naar het geopende bestand.

6.4 Knalkey klasse en operaties

Hier wordt de structuur van de klasse discrepantie weergegeven en de operaties gebruikt in de sequence diagrammen worden beschreven.

De klasse Discrepantie:

Attribuut	Type
CodeNieuw	String (20)
CodeOud	String (20)
IdKnkbk	Integer
Niveau	Integer

De operaties gebruikt in de sequence diagrammen van de Knalkey:

Naam: GetLowerElements (ten behoeve van tabel Bouwdeel)
Preconditie: De tabel bouwdeel is geopend.
Postconditie: De elementen onder een bouwelement zijn gevonden.
Omschrijving: Zoekt onderliggende elementen op aan de hand van de code.

Naam: Locate (BouwelementId)
Preconditie: De tabel waar het element in moet worden opgezocht is geopend.
Postconditie: Het record met BouwelementID is het actieve record.
Omschrijving: Zoekt het record met waar BouwelementID overeenkomt met het Id in het record.

Naam: GetLowerElements (ten behoeve van tabel Bestand)
Preconditie: De tabel Bestand is geopend.
Postconditie: De elementen onder een bouwelement zijn gevonden.
Omschrijving: Zoekt onderliggende elementen op aan de hand van de VaderID en ID.

Naam: IsDiscrepantie.

Preconditie: Er bestaat een project.
Postconditie: Er is duidelijk of de code al bestaat, als dit het geval is is er een nieuwe code gegenereerd.
Omschrijving: Zoekt een kopregel met dezelfde code. Als deze niet bestaat is het geen discrepantie. Als deze wel bestaat moet er een unieke code worden samengesteld.

Naam: Setinfo(kopdata)
Preconditie: De kopregel bestaat.
Postconditie: De gegevens van het bouwelement zijn in de kopregel geplaatst.
Omschrijving: Zet de gegevens uit de Knaakbaak tabellen over in de kopregel.

Naam: Setinfo(Receptdata)
Preconditie: De receptregel bestaat.
Postconditie: De code en de hoeveelheid is in de receptregel geplaatst.
Omschrijving: Zet de gegevens uit de knaakbaak tabellen over in de kopregel.

Naam: GetCode (knkbkID, niveau)
Preconditie: -
Postconditie: De juiste code wordt geretourneerd.
Omschrijving: Zoekt de discrepantie met het meegegeven ID en niveau. Als deze wordt gevonden wordt de nieuwe code als het resultaat opgegeven. anders blijft het resultaat leeg.

Naam: GetKopregels
Preconditie: -
Postconditie: Alle Kopregels zijn doorlopen.
Omschrijving: Doorloopt de lijst met kopregels.

Naam: IsLowestKopregel (Kopregel)
Preconditie: -
Postconditie: Er is bepaald of de kopregel van het laagste niveau is.
Omschrijving: Bepaalt of de kopregel van het laagste niveau is. Als de kopregel recepten bevat dan is de kopregel niet van het laagste niveau

Naam: CheckMultiValues
Preconditie: -
Postconditie: Er is bepaald of onder de te onderzoeken kopregel elementen hangen die meerdere keren dezelfde kostenpost hebben
Omschrijving: Gaat de elementen na die onder de kopregel hangen en houd bij hoeveel keer een kostenpost voorkomt. Als dit er meer als 1 is dan wordt CheckMultiValues true.

7 Testplan

In dit plan zal beschreven worden hoe er getest gaat worden. De test is een middel om vast te stellen of de gestelde eisen en verwachtingen in het project behaald zijn. Er worden 2 soorten tests onderscheiden namelijk de acceptatietest en de systeemtest. In dit project wordt alleen de systeemtest uitgevoerd omdat de interface van de Knalkey niet uitgebreid en ingewikkeld is. De systeemtest is het middel van de ontwikkelaars om na te gaan of alle ontwikkelde delen functioneren en met elkaar beantwoorden aan de specificaties. Bij de systeemtest is ook de binnenkant belangrijk. Nagegaan zal worden of alle componenten naar behoren functioneren. De systeemtest omvat zowel de handmatige delen als de geautomatiseerde delen en hun onderlinge samenwerking.

Hoewel het testplan niet direct gericht is op het voltooien van het informatiesysteem wordt het testplan toch opgesteld. De verrichte inspanning is namelijk terug te vinden in de uiteindelijke kwaliteit. Het is namelijk niet wenselijk dat er fouten in het geconverteerde kostenbestand aanwezig zijn.

Teststrategie

De teststrategie van de systeemtest is het black-box testing. Hierbij zal er niet worden gekeken naar de interne werking van de applicatie maar zal er alleen gekeken worden naar de invoer en de uitvoer van de applicatie.

Testomgeving

Voor het testen is een werkstation benodigd. Op dit werkstation dient de Knaakbaak en de Calkey aanwezig te zijn om de invoer en de uitvoer te controleren. Op deze manier kan de conversie van de Knalkey worden gecontroleerd. Allereerst wordt de Knalkey getest door de ontwikkelaar zelf. Normaliter wordt er bij voorkeur getest door iemand anders als de ontwikkelaar. Aangezien het testen alleen het vergelijken van bestanden (structuur en bedragen) betreft kan in dit geval de Knalkey ook door de ontwikkelaar zelf getest worden.

Testprocedures

Als de test uitgevoerd wordt is het zaak dat deze wordt uitgevoerd zoals in dit document beschreven. Indien dat niet op deze manier gebeurd is het niet mogelijk om waarde aan de test te hechten. Het selecteren van de bestanden en de conversie van de Knalkey zal getest worden. Uit de conversie van de Knalkey komen ook de discrepanties voort. Per discrepantie zal er gekeken moeten worden of dit werkelijk een discrepantie is en of de nieuw gegenereerde code uniek is en of de code aan het juiste element gekoppeld is. Ook het comprimeren van het laagste niveau zal getest moeten worden. Deze functies zullen bij het projectbestand en bij het kostenbestand getest moeten worden. Indien de bovenstaande zaken door de test zijn gekomen zal de Knalkey functioneel goedgekeurd worden.

Testgevallen

Bij het testen wordt het kostenbestand, meegeleverd bij de knaakbaak, gebruikt. Dit kostenbestand dient in de knaakbaak ingelezen te worden als project bestand. Vervolgens dient het totaal van het kostenbestand afgelezen te worden. Als het bestand is geconverteerd worden de totalen met elkaar vergeleken. Indien deze totalen aan elkaar gelijk zijn kan er van uit gegaan worden dat de bedragen op de juiste manier zijn geconverteerd. Vervolgens wordt de structuur van het bestand in de Calkey vergeleken met het bestand in de Knaakbaak. Indien het laagste niveau gecomprimeerd is behoren de structuren met uitzondering van het laagste niveau en het niveau daarboven gelijk aan elkaar te zijn. De bedragen van het laagste niveau dienen in het niveau daarboven weergegeven te worden. Als het laagste niveau niet gecomprimeerd wordt dan behoren de structuren volledig aan elkaar gelijk te zijn. Op deze manier is ook het projectbestand te testen met uitzondering dat het projectbestand niet eerst ingelezen dient te worden in de Knaakbaak.

Indien er discrepanties zijn opgetreden worden deze weergegeven door de Knalkey. Allereerst dient er te worden getest of alle discrepanties daadwerkelijk gesignaleerd zijn. De tabellen van de Knaakbaak kunnen door MS Access worden gesorteerd op code. Op deze manier staan identieke codes direct onder elkaar. Indien het bestand geconverteerd is, behoort de Knalkey deze identieke codes weer te geven. Het aantal discrepanties dat de Knalkey dient weer te geven is het aantal identieke codes min 1.

Indien de bedragen of de structuren niet met elkaar overeen komen dient te worden genoteerd waar in de boomstructuur de afwijking is opgetreden. Dit wordt gedaan door het iedere keer

het bovenliggende item te noteren totdat het bovenste item is bereikt. Indien meerdere fouten gebaseerd zijn op dezelfde bug is een eenmalige notatie voldoende.