DE HAAGSE HOGESCHOOL

logica

Afstudeerverslag

“Implementatie SOM Selector”

november 2007 – maart 2008

J.W. van Rossem, 20055030

Algemene Gegevens

Titel:	Implementatie “SOM Selector”
Opleiding:	Informatica
Onderwijsinstelling:	Haagse Hogeschool
Afstudeerperiode:	19 november 2007 – 28 maart 2008

Examinandus

Naam:	Dhr. J.W. van Rossem
Studentnummer:	20055030
Straat / nummer:	Herenstraat 30
Postcode / plaats:	2291 BH Wateringen
Telefoonnummer:	06-24750244
Emailadres:	20055030@student.hhs.nl

Examinatoren

Naam:	Dhr. M. Reijnhoudt
Emailadres:	m.reijnhoudt@hhs.nl
Telefoonnummer:	070 – 4458473

Naam:	Dhr. V.H.F. Hermans
E-mailadres:	v.h.f.hermans@hhs.nl
Telefoonnummer:	070 – 4458460

Bedrijfsgegevens

Naam:	Logica Nederland B.V.
Afdeling:	Working Tomorrow Rijswijk
Straat / nummer:	Laan van Zuid Hoorn 70
Postcode / plaats:	2289 DE Rijswijk
Telefoonnummer:	070 – 3756000

Inhoudelijk begeleider

Naam:	Dhr. Mark Otter
Telefoonnummer:	070 – 3756000
Emailadres:	mark.otter@logica.com
Functie:	Consultant

Procesbegeleider

Naam:	Dhr. Jeroen Schipperijn
Telefoonnummer:	070 – 3756000
Emailadres:	jeroen.schipperijn@logica.com
Functie:	Projectleider van Working Tomorrow

Opdrachtgever

Naam:	Dhr. Michel Hulshof
Telefoonnummer:	070 – 3756000
Emailadres:	michel.hulshof@logica.com
Functie:	Business consultant

Referaat

Jorg van Rossem, Afstudeerverslag – “Implementatie SOM Selector”, Rijswijk, Logica Nederland, maart 2008.

Dit verslag beschrijft het proces rondom de werkzaamheden die in het kader van de bovengenoemde afstudeeropdracht zijn uitgevoerd. In de periode van november 2007 tot maart 2008 is de afstudeeropdracht uitgevoerd door een student aan de opleiding Informatica van de Haagse Hogeschool.

De afstudeeropdracht betrof het implementeren van een tool wat projectmanagers ondersteuning kan bieden bij het kiezen van een software ontwikkelmethode (SOM) voor hun projecten. Binnen de afstudeeropdracht is de gehele software ontwikkel cyclus doorlopen.

Descriptoren:
Decision Support System (DSS)
Software Development
Software Development Methods

Voorwoord

Na 18 weken hard gewerkt te hebben, is mijn project en afstudeerverslag dan voltooid. Ik zie het als een mooie afsluiting van mijn opleiding. Ik wil dan ook in het bijzonder mijn drie collega studenten Maarten, Caspar en Laurens bedanken voor de leuke tijd die ik heb gehad bij al die projecten die we samen uitgevoerd hebben. Vanuit Logica wil ik Michel Hulshof en Mark Otter bedanken voor het enthousiasme wat zij toonden bij het begeleiden van mijn opdracht. Ook wil ik Jeroen bedanken voor de begeleiding vanuit Working Tomorrow. Daarbij wil ik mijn SLB'er Wilma Stroomer bedanken voor de begeleiding op school tijdens mijn opleiding en Martin Reijnhoudt en Victor Hermans voor de begeleiding vanuit school tijdens mijn afstudeerperiode.

Als laatst wil ik mijn familie bedanken voor de steun tijdens de afstudeerperiode en mijn vriendin, Esther, die altijd klaar stond om te helpen en me altijd heeft gemotiveerd zodat ik mijn project en afstudeerverslag succesvol kon afronden.

Jorg van Rossem, maart 2008.

Inhoudsopgave

1 Inleiding

Afstuderen is onderdeel van de afsluiting van de opleiding ‘Informatica’ aan de Haagse Hogeschool. In deze periode van 18 weken moet een student een opdracht van HBO-niveau uitvoeren binnen een organisatie dat aansluit op de studie. Ik heb dit gedaan bij Logica en heb hier de implementatie van de SOM Selector uitgevoerd. Dit document beschrijft de aanpak, werkwijze, ondernomen stappen, de gemaakte keuzes en de evaluatie van de afstudeerstage.

Dit document is geschreven voor de examinatoren en de gecommitteerde die het project vanuit school zullen gaan beoordelen. Het doel van het document is dan ook om zoveel mogelijk inzicht te geven in het project, zodat het mogelijk wordt deze te beoordelen.

Het eerstvolgende hoofdstuk begint met een toelichting van de organisatie. In het hoofdstuk “Opdrachtomschrijving” wordt de opdracht beschreven zoals deze in definitieve vorm is vastgesteld. Het hoofdstuk “Uitgangssituatie” biedt inzicht in de geschiedenis van het project. In het hoofdstuk “Totstandkoming van het plan van aanpak” wordt beschreven hoe ik tot het plan van aanpak ben gekomen. Het hoofdstuk “Inception fase” beschrijft welke documentatie ik heb bestudeerd. Het hoofdstuk “Elaboration fase: 1^{e} Iteratie” laat zien hoe ik tot het class diagram ben gekomen. “Elaboration fase: 2^{e} iteratie” beschrijft de ontwikkeling van de eerste functionaliteit van de SOM Selector. Vervolgens wordt in “Construction fase – 1^{e} iteratie” beschreven hoe ik de andere functionaliteit ontwikkel. “Transition fase” beschrijft het onderzoek naar informatie uit de SOM Selector. Het verslag wordt afgesloten met een evaluatie van zowel de procesgang als de producten.

De definitieve opdrachtomschrijving is opgenomen als interne bijlage. Alle andere documenten die zijn ontstaan zijn opgenomen in de externe bijlage.

2 Organisatie

Dit hoofdstuk beschrijft de organisatie Logica, waar de afstudeeropdracht wordt uitgevoerd. Tevens beschrijft dit hoofdstuk het afstudeerprogramma van Logica, Working Tomorrow.

2.1 Logica

De afstudeeropdracht wordt uitgevoerd bij Logica te Rijswijk. Logica plc. is op 30 december 2002 ontstaan uit het voormalige Logica plc. (60%) en het voormalige CMG plc. (40%). Beide IT-dienstverleners zijn van oorsprong Engelse bedrijven, maar CMG was in Nederland veel groter dan de Engelse moeder. Sinds 13 januari 2006 is tevens het Franse Unilog onderdeel van het bedrijf, waarmee het een derde thuismarkt heeft gecreëerd[1]. Logica is een internationale IT-dienstverlener en heeft wereldwijd momenteel 40.000 werknemers in 40 verschillende landen in dienst. Het behoort, ook qua omzet, tot de internationale top-20 in de IT-dienstverlening. De omzet uit de traditionele IT-dienstverlening wordt voornamelijk in Europa en Australië (continent) gehaald. De omzet uit de rest van de wereld is sterk gebonden aan de telecommunicatie-industrie. Logica levert diensten op tal van terreinen, zoals management- en IT-consultancy, systeemontwikkeling en –integratie en neemt voor klanten complete bedrijfsprocessen in beheer. Het bedrijf ontwikkelt en implementeert oplossingen voor klanten over de hele wereld. Zij maakt daarbij gebruik van geavanceerde technologieën die direct doorwerken in de bedrijfsresultaten van de klant. Logica heeft dit verwoord in haar missionstatement: "*To help leading organisations worldwide achieve their business objectives through the innovative delivery of information technology and business process solutions*"[2].

2.2 Working Tomorrow

De afstudeeropdracht valt binnen het programma Working Tomorrow van Logica. Binnen dit programma houdt men zich bezig met het toetsen van nieuwe technologische ontwikkelingen op haalbaarheid en mogelijkheden. De projecten binnen Working Tomorrow worden uitgevoerd door studenten die hiermee hun afstudeeropdracht kunnen vervullen. Studenten krijgen de kans om een innovatieve opdracht binnen een bedrijfsomgeving uit te voeren. Working Tomorrow is een landelijk programma, maar wordt regionaal geleid. Het programma is bij de vestiging Rijswijk gestart in februari 2007 en biedt plaats aan ongeveer achttien studenten om binnen het programma aan hun afstudeeropdracht te werken.
Het Working Tomorrow programma heeft 3 hoofddoelen:

- Het vinden van toekomstige werknemers.
- Verhogen van de reputatie van Logica op het gebied van innovatie.
- Demo's en prototypen van projecten gebruiken voor het verkrijgen van betaalde opdrachten.

[1] Bron: http://nl.wikipedia.org/wiki/LogicaCMG

[2] Bron: www.logicacmg.com

3 Opdrachtomschrijving

Dit hoofdstuk bevat de probleemstelling en de doelstelling van de definitieve opdrachtomschrijving. In bijlage A is de volledige opdrachtomschrijving te vinden.

3.1 Probleemstelling

In opdracht van het Competence Centre Software Ontwikkel Methoden van Logica (CCSOM) heeft Klaas-Jan Molendijk van de Universiteit Leiden de kennis en ervaring omtrent software ontwikkel methoden (SOMmen) beschreven, gekwantificeerd en een beslismodel beschreven waarmee de meest geschikte SOM kan worden bepaald op basis van een projectprofiel. De thesis van de Klaas-Jan is een belangrijk uitgangspunt voor de implementatie van de SOMSelector.
Jorn van Veelen van de HHS heeft de implementatie van de SOM Selector al eerder uitgevoerd, maar de uitgangssituatie verschilt met deze opdracht, want het onderzoek van Klaas-Jan was ten tijde van de opdracht nog gaande. Bij de aanvang van deze opdracht is er wel een afgerond en geaccepteerd afstudeerverslag van Klaas-Jan. Jorn heeft wel de bruikbaarheid en haalbaarheid aangetoond van SOM Selector. Dit zal verder worden toegelicht in hoofdstuk 4.

3.2 Doelstelling

Het doel van de opdracht is om het beslismodel te implementeren in een decision support tool, de zogenaamde SOM Selector. De SOM Selector wordt geïntegreerd in een website over Software Ontwikkel Methoden. De SOM Selector zal gebruik maken van een kennisdatabase wat informatie bevat over SOMmen zoals beschreven staat in de thesis van Klaas-Jan. Voor de SOM Selector bestaat een lijst met functionaliteiten dat geprioriteerd is met behulp van de MoSCoW-methode. Voor sommige van deze functionaliteiten is reeds een functionele specificatie beschikbaar, die eventueel aangepast moet worden.
De SOM Selector zal op basis van deze functionele specificaties worden ontwikkeld. Door de lengte van MoSCoW-lijst is met de opdrachtgever afgesproken om alleen de belangrijkste functionele eisen te realiseren. De belangrijkste functionele eisen zullen iteratief worden gerealiseerd.

De belangrijkste functionele eisen zijn:

- Het kunnen invoeren van een projectprofiel, door middel van ruim 20 projectkarakteristieken.
- Berekenen van een advies dat is gebaseerd op het beslismodel en de inhoud van de kennisdatabase. Dit advies zal bestaan uit een gesorteerde lijst van meest geschikte SOMen, waarbij iedere SOM een score heeft tussen 0,00 en 10,00.
- Presenteren van de adviesonderbouwing. Deze onderbouwing bestaat uit de kwantificatie van na te streven doelstelling, eventueel aangevuld met risico's in een projectsituatie en de proceselementen van een SOM.
- Opslaan van het projectprofiel en (onderbouwing van) het bijbehorende advies. Dit zorgt ervoor dat de gebruikers hun advies opnieuw kunnen bekijken.

4 Uitgangssituatie

In de probleemstelling in hoofdstuk 3 is beschreven wat de geschiedenis is van dit project. De vorige projecten hebben een aantal producten opgeleverd wat wellicht bruikbaar kan zijn voor mijn project. Hieronder zal ik beschrijven wat deze producten inhouden.

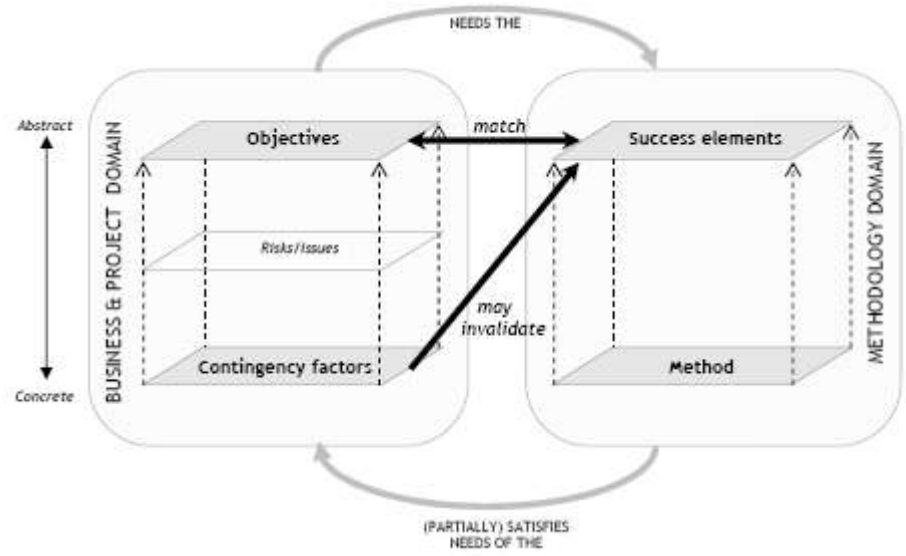
4.1 Onderzoek naar software ontwikkelmethoden

Klaas-Jan Molendijk heeft een wetenschappelijk onderzoek gedaan naar de projecteigenschappen die van invloed zijn op de selectie van een software ontwikkelmethode en de eigenschappen van de software ontwikkelmethoden (RUP, SDM, DSDM, SCRUM, XP, MSF4Agile en MSF4CMMI) zelf. Tevens heeft hij een beslismodel voorgesteld waarmee projectmanagers advies kunnen krijgen over de meest geschikte SOM voor hun specifieke project.

Klaas-Jan had in zijn onderzoek een beslismodel beschreven waarmee je op basis van projecteigenschappen een advies van de beste software ontwikkelmethode krijgt. Ik heb er voor gekozen om alleen het beslismodel te beschrijven omdat dat relevant is voor dit project. Het beslismodel is opgesteld door middel van een aantal achtereenvolgende stappen. Klaas-Jan heeft het beslismodel conceptueel en operationeel beschreven. Beide modellen zal ik hieronder beschrijven.

4.1.1 Conceptuele model

Het conceptuele model is bedoeld om de denkwijze uit te leggen. Het model beantwoordt de vraag of een software ontwikkelmethode kan worden gekozen op basis van projecteigenschappen. In figuur 4.1 is het conceptuele model weergegeven.



Figuur 4.1 Het conceptuele model

Voor de vier grijze vakken zal ik beschrijven wat de tussenliggende relaties inhouden.

Contingency Factor → Objective
Dit is de relatie tussen projecteigenschappen (contingency factors) en doelstellingen (objectives). Het idee van Klaas-Jan was dat bij een projecteigenschap risico's horen. En die risico's kunnen worden beheerst door middel van de doelstellingen te behalen. Hieronder zal ik een voorbeeld laten zien:

Projecteigenschap: Complexiteit van de software.
Bijbehorend risico: integratie problemen, veel werk opnieuw doen, laag moraal, etc.
Doelstelling: Het begrijpen van de complexiteit in plaats van het over je heen laten komen.

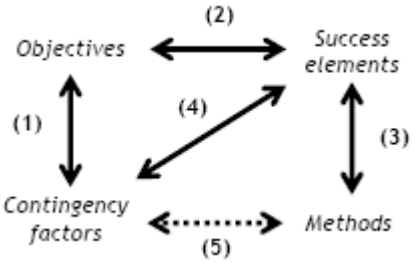
Objective → Succes element → Method
Dit is de relatie tussen een doelstelling en een succes element van een methode. Een succes element is een sterke eigenschap van een ontwikkelmethode. Een voorbeeld van een succes element van RUP is bijvoorbeeld "architectuur gedreven". Klaas-Jan beschrijft dat door middel van deze relatie een doelstelling en een succes element van een methode elkaar kunnen matchen. Dus om de doelstelling te halen kan het succes element van een methode gebruikt worden.
Hieronder zal ik een voorbeeld laten zien waarbij het succes element helpt bij het behalen van de doelstelling.

Doelstelling: Het begrijpen van de complexiteit in plaats van het over je heen laten komen.
Succes element: Iteratief ontwikkelen.
Methode: RUP

Contingency Factor → Succes Element
Een succes element van een methode kan een perfecte oplossing zijn voor een doelstelling. Maar dan is het nog wel afhankelijk of dit kan worden gebruikt in een bepaalde situatie. Deze relatie laat zien dat bepaalde projecteigenschappen een succes element ongeldig kan verklaren. Bijvoorbeeld een succes element van DSDM: continue feedback van de klant. Dit is alleen mogelijk als de klant ook actief participeert en als de functionaliteit van de software heel zichtbaar is (beide zijn ook projecteigenschappen).

4.1.2 Operationele model

Dit model zal de vertaling zijn van het conceptuele model. Het operationele model beschrijft hoe het verkrijgen van een match tussen projecteigenschappen en methoden gerealiseerd kan worden. Dit model zal ook de basis zijn voor de SOM Selector. In figuur x is het operationele model te zien.



Figuur 4.2 Het operationele model

De vijf relaties die in dit model te zien zijn zal ik hieronder beschrijven. Ik zal per relatie beschrijven hoe Klaas-Jan vond dat dit gerealiseerd kon worden.

Relatie 1: Contingency factor ←→ Objective
Deze relatie beschrijft welke doelstellingen relevant zijn in welke situatie. Klaas-Jan beschrijft 21 projecteigenschappen in zijn thesis. Een projecteigenschap kan verschillende doelstellingen impliceren. Elke projecteigenschap wordt gekwalificeerd met een keuze uit een lijst van vooraf gedefinieerde waarden. In tabel 4.1 is een voorbeeld te zien.

Projecteigenschap	Vooraf gedefinieerde waarden
Time pressure	Very strong
	Strong
	Moderate
	Weak
	Very weak

Tabel 4.1 Voorbeeld van een projecteigenschap met vooraf gedefinieerde waarden

Niet alle doelstellingen zijn even belangrijk. In het onderzoek heeft Klaas-Jan de doelstellingen tegen elkaar op gewogen om aan te geven welke doelstellingen belangrijk zijn en welke niet. Omdat dit voor elk project ook kan verschillen worden ze in de berekening ook gewogen zodat de belangrijkste doelstellingen voor dat project ook het zwaarst meetellen. De laatste weging zal met behulp van de gekwalificeerde projecteigenschappen worden gedaan.

In tabel 4.2 staat een voorbeeld van een dergelijke weging.

Ensuring the maintenance ease of software										
Expected lifetime	very long	very long	very long	very long	very long	long	long	long	long	long
Softw.complexity	very high	high	moderate	low	very low	very high	high	moderate	low	very low
Relative importance	1,0	1,0	0,9	0,8	0,7	0,8	0,8	0,7	0,6	0,5

Tabel 4.2 Voorbeeld van een weging

In tabel 4.2 staat de doelstelling “Ensuring the maintenance ease of software”. De projecteigenschappen “Expected lifetime” en “Software complexity” hebben beide invloed op deze doelstelling. Hoe hoger de “Relative Importance” oftewel belangrijkheid van de doelstelling is, hoe zwaarder die wordt meegeteld in de berekening. De hoogte van de belangrijkheid van de doelstelling is afhankelijk van de opties die worden gekozen in het project. Dus als bij de projecteigenschap “Expected lifetime” long wordt gekozen en bij “Software complexity” low dan is de belangrijkheid van de doelstelling 0.6.

Een doelstelling kan bij verschillende projectsituaties belangrijk zijn. Daarom zijn er bevredigingsniveaus (satisfaction level) opgenomen. Aan de hand van het voorbeeld in tabel 4.3 en figuur 4.3 zullen bevredigingsniveaus worden uitgelegd.

Enabling a quick time-to-market of working software					
Time pressure	very strong	strong	moderate	weak	very weak
Relative importance	1,0	0,8	0,6	0,2	0,0
Satisfaction level A (=standard)	100%	50%	0%	0%	0%
Satisfaction level B	0%	50%	100%	100%	0%

Tabel 4.3 Doelstelling met meerdere bevredigingsniveaus

- Satisfaction Level A. Enabling a delivery of working software to the business measured in weeks.
- Satisfaction Level B. Enabling a delivery of working software to the business measured in months.

Figuur 4.3 De bevredigingsniveaus van de doelstelling "Enabling a quick time-to-market"

In het voorbeeld hierboven is te zien dat de doelstelling “Enabling a quick time-to-market of working software” één projecteigenschap heeft wat invloed kan uitoefenen namelijk “Time pressure”. Omdat sommige projecten veel langer duren dan andere projecten verschilt de interpretatie van “quick time-to-market” per project. Als een project 2 jaar duurt, is de tijdsdruk minder relevant, maar “quick time-to-market” kan nog steeds gelden. Daarom zijn er twee bevredigingsniveaus opgenomen.

- Level A: “quick time-to-market” gemeten in weken.
- Level B: “quick time-to-market” gemeten in maanden.

De projectmanager kan door middel van deze bevredigingsniveaus zien of de doelstelling geldt voor zijn situatie.

Relatie 2 en 3: Objectives ←→ Succes elements ←→ Methods

Voor elke methode is een evaluatie gedaan om te zien hoe belangrijk een doelstelling is bij die methode. De numerieke score van 0 tot 10 wordt gebruikt om te laten zien hoe goed een methode de doelstellingen kan bevredigen. 10 betekent dat de methode perfect aansluit bij de doelstelling. 0 betekent dat de methode de doelstelling totaal niet bevredigt. De numerieke score wordt informatief onderbouwd door de succeselementen. De succeselementen hebben verder geen invloed op de berekening. In tabel 4.4 staat een voorbeeld van de numerieke score waarin te zien is dat RUP beter inzetbaar is bij complexe systemen dan XP.

	RUP	SDM	XP
Doelstelling: Het begrijpen van de complexiteit in plaats van het over je heen laten komen.	9	8	6

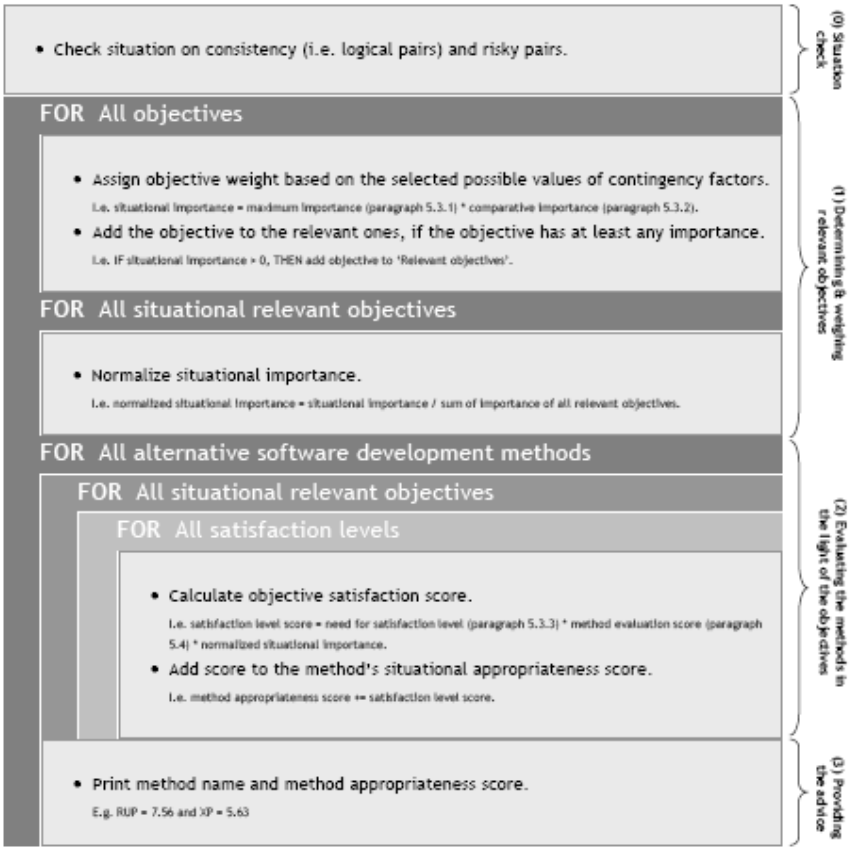
Tabel 4.4 Voorbeeld van de numerieke score voor een doelstelling per methode

Relatie 4: Succes elements ←→ Contingency factors

Deze relatie zorgt ervoor dat een waarde van een projecteigenschap (bijvoorbeeld voor projecteigenschap “Requirements clarity” de waarde “High”) een succeselement van een methode ongedaan kan maken. Bijvoorbeeld XP biedt een uitstekende flexibiliteit bij veranderingen en groeiende inzichten zodra er een actieve klant beschikbaar is. Zodra de klant dit niet is zal XP geen goede methode zijn voor jouw project. Wat de score dan ook is.

Relatie 5: Het matchen van een project en methodes.

Deze relatie bestaat uit het gehele model in de vorm van een algoritme. Dit algoritme produceert een lijst met methodes en de scores die bij situationele project horen. Dit algoritme is uiteindelijk ook gebruikt in de Excel sheet en de SOM Selector. In figuur 4.4 is het algoritme te zien.



Figuur 4.4 Het algoritme voor het berekenen van de scores

Het algoritme bestaat uit 4 stappen. Ik zal per stap in eigen woorden beschrijven wat er gebeurt.

Stap 0: situational check

Hier worden voor de projecteigenschappen gekeken of die risicovol of logisch zijn ten opzichte van elkaar. Bijvoorbeeld als de verandering van functionele eisen veel voor komt en het project werkt met een vaste prijs contract is dit heel risicovol. De situatie moet logisch zijn en er moeten geen risicovolle paren in zitten.

Stap 1: Determening & weighing relevant objectives

Hier wordt voor elke doelstelling de belangrijkheid berekend voor het project waarvoor het algoritme is uitgevoerd. Als een doelstelling niet relevant is voor het project zal deze verder niet worden gebruikt in de berekening. Bijvoorbeeld het in één keer perfect ontwikkelen is alleen relevant als het gaat om vaste functionele eisen.

Stap 2: Evaluating the methods in the light of the objectives

Hier zullen de eindcijfers per software ontwikkelmethode worden berekend voor de projectsituatie. Dit wordt gedaan door per relevante doelstelling een subscore te berekenen per methode. Als je alle subscores van één methode bij elkaar optelt krijg je de uiteindelijke score voor die methode. In figuur

4.5 staat een voorbeeld van een projectsituatie ter illustratie. De getallen tussen haakjes zijn de subscores. Het vetgedrukte cijfer is het eindcijfer.

Relevant objectives	Situational importance	RUP	SDM	DSDM	XP	SCRUM	MSF4AGILE	MSF4CMMI
Proving and following a technology's performance	22,2%	7 (1,55)	4 (0,89)	8 (1,77)	8 (1,77)	8 (1,77)	8 (1,77)	8 (1,77)
Being able to develop 'invisible' software	21,2%	9 (1,91)	9 (1,91)	3 (0,64)	3 (0,64)	3 (0,64)	3 (0,64)	3 (0,64)
Keeping a thorough grip on the outsourcing partner's work	18,8%	6 (1,13)	3 (0,57)	8 (1,51)	8 (1,51)	8 (1,51)	8 (1,51)	8 (1,51)
Compensating for they lack of customer domain knowledge	17,4%	5 (0,87)	2 (0,35)	8 (1,39)	8 (1,39)	8 (1,39)	8 (1,39)	8 (1,39)
Providing methodical support as a project leadership necessity	10,9%	9 (0,98)	8 (0,87)	4 (0,44)	1 (0,11)	2 (0,22)	3 (0,33)	4 (0,44)
Enlarging organizational commitment for the project	4,9%	6 (0,29)	2 (0,1)	10 (0,49)	10 (0,49)	10 (0,49)	10 (0,49)	10 (0,49)
Providing a comfort zone of 'freedom' to the customer	4,6%	5 (0,23)	3 (0,14)	8 (0,37)	9 (0,42)	9 (0,42)	9 (0,42)	8 (0,37)
		6,96	**4,82**	**6,6**	**6,32**	**6,43**	**6,54**	**6,6**

Figuur 4.5 Een voorbeeld van een berekening

Stap 3: Providing the advice

Hier wordt het advies getoond aan de gebruiker. Bijvoorbeeld RUP: 7.56, XP: 6.45.

4.2 Excel prototype

Klaas-Jan heeft een Excel sheet gemaakt waarmee hij kon tonen dat het beslismodel kan worden geïmplementeerd. In de Excel sheet is het mogelijk om een advies te verkrijgen. Dit was een goed prototype voor de SOM Selector. Door het gebruik van Excel kon de gewenste functionaliteit niet worden geïmplementeerd. Bijvoorbeeld de adviesonderbouwing of het toevoegen van nieuwe kennis en ervaringen. In de externe bijlagen is de Excel sheet terug te vinden.

De grootste tekortkoming van Excel is dat het alleen een score geeft. Het onderzoek heeft aangegeven dat de score eigenlijk niet zo heel relevant is, maar juist de redenering van die score. Welke risico's, doelstellingen en succeselementen maken een SOM belangrijker dan een andere.

Doordat de thesis het beslismodel beschrijft en de Excel sheet als prototype voor de SOM Selector dient vormen deze samen een goede basis voor ontwikkeling van de uiteindelijke SOM Selector.

4.3 Lijst met functionaliteiten

De lijst met functionaliteiten is opgesteld door Jorn van Veelen en de opdachtgever en is gebruikt bij het vaststellen van de functionele eisen die relevant zijn voor mijn project. Deze lijst bestaat uit functionaliteiten die in de SOM Selector kunnen komen. Omdat de lijst is geprioriteerd op basis van de MoSCoW-methode zal in samenspraak met de opdrachtgever worden besproken welke functionaliteiten zullen worden gerealiseerd. Hieronder staan 2 functionaliteiten uit de lijst ter illustratie. De volledige lijst zal terug te vinden zijn in de externe bijlagen.

P = Prioriteit

M = Must have

S = Should have

C = Could have

W = Won't have this but would like in the future

S = Stakeholder

Nr.	Naam	Omschrijving	P	S
1.1	Advies verkrijgen	Invoeren van projectprofiel (kwantificatie van projectkarakteristieken) en een advies over de meest/minst geschikte SOM o.b.v. kennis en model zoals vastgelegd in de (laatste versie van de) thesis van Klaas-Jan Molendijk.	M	Michel Hulshof

Tabel 4.5 Functionaliteit advies verkrijgen

Nr.	Naam	Omschrijving	P	S
3.1	SOM-profielen beheren	Beheer van SOM-profielen in de SSKB.	S	-

Tabel 4.6 Functionaliteit SOM-Profielen beheren

4.4 Ontwerpdocumentatie Jorn van Veelen

Jorn van Veelen is voor zijn afstudeeropdracht ook met de SOM Selector bezig geweest. Het probleem bij Jorn was dat de thesis van Klaas-Jan Molendijk nog niet definitief was. Hierdoor had hij geen vaste leidraad voor zijn project. Dit is ook een van de redenen om niet te veel naar zijn ontwerpdocumentatie te kijken.

4.5 Code Jorn van Veelen

Jorn had zijn programma vooral gericht op een oude versie van de thesis van Klaas-Jan. Hierdoor kwam de functionaliteit niet overeen met de definitieve thesis. Ook had Jorn niet object georiënteerd geprogrammeerd. Dit zorgde ervoor dat er geen structuur in het programma zat. De geprogrammeerde code van Jorn van Veelen was hierdoor niet bruikbaar, Jorn had wel de bruikbaarheid en haalbaarheid van de SOM Selector getoond.

5 Totstandkoming van het plan van aanpak

In dit hoofdstuk zal ik beschrijven hoe ik tot mijn plan van aanpak ben gekomen. Het plan van aanpak zorgde ervoor dat ik goed nadacht hoe ik mijn project het beste kon uitvoeren. Allereerst zal ik voor de op te leveren producten per product een korte omschrijving geven. Daarna zal ik de risico's beschrijven die geïdentificeerd zijn. Ook zal ik beschrijven welke ontwikkelmethode ik heb gekozen en waarom. Daarna zal ik de technieken beschrijven die ik heb gebruikt in mijn project. De aanpak die ik heb gehanteerd wordt beschreven en de planning en hoe ik daartoe ben gekomen. Het volledige plan van aanpak is terug te vinden in de externe bijlagen.

5.1 Eindproducten

De eindproducten die zijn opgesteld zullen hier worden beschreven. Per eindproduct zal een korte omschrijving worden gegeven.

Product	Omschrijving
SOM Selector	De software die wordt opgeleverd aan de opdrachtgever.
Systeemdocumentatie	Documentatie die ontstaan is bij het opzetten van het systeem.
Onderzoeksrapport presentatie SOM-kennis	Het onderzoek voor de presentatie van informatie uit de thesis als content op een website

Tabel 5.1 Eindproducten

5.1.1 Som Selector

De SOM Selector is de software die moet worden overgedragen aan de opdrachtgever. Dit zal bestaan uit de source-code en de database. Omdat de SOM Selector wordt doorontwikkeld zal de source-code en de database aan de coding standards en SQL standards moeten voldoen. Zodra er problemen of wijzigingen ontstaan, kunnen deze hierdoor efficiënt worden opgelost of uitgevoerd.

5.1.2 Systeemdocumentatie

De systeemdocumentatie bestaat uit de software requirements specification (SRS), het designmodel en de testrapporten. Deze documentatie is van groot belang voor het verdere verloop van de opdracht. Deze documenten zorgen ervoor dat de kwaliteit van de opdracht hoog blijft. Dit moet dus kloppen met wat er is geprogrammeerd en wat de opdrachtgever voor ogen had.

5.1.3 Onderzoeksrapport presentatie content

Dit onderzoeksrapport zal het resultaat zijn van een kort onderzoek naar de informatie over software ontwikkelmethoden in de database van de SOM Selector. Hoe kan dit het beste worden getoond op een website. Dit rapport geeft de opdrachtgever inzicht in welke informatie uit de database van de SOM Selector hij op de website kan tonen. Dit zal van belang zijn als de SOM Selector zich verder ontwikkelt en de opdrachtgever een informatief gedeelte over software ontwikkelmethoden op de website wil hebben.

5.2 Risico's identificeren

Om de risico's te identificeren heb ik de lijst met functionaliteiten gebruikt als leidraad. Door de beschikbare tijd is in overleg gekozen om eerst de "Must-haves" over iteraties te verdelen. Wellicht als het project sneller zou verlopen zou ik een extra iteratie op kunnen nemen.
Hieronder staan de "Must-haves" van de lijst met functionaliteiten:

P = Prioriteit
De betekenis van de prioritering is aangegeven in paragraaf 4.3
S = Stakeholder

Nr.	Naam	Omschrijving	P	S
1.1	Advies verkrijgen	Invoeren van projectprofiel (kwantificatie van projectkarakteristieken) en een advies over de meest/minst geschikte SOM o.b.v. kennis en model zoals vastgelegd in de (laatste versie van de) thesis van Klaas-Jan Molendijk.	M	Michel Hulshof
1.2	Projectprofiel & advies opslaan	De mogelijkheid om na het verkrijgen van een advies het projectprofiel én het advies op te slaan, zodat dit op een willekeurig moment geopend kan worden. Omdat over de tijd adviezen voor eenzelfde projectprofiel kunnen verschillen door de wijzigende SSKB zou het opslaan van het advies op het tijdstip van aanvragen gewenst zijn.	M	Michel Hulshof
1.3.1	Adviesonderbouwing aanvragen: SOM-score	Inzicht in succeselementen die ten grondslag liggen aan een SOM score.	M	Michel Hulshof, Klaas-Jan
1.3.2	Adviesonderbouwing aanvragen: sucesselementen	Inzicht in de succeselementen die een doelstelling implementeren.	M	Michel Hulshof, Klaas-Jan
1.3.3	Adviesonderbouwing aanvragen: risicoprofiel	Inzicht in de risico's die het projectprofiel impliceert.	M	Michel Hulshof, Klaas-Jan
1.3.4	Adviesonderbouwing aanvragen: Doelstellingen	Inzicht in de doelstellingen die door het risicoprofiel worden geïmpliceerd.	M	Michel Hulshof

Tabel 5.2 "Must-haves" van de lijst met functionaliteiten

5.2.1 Risico 1: Classdiagram is niet volgens het beslismodel

Ik kwam er al snel achter dat functionaliteit 1.1 de belangrijkste functionaliteit was. Als die functionaliteit niet ontwikkeld kon worden, dan zou het hele project niet door kunnen gaan. Functionaliteit 1.1 is namelijk de basis voor het systeem en zonder deze basis zou de SOM Selector geen nut hebben. Dit vond ik zo een groot risico dat ik heb besloten dat risico op te delen in 2 risico's. Het eerste risico was gericht op het classdiagram. Dit moest volledig overeen komen met het beslismodel van Klaas-Jan Molendijk.

Risico	Maatregel	Kans	Invloed
Classdiagram is niet volgens het beslismodel van Klaas-Jan Molendijk	Classdiagram veel terugkoppelen met de klant, zodat er geen fouten insluipen.	Middel	Hoog

Tabel 5.3 Risico 1: Classdiagram is niet volgens het beslismodel

5.2.2 Risico 2: Functionaliteit 1.1 kan niet worden geïmplementeerd

Het tweede risico dat ik had opgenomen beschrijft de vraag of functionaliteit 1.1 wel kan worden geïmplementeerd. De rest van de functionaliteiten zijn namelijk afhankelijk van de eerste functionaliteit. Je moet bijvoorbeeld eerst een advies verkrijgen voordat je hem kan opslaan.

Risico	Maatregel	Kans	Invloed
Functionaliteit 1.1 uit de lijst met functionaliteiten kan niet worden geïmplementeerd	Functionaliteit 1.1 als eerst ontwikkelen, zodat er snel duidelijk is of de SOM Selector kan worden gerealiseerd.	Middel	Hoog

Tabel 5.4 Risico 2: Functionaliteit 1.1 kan niet worden geïmplementeerd

Dit kwam er op neer dat zodra het classdiagram af was het project werd gericht op functionaliteit 1.1. Als deze functionaliteit niet kon worden gerealiseerd zou het project een andere wending moeten nemen. Dit was ook de reden om dit risico zo snel mogelijk aan te pakken.

5.2.3 *Risico 3: Uitloop van een iteratie.*

Het volgende risico was of er uitloop kon ontstaan van een iteratie. Dit was goed mogelijk omdat het beslismodel complex in elkaar zat. Zodra een iteratie uitloop zou hebben moest er direct met de klant worden gecommuniceerd wat voor maatregelen er moesten worden genomen.

Risico	Maatregel	Kans	Invloed
Uitloop van een iteratie.	Lijst met functionaliteiten opnieuw prioriteren met de klant.	Laag	Middel

Tabel 5.5 Risico 3: Uitloop van een iteratie

5.2.4 *Risico 4: Onvoldoende kennis van de ontwikkeltool*

Het laatste risico wat ik had opgenomen was onvoldoende kennis van de ontwikkeltool. In mijn tijd op school hebben wij 10 weken besteed aan Visual Studio 2003. Ik wist niet zeker of dit voldoende was voor mij om bij dit project de tool te ontwikkelen.

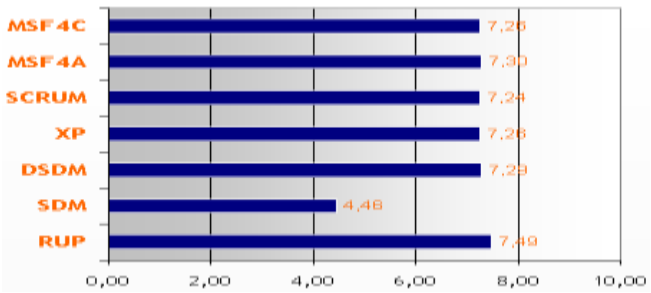
Risico	Maatregel	Kans	Invloed
Onvoldoende kennis van de ontwikkeltool	Mij verdiepen in Visual Studio door middel van boeken en internet.	Laag	Middel

Tabel 5.6 Risico 4: Onvoldoende kennis van de ontwikkeltool

Als ik er achter zou komen dat mijn kennis toch onvoldoende was dan zou ik contact op nemen met mijn begeleider. Met hem zou ik dan bespreken wat voor maatregelen we zouden kunnen nemen om te zorgen dat ik de functionaliteiten alsnog zou kunnen ontwikkelen.

5.3 Kiezen van een ontwikkelmethode

Voor het kiezen van een ontwikkelmethode voor mijn project heb ik een bepaalde afweging gemaakt, of een ontwikkelmethode gebruiken waar ik ervaring mee heb en mij daarin verder verdiepen, of een nieuwe ontwikkelmethode gebruiken en die helemaal uitzoeken. Bij het gebruik van de Excel sheet voor mijn eigen project kwam er uit dat er 5 methoden waren die ongeveer even relevant waren. Dit bood dus weinig hulp. In figuur 5.1 is de uitkomst te zien van de Excel sheet.



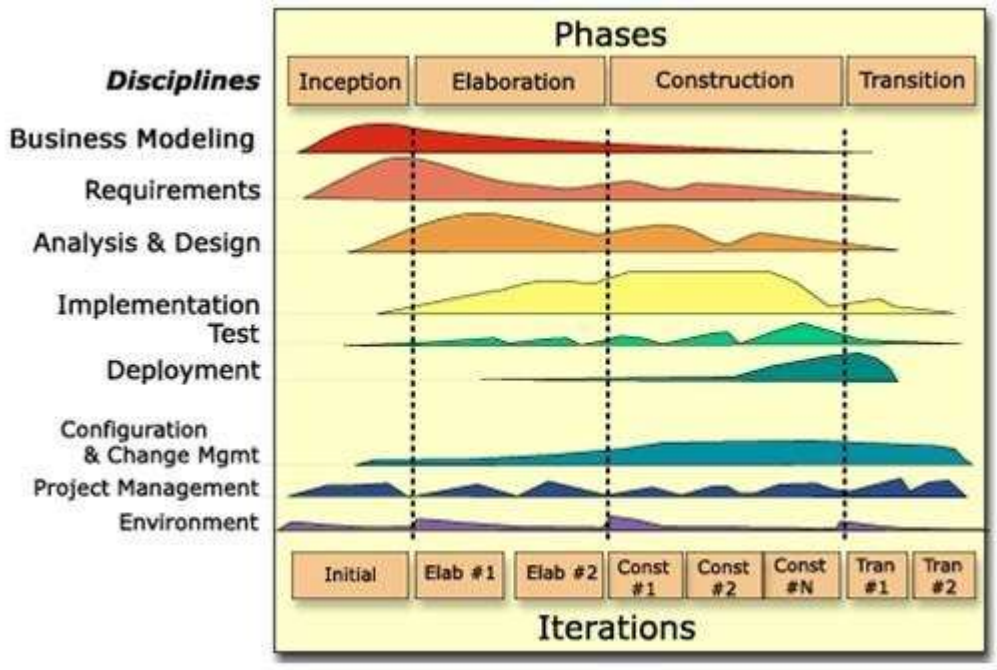
Figuur 5.1 Uitkomst van de Excel-sheet

Ik heb uiteindelijk gekozen voor het gebruiken van een ontwikkelmethode waar ik ervaring mee heb. Omdat wij Rational Unified Process (RUP) op school veel hebben gehad maar naar mijn mening het nooit goed hebben toegepast vond ik het een goed idee om te kijken of ik RUP voor dit project op de juiste manier kon toepassen. Hieronder zal ik in het kort omschrijven wat RUP precies inhoudt.

5.3.1 Rational Unified Process

Om RUP goed te gebruiken bij mijn project heb ik allereerst gekeken wat RUP precies inhoud. RUP is één van de meest gebruikte software ontwikkelmethoden. Uit het onderzoek van Klaas-Jan blijkt dat het na SDM de meest gebruikte is. Het is een iteratieve software ontwikkel methode die de hele software levenscyclus doorloopt, en ontwikkeld is door Rational Software Corporation (wat later is overgenomen door IBM). RUP is gebaseerd op aloude software ontwikkel principes zoals iteratief ontwikkelen en requirements driven. Andere belangrijke principes zijn het gebruik van visuele modellering (met UML) en constante verificatie van kwaliteit.

RUP is verdeeld in twee dimensies: (1) fasen, wat staat voor de 4 fasen dat een project doorloopt over tijd. En (2) disciplines, wat staat voor de activiteiten die plaatsvinden binnen het project. De vier fasen zijn: Inception, Elaboration, Construction en Transition. Elk van deze fasen heeft zijn eigen focus. Bijvoorbeeld, implementation (een van de disciplines) krijgt de meeste aandacht binnen de Construction fase. Daarentegen krijgen business modeling, requirements en analysis & design weer meer aandacht binnen de Elaboration fase. Alle disciplines vinden overigens wel elke fase plaats. De disciplines zijn: business modeling, requirements, analysis & design, implementation, test, deployment, configuration & change management, project management and environment. In figuur 5.2 is een visuele beschrijving van RUP te zien.



Figuur 5.2 Visuele beschrijving van RUP

5.4 Implementatie van RUP binnen mijn project

Hier wordt beschreven hoe ik RUP wilde gaan gebruiken binnen mijn project. Als eerst zal ik kort toelichten hoe ik RUP gebruikte bij oude projecten. Mijn idee was om met behulp van de lijst met functionaliteiten de risico's te identificeren. Zodra de risico's bekend waren heb ik een aanpak opgesteld. Hierin heb ik beschreven hoeveel iteraties ik ga doen binnen elke fase en wat ik per iteratie ga doen.

5.4.1 Implementatie van RUP bij oude projecten

In de tijd dat ik op school zat heb ik vaak bij projecten RUP gebruikt. Het probleem was alleen dat er geen iteraties werden gebruikt. Eigenlijk zou je kunnen zeggen dat we altijd waterval methode hebben gebruikt en het RUP hebben genoemd. Dit wou ik bij mijn project voorkomen en daarom heb ik eerst uitgezocht hoe iteraties werken en hoe ik die kan toepassen op mijn project.

5.5 Te gebruiken methoden & technieken

De technieken die gebruikt zijn zullen hier worden beschreven. Per techniek zal kort worden beschreven wat het inhoudt.

5.5.1 Unified Modeling Language (UML)

UML is een modelleertaal die ontworpen is door Grady Booch, James Rumbaugh en Ivar Jacobson. UML is ontworpen om objectgeoriënteerde ontwerpen voor een informatiesysteem te kunnen maken. Ik heb voor UML gekozen omdat dit een sterke relatie heeft met RUP.
De volgende diagrammen heb ik gebruikt van UML:

- Use case diagram
- Class diagram
- Sequence diagram

5.5.2 Test Management Approach (TMap)

TMap is een testaanpak ontwikkeld door Sogeti. Ik heb voor deze testaanpak gekozen omdat ik hier ervaring mee heb. Onder TMap vallen een aantal testtechnieken. De testtechniek die ik heb toegepast op mijn project is de algoritmetest.

5.5.3 Entity Relationship Diagram (ERD)

Een ERD diagram wordt gebruikt om de relaties aan te geven tussen entiteiten met behulp van visuele objecten. Dit is een representatief beeld van de database.

5.6 Aanpak van mijn project

In mijn plan van aanpak heb ik beschreven hoe ik mijn project ga aanpakken. Ik heb beschreven wat de aanpak was voor elke fase. Door per fase te beschrijven hoeveel iteraties erin komen en precies te beschrijven wat er elke iteratie moet worden bereikt wordt de lijn die door mijn project loopt een stuk

duidelijker voor mijzelf. Hieronder zal ik per fase beschrijven wat de aanpak was en waarom ik daarvoor had gekozen.

5.6.1 Inception fase

De inception fase is de opstartfase van mijn project. Omdat de lijst met functionaliteiten en de stakeholders al bekend waren heb ik er minder tijd aan besteed dan normaal. Ik heb gekozen om mij in deze fase vooral te richten op het plan van aanpak en het inlezen van de documentatie van Jorn van Veelen en Klaas-Jan Molendijk. Daarnaast heb ik gekozen om geen Vision document op te stellen omdat de lijst met functionaliteiten samen met het plan van aanpak hier voldoende inzage in biedt.

5.6.2 Elaboration fase

De elaboration fase heb ik opgedeeld in twee iteraties. Omdat bij het identificeren van de risico's naar voren kwam dat het classdiagram een probleem kon worden heb ik besloten om het classdiagram op zichzelf in één iteratie te realiseren. Hierdoor kon ik het eerste risico goed afvangen.

In de tweede iteratie zou de eerste functionaliteit van de SOM Selector worden ontwikkeld. Namelijk advies verkrijgen. Door deze functionaliteit ook in een aparte iteratie te ontwikkelen is de impact op het verdere project geminimaliseerd. Hierdoor kunnen de belangrijkste risico's worden afgevangen.

5.6.3 Construction fase

Bij de construction fase is gekozen om 2 iteraties op te nemen. In de eerste iteratie wordt de functionaliteit projectprofiel & advies opslaan ontwikkeld. Hiermee kunnen gebruikers hun advies opslaan en later inzien en adviezen van andere gebruikers inzien.

Voor de tweede iteratie zouden de functionaliteiten voor de adviesonderbouwing worden ontwikkeld (functionaliteiten 1.3.1 t/m 1.3.4). Deze functionaliteiten zijn namelijk even belangrijk als het advies verkrijgen zelf omdat je hierdoor inzicht krijgt in de onderbouwing van het advies.

5.6.4 Transition fase

De transition fase is ook opgedeeld in twee iteraties. De eerste iteratie draait om de overdacht en implementatie van de SOM Selector, de gebruikershandleidingen en de help functies. Voor de gebruikershandleidingen en de helpfuncties zal op basis van de gerealiseerde functionaliteiten tijd worden ingepland. Dit heb ik zo gedaan omdat het nog onduidelijk was hoe het project ging verlopen.

De tweede iteratie van de transition fase werd gericht op het onderzoek naar hoe de informatie uit de SOM Selector het beste kan worden gepresenteerd op een website. De informatie die beschikbaar is in de SOM Selector is op zichzelf al waardevol voor een gebruiker. Hier moet dus worden onderzocht of de informatie uit de SOM Selector bruikbaar is en hoe deze informatie het beste kan worden getoond. Dit zal resulteren in een rapport wat wordt opgeleverd aan de opdrachtgever.

5.6.5 Disciplines per fase

De disciplines van RUP zullen elke iteratie worden uitgevoerd. De producten binnen een discipline zullen ook groeien naarmate het project vordert. Omdat ik het project alleen heb uitgevoerd, zijn niet alle disciplines opgenomen.
Hieronder staat een lijst met de disciplines die relevant waren voor mij project met daarbij het resultaat van de discipline:

- Requirements → Software Requirements Specification
- Analysis & Design → Design model
- Implementation → Het ontwikkelde systeem
- Test → Test rapport
- Project management → Plan van aanpak en iteratieplannen
- Configuration management → Versiebeheer van bovenstaande producten.

5.7 Planning

De planning is opgesteld op basis van RUP. Voor elke fase zullen de iteraties worden ingepland. Per iteratie zal worden beschreven hoe lang een discipline moet gaan duren.

Daarnaast is er per fase één dag ingepland om aan het afstudeerverslag te werken.

De volgende planning is daardoor tot stand gekomen:

Elaboration fase 1e iteratie	week	dagen	Begindatum	Einddatum	
Requirements	49-50	5	6 december 2007	12 december 2007	
Analysis & design	50-52	10	12 december 2007	28 december 2007	
		15			
Elaboration fase 2e iteratie					
Requirements	1	2	2 januari 2008	4 januari 2008	
Analysis & design	2-3	6	7 januari 2008	14 januari 2008	
Implementation	3-4	6	14 januari 2008	22 januari 2008	
Test	4	3	22 januari 2008	24 januari 2008	
Afstudeerverslag	4	1	25 januari 2008	25 januari 2008	
		18			
Construction 1e iteratie					
Requirements	5	2	28 januari 2008	30 januari 2008	
Analysis & design	5-6	3	30 januari 2008	4 februari 2008	
Implementation	6-7	5	4 februari 2008	11 februari 2008	
Test	7	3	12 februari 2008	14 februari 2008	
Afstudeerverslag	7	1	15 februari 2008	15 februari 2008	
		14			
Transition 1e iteratie	8	5	18 februari 2008	22 februari 2008	
Transition 2e iteratie Onderzoek presentatie content	9	5	25 februari 2008	29 februari 2008	
Uitloop	10-11	10	3 maart 2008	14 maart 2008	
Afstudeerverslag	12-13	9	17 maart 2008	27 maart 2008	
Totaal geplande dagen		66			
Totaal geplande dagen + uitloop		76			

Afstudeerverslag

Tabel 5.7 Planning

6 Inception fase

Hier zal ik beschrijven hoe de uiteindelijke inception fase is doorlopen. Welke activiteiten zijn er gedaan binnen die fasen. Hoe ben ik tot die activiteiten gekomen.

6.1 Inlezen reeds bestaande documentatie

De documenten die beschreven zijn in hoofdstuk 4 uitgangssituatie zijn voor het grootste gedeelte in deze fase bestudeerd. Ook is de programmatuur van Jorn bekeken of er wellicht nog bruikbare onderdelen in zaten. Voor een duidelijke omschrijving van de documenten verwijs ik naar hoofdstuk 4. Hieronder zal ik beschrijven wat ik heb hergebruikt.

Functionele en niet-functionele eisen

Samen met de opdrachtgever is besloten dat deze ook gelden voor dit project. De functionele en niet-functionele eisen zijn opgenomen in de SRS. De SRS is terug te vinden in de externe bijlagen.

7 Elaboration fase – 1e iteratie

De eerste iteratie ging over het ontwikkelen van het class diagram. Als eerst heb ik het iteratieplan beschreven. In paragraaf 6.2 wordt de totstandkoming van het class diagram uitvoerig behandeld.

7.1 Iteratieplan

Het iteratieplan is voorafgaande aan de iteratie geschreven. Hierin staat beschreven wat de doelstellingen en risico's zijn voor deze iteratie. Daarnaast staat er een detailplanning voor deze iteratie in. Alle iteratieplannen zijn opgenomen in de externe bijlagen

7.2 Class diagram

Het class diagram is de basis voor de hele iteratie. Door dit te ontwikkelen ontstaat er een basis voor de rest van het systeem. Zodra het class diagram goedgekeurd was door de opdrachtgever kon de functionaliteit worden ontwikkeld. Het risico of het class diagram overeen kwam met het beslismodel was op dit punt ook belangrijk. Zodra het overeen kwam had het project ook een grotere kans van slagen. Het class diagram is gevalideerd door de opdrachtgever en Klaas-Jan.

7.2.1 Beslismodel en het Excel prototype als uitgangspunt

Voor het ontwerpen van het class diagram is de conceptuele en operationele beschrijving van het beslismodel en het Excel prototype als uitgangspunt genomen. De tijd die ik had was ook te kort om mij te richten op het beslismodel zelf. Ik ga er dus vanuit dat het beslismodel klopt en heb dit verder inhoudelijk niet beoordeeld.

Het Excel prototype is door Klaas-Jan ontwikkeld omdat hij zijn ontwikkelde beslismodel wilde valideren en demonstreren. Het Excel prototype kan een beperkt advies tonen. Hierdoor is dit een geschikt uitgangspunt voor de SOM Selector. Voor de ontwikkeling van het class diagram zal ik ook gebruik maken van de Excel sheet. De Excel sheet maakt gebruik van dezelfde terminologie als in het beslismodel en de berekening die wordt uitgevoerd volgt ook het algoritme uit het beslismodel.

7.2.2 Componenten van het class diagram

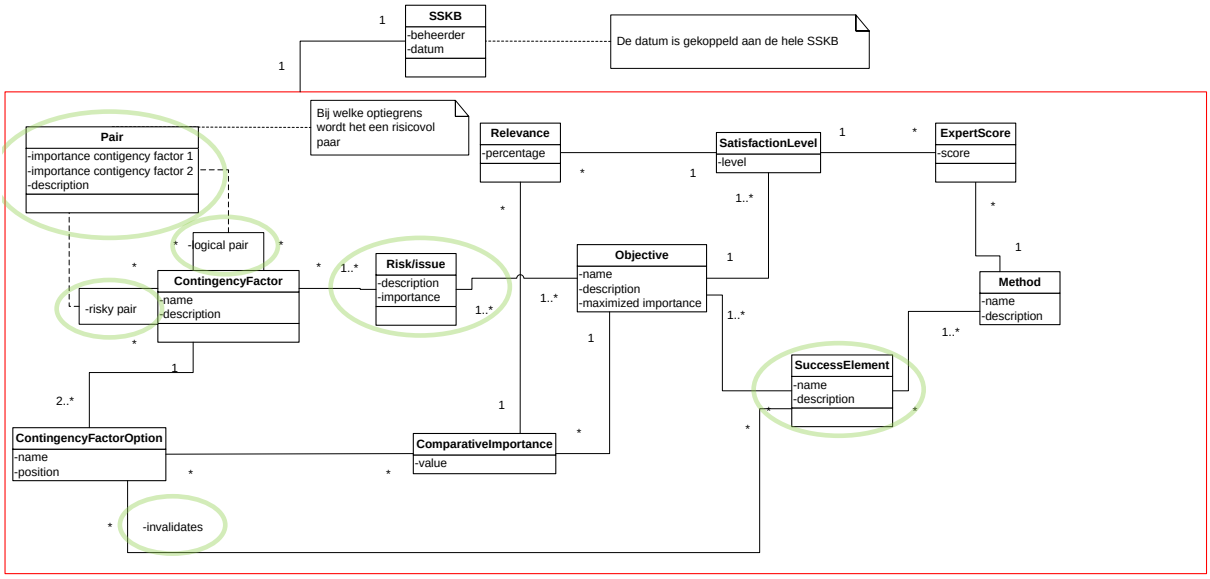
Het class diagram bestaat uit drie componenten. Genaamd SSKB, Project profile en Advice. Deze drie vormen samen het class diagram. In de volgende paragrafen zal per component een beschrijving worden gegeven. Daarbij zullen de relaties tussen de componenten ook worden beschreven.

7.2.3 SOM Selector Knowledge Base (SSKB)

De SSKB is een kennisdatabase over SOMmen waarmee de adviesberekening kan worden uitgevoerd. De bedoeling van de SSKB is dat de gegevens waarmee de berekening wordt uitgevoerd los staan van het advies. De inhoud van de SSKB kan namelijk nog veranderen. Bijvoorbeeld als er een nieuwe software ontwikkelmethode bij komt. Hierdoor is bij het opstellen van het class diagram rekening gehouden met de scheiding van de SSKB en het advies.

Om tot dit model te komen is er veel gecommuniceerd met de opdrachtgever. Die had aan het begin een ander beeld van de SSKB dan mij. Hij had namelijk het idee dat succes elementen, risico/issue en logische en risicovolle paren van projecteigenschappen invloed hebben op de berekening, terwijl deze als toelichting worden gebruikt. Dit beeld is ontstaan doordat hij de verschillende versies van het onderzoek van Klaas-Jan door elkaar haalde. Om dit beeld te weer goed te krijgen heb ik samen met Klaas-Jan en de opdrachtgever het class diagram besproken. Tijdens dit gesprek was de opdrachtgever het uiteindelijk eens met het class diagram wat ik had opgesteld.

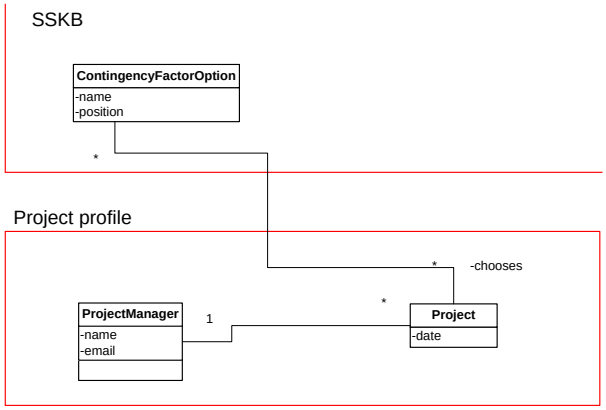
In figuur 7.1 is de structuur van de SSKB te zien in het class diagram. De groen omcirkelde classes hebben geen invloed op de berekening.



Figuur 7.1 De SSKB in het class diagram

7.2.4 Projectprofiel

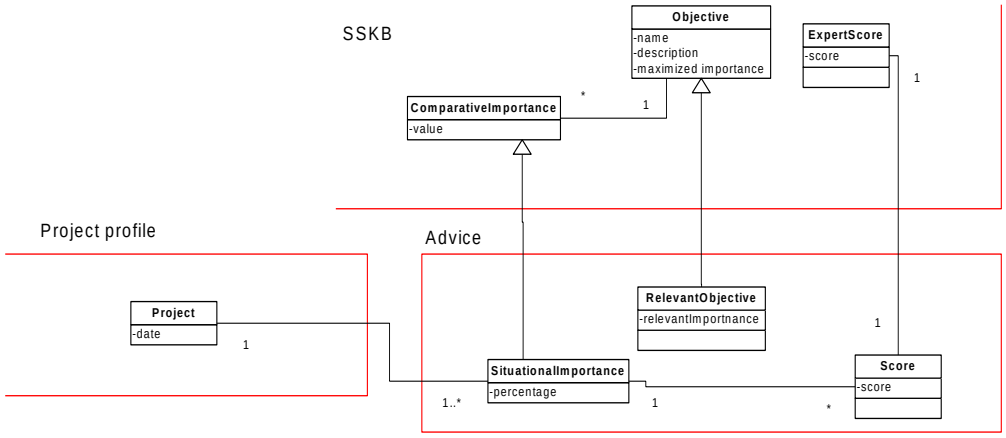
Het projectprofiel is de input van de adviesberekening, ingevoerd door de gebruiker. De gebruiker kwalificeert elke projecteigenschap door middel van een keuze uit vooraf gedefinieerde waarden. Het projectprofiel wordt gevormd door de gekwalificeerde projecteigenschappen. Aan dit profiel zal een projectleider worden gekoppeld, zodat een projectleider zijn profielen later kan inzien. In figuur 7.2 zal het gedeelte van projectprofiel en de relatie met de SSKB in het class diagram worden getoond:



Figuur 7.2 het projectprofiel in het class diagram

7.2.5 Advies

Het advies is de uitkomst van adviesberekening van de SOM Selector. Voor het berekenen van het advies zal gebruik worden gemaakt van de SSKB en het projectprofiel. Het advies bevat per doelstelling en per methode een tussenscore. Alle tussenscores per methode vormen het eindcijfer. In figuur 7.3 is een gedeelte van het class diagram te zien waarin de relaties tussen de onderdelen advies, SSKB en het projectprofiel worden getoond.



Figuur 7.3 De relaties tussen de onderdelen SSKB, advies en project profiel

7.2.6 Opties voor de relaties tussen SSKB, projectprofiel en advies

Voordat het definitieve class diagram was ontwikkeld ontstond het probleem hoe het advies en profiel konden worden opgeslagen. Hier moest namelijk rekening mee worden gehouden bij het opstellen van het class diagram. Hieronder zal ik de opties die als oplossing kunnen dienen beschrijven met een omschrijving waarom er wel of niet voor gekozen is.

Optie 1: Het advies wordt naar een pdf bestand geschreven. Het profiel wordt opgeslagen in de database.

Door het advies te tonen in een pdf bestand wordt de opdracht een stuk ingewikkelder, Bij deze optie zou moeten worden uitgezocht hoe het opbouwen van een pdf bestand in elkaar zit. Dit zou teveel tijd kosten. Veel features kunnen hiermee niet gerealiseerd worden.

Optie 2: Advies én profiel worden opgeslagen in de database.

Door het advies op te slaan in de database zal het advies altijd hetzelfde blijven, ook als de SSKB in de toekomst verandert. Als het advies dan wordt getoond, dan zal de berekening niet opnieuw hoeven te worden uitgevoerd.

Optie 3: Alleen Profiel wordt opgeslagen in de database. Het advies zal elke keer opnieuw worden berekend.

Door alleen het profiel op te slaan worden er geen dubbele gegevens opgeslagen. Alleen wordt het advies elke keer opnieuw berekend, wat betekent dat als de SSKB zou veranderen het advies ook anders zou zijn. Een nadeel is dat een overzicht van alle adviezen met de bijbehorende scores wat lastiger te tonen is. Omdat per advies de berekening moet worden uitgevoerd.

Uiteindelijk heb ik in samenspraak met de opdrachtgever gekozen voor optie 3. Dit was naar mijn mening de beste oplossing omdat de SSKB naar verloop van tijd wellicht heel anders kan zijn.

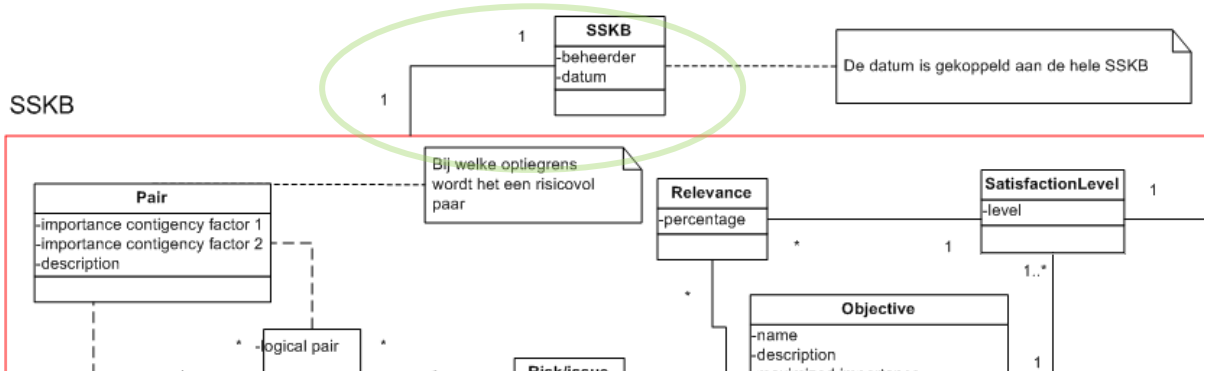
7.2.7 Veranderende SSKB

Omdat door bijvoorbeeld nieuwe inzichten in ontwikkelmethoden de SSKB kan veranderen moest er dus een bepaalde vorm van versiebeheer in komen om te zorgen dat de adviezen onveranderd bleven. In overleg met de opdrachtgever zijn we tot twee opties gekomen. Daarna is er een keuze gemaakt voor de manier waarop het versiebeheer wordt toegepast. Hieronder zal worden getoond wat de opties waren en voor welke optie uiteindelijk gekozen is.

Opties van versiebeheer voor de SSKB:

- Volledige kopie van de SSKB naar een nieuwe database.
- Versienummer gekoppeld aan elke databasetabel van de SSKB

Het voordeel van een volledige kopie van de SSKB is dat het geen invloed zou hebben op het class diagram. Daarentegen moet de SOM Selector met meerdere databases om kunnen gaan en zou de opdrachtgever . Om een versienummer op te nemen krijg je het probleem dat elke class gekoppeld moet zijn aan het versienummer. Uiteindelijk heb ik in samenspraak met de opdrachtgever voor het laatste gekozen omdat het toevoegen van een versienummer in elke class goed implementeerbaar is.



Figuur 7.4 Versiebeheer toegepast in het class diagram

In figuur 7.4 is de uiteindelijke implementatie van versiebeheer in de SSKB te zien. Door de datum waarop de SSKB is aangemaakt kunnen oude adviezen worden ingeladen.

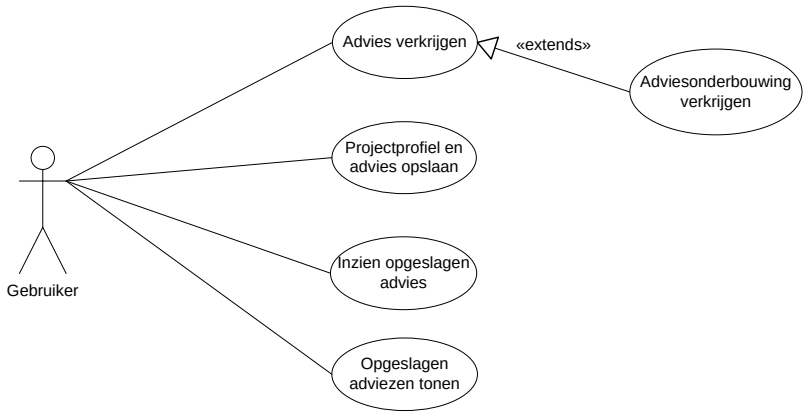
8 Elaboration fase – 2e iteratie

De tweede iteratie van de elaboration fase ging om het ontwikkelen van functionaliteit 1.1: Advies verkrijgen. Binnen deze iteratie heb ik een aantal activiteiten uitgevoerd. Ik zal per activiteit beschrijven wat ik heb gedaan en welke keuzes ik heb gemaakt.

8.1 Het use case diagram opstellen

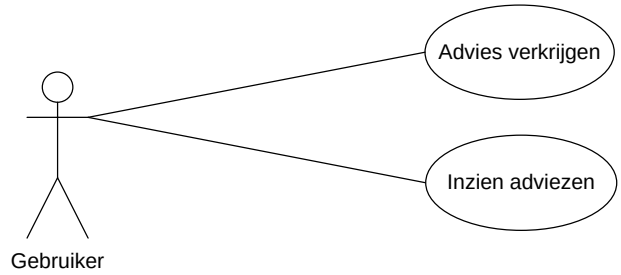
Bij het opstellen van het use case diagram liep ik tegen een aantal problemen op. Het eerste probleem was of ik de use cases niet teveel opsplitste en het tweede probleem was of ik de use cases wel op de juiste manier beschreef.

De eerste versie van het use case diagram zag er als volgt uit:



Figuur 8.1 het eerste use case diagram

Dit use case diagram is te uitgebreid. Ik had dit gedaan omdat ik dacht dat er per webpagina die ik ontwikkelde een use case moest zijn. Hierdoor werd niet duidelijk wat de eindgebruiker uiteindelijk wil doen met het systeem. Ik heb er daarna voor gekozen om het use case diagram te verduidelijken zodat het beter naar voren komt wat de eindgebruiker uiteindelijk doet. Dit heb ik gedaan door middel van meerdere use cases bij elkaar te voegen tot één use case. Bij de use case omschrijving heb ik een alternatief verloop opgenomen om de verschillende paden van de samengevoegde use cases te beschrijven. Uiteindelijk ben ik tot het volgende use case diagram gekomen.



Figuur 8.2 het uiteindelijke use case diagram

Voor de aanvullende eisen van een use case heb ik besloten om als extra hoofdstuk op te nemen binnen de use case omschrijving. Ik wou dit binnen de use case zelf houden omdat het hierdoor overzichtelijk is.

8.2 Use case: Advies verkrijgen

In tabel 8.1 staat de beschrijving van de use case: Advies verkrijgen.

Actors

- Project manager

Pre-condities

- Geen

Standaard verloop

1. Het systeem toont de pagina met projecteigenschappen.
2. De actor vult zijn projectprofiel in en kiest voor “Advies verkrijgen”.
3. Het systeem berekent het advies.
4. Het systeem toont het advies op het scherm met behulp van scores van doelstellingen en methodes, en een totaalscore per methode.
5. Use case stopt.

Uitzonderingen

- Geen

Post-condities

- Het advies is getoond op het scherm.

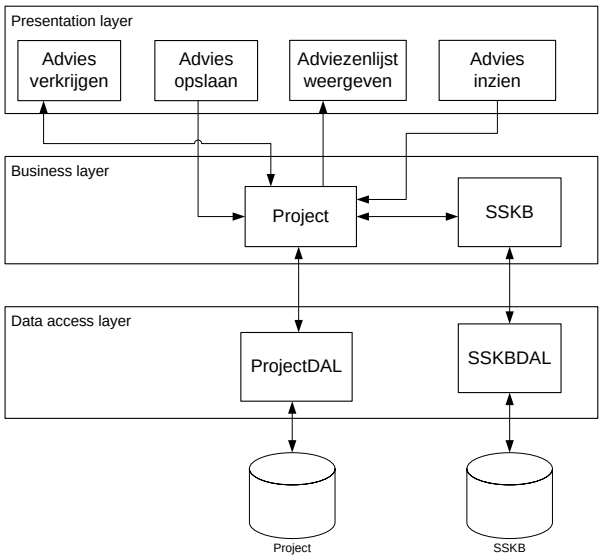
Aanvullende eisen

Het advies zal als een tabel worden getoond met daarin rond de 21 objectives, gesorteerd op de belangrijkste. Bij elk objective zal een deelscore te zien zijn voor een bepaalde methode. Door alle deelscores per methode op te tellen kom je op de uiteindelijke eindscore. De eindscore zal onderaan komen te staan.

Tabel 8.1 Use case: Advies verkrijgen

8.2.1 Het kiezen van een architectuur

Bij het kiezen van de architectuur heb ik goed gekeken naar de toekomst van de SOM Selector omdat de SOM Selector wellicht in de toekomst als Web-part of Webservice wordt gebruikt. Er moest dus rekening mee worden gehouden dat de business laag los stond van wat er op het scherm te zien is, omdat als er een nieuwe presentatie laag bij zou komen dit implementeerbaar moet zijn zonder de business laag aan te passen. Daarnaast is er gekozen voor een data access laag zodat de database gescheiden is van de business laag. Ik heb hiervoor gekozen omdat de SOM Selector nog in ontwikkeling is en de kans groot is dat op beide lagen wijzigingen zullen komen. Hierdoor zal een wijziging in de presentatie en data access laag niet een directe impact hebben op de business laag. In figuur 8.3 is de architectuur te zien in een diagram.



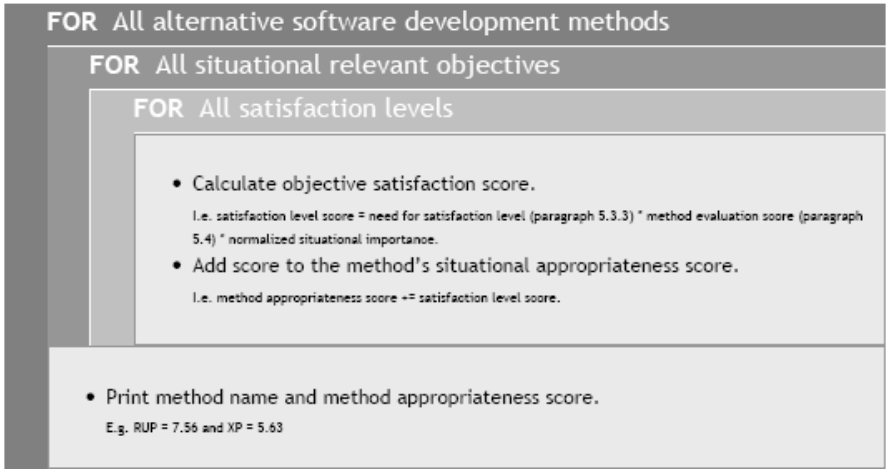
Figuur 8.3 Architectuur

Voor de scheiding binnen de business laag tussen Project en SSKB is gekozen omdat de SSKB zich expliciet richt op het berekenen van een advies en Project zich richt op het projectbeheer. Door deze scheiding komt de grens tussen Project en SSKB een stuk zichtbaarder naar voren en zorgt dit voor overzicht.

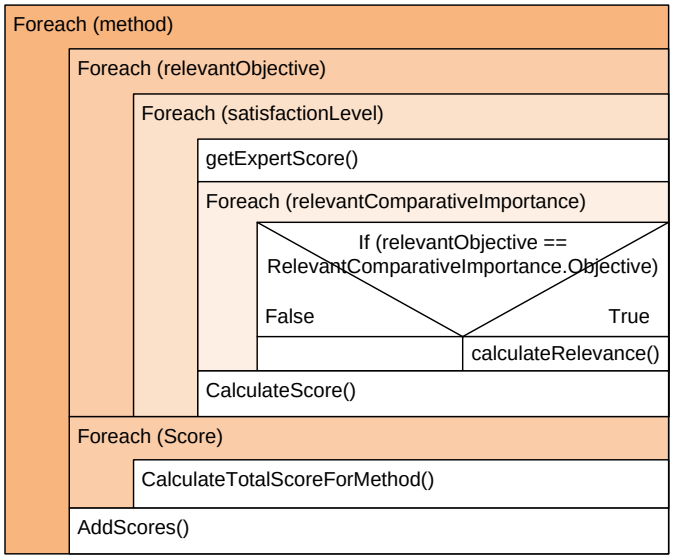
Dezelfde scheiding is ook opgenomen in de data access laag omdat hierdoor Project en SSKB niet hetzelfde data acces layer object gebruiken. Ik heb hiervoor gekozen omdat als er een nieuwe SSKB komt dit geen invloed heeft op Project.

8.3 Het beschrijven van het algoritme voor het berekenen van het advies

Het algoritme wat gebruikt wordt om een advies te berekenen wordt beschreven in paragraaf 4.1. In de thesis van Klaas-Jan wordt het algoritme in een Programma Structuur Diagram (psd) formaat beschreven. Omdat dit niet direct implementeerbaar is had ik besloten om een nieuwe psd voor de SOM Selector te maken en het verloop te beschrijven wat in de programmatuur gebruikt ging worden. In de onderstaande figuren (8.4 & 8.5) zijn beide psd's te zien.



Figuur 8.4 PSD onderzoek naar ontwikkelmethoden

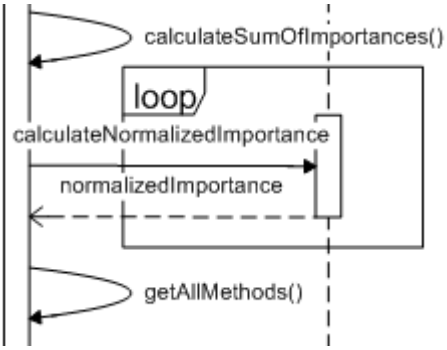


Figuur 8.5 PSD voor implementatie van de SOM Selector

Door deze vertaalslag te maken, zorgde ik ervoor dat het algoritme door mij implementeerbaar is.

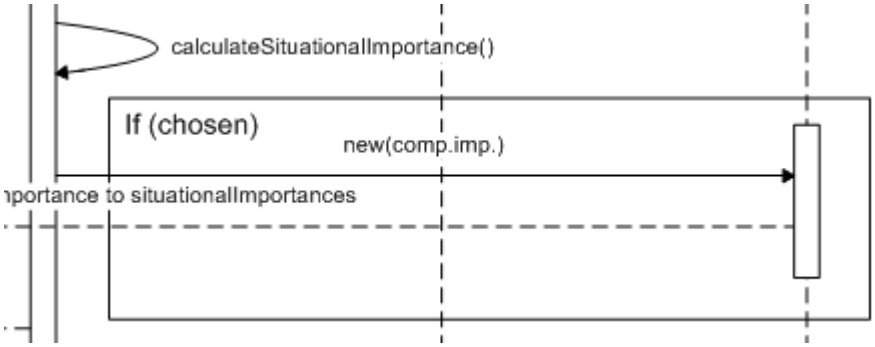
8.4 Het sequence diagram opstellen voor de use case “Advies verkrijgen”

Nadat het use case diagram is opgesteld heb ik een sequence diagram gemaakt voor de use case “Advies verkrijgen”. Omdat er maar één use case is die relevant is voor deze iteratie heb ik alleen deze use case vertaald naar een sequence diagram. Omdat het maken van een sequence diagram op zichzelf niet relevant is voor het afstudeerproces beschrijf ik waarom ik bepaalde technieken toepas op het sequence diagram zodat het een logisch diagram wordt. Het gehele sequence diagram is terug te zien in het design model in de externe bijlage.



Figuur 8.6 Voorbeeld van een loop in een sequence diagram

In figuur 8.5 ziet u een voorbeeld van een loop. Dit heb ik gebruikt in het sequence diagram omdat ik hierdoor duidelijk kon zien wat herhaald werd. Hierdoor kon het algoritme duidelijk worden vertaald in het sequence diagram.

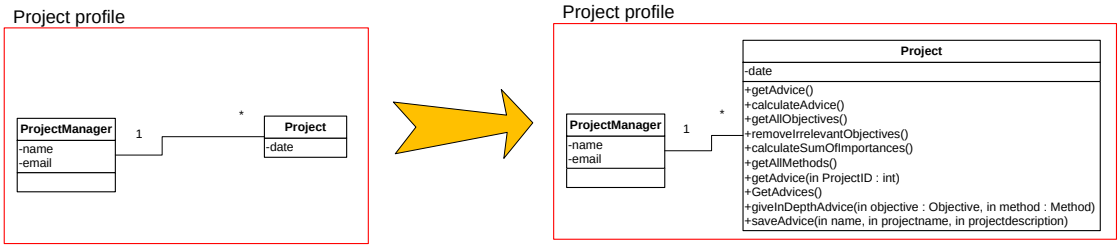


Figuur 8.7 voorbeeld van een "if" in een sequence diagram

Door gebruik te maken van een IF kan het verloop alleen kan worden doorlopen zodra het argument voldoet aan het criterium. Omdat in het algoritme een aantal keren de if-statements voorkomen heb ik deze opgenomen in het sequence diagram zodat het duidelijk werd wat het verloop was. Hierdoor kan het sequence diagram goed worden gebruikt als beschrijving van de programmatuur.

8.5 Implementatie class diagram maken

Het class diagram is in deze iteratie geëvolueerd naar een implementatie class diagram. Bij het opstellen van het sequence diagram zijn methoden ontstaan die bij een specifieke class horen. Dit houdt in dat de methodes die bij een class horen ook in het model worden weergegeven. In figuur 8.8 zal de evolutie te zien zijn wat het class diagram ondervond.

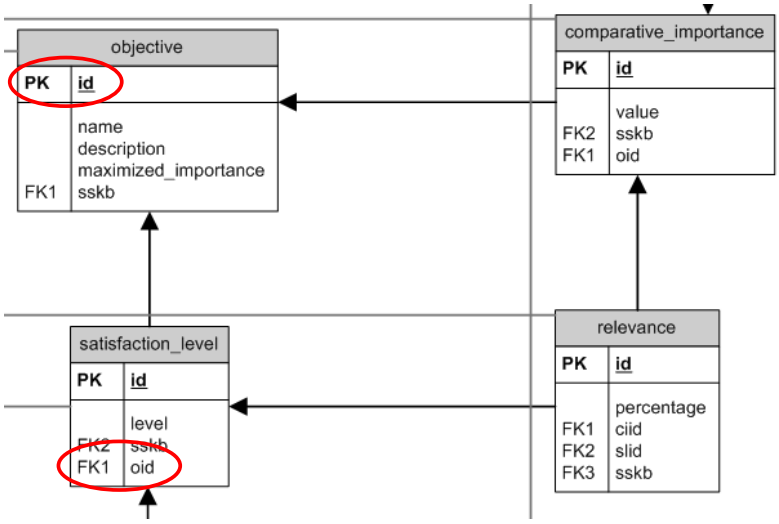


Figuur 8.8 Van class diagram naar implementatie class diagram

8.6 Het ERD diagram maken

Het ERD diagram wordt gebruikt om de vertaling te maken van het class diagram naar database niveau. Bij het maken van dit diagram zijn een aantal punten naar voren gekomen. Hieronder zal ik beschrijven welke keuzes ik heb gemaakt bij de ontwikkeling van het ERD diagram. Ik zal delen van het ERD diagram gebruiken als verduidelijking van de gemaakte keuzes.

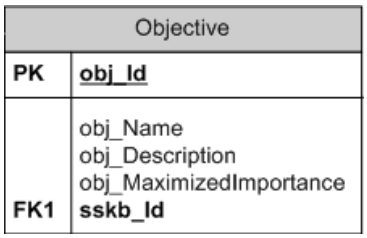
Bij de ontwikkeling van het ERD diagram merkte ik dat elke tabel een vreemde sleutel had en dat de naamgeving daarvan telkens onduidelijk werd. Zodra het ERD diagram zou worden uitgebreid kon dit problemen gaan veroorzaken. In figuur 8.9 staat de eerste versie van het ERD diagram.



Figuur 8.9 eerste versie van het ERD diagram

Zoals te zien is bij de rood omcirkelde attributen, verwijst de vreemde sleutel “oid” in “satisfaction_level” naar de primaire sleutel “id” in “objective”. Maar als er nu een extra tabel bij zou komen met de beginletter o (bijvoorbeeld “overall”) dan zou daar een andere schrijfwijze voor moeten worden verzonnen. Als ik mijn schrijfwijze aan zou houden, zou de verwijzing naar de primaire sleutel eveneens “oid” moet heten. Hierdoor zou ik dus moeten afwijken van de schrijfwijze die werd toegepast.

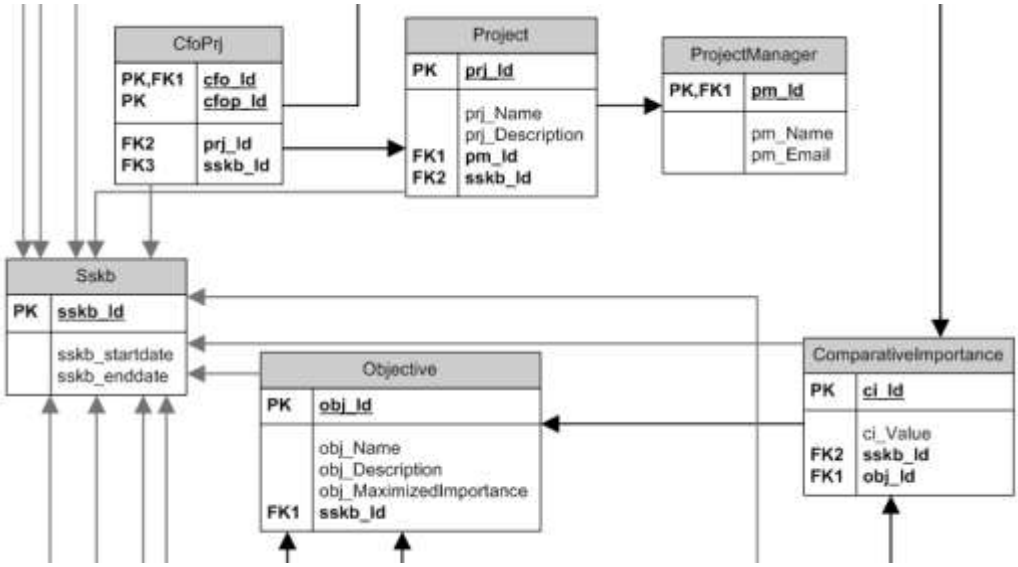
Omdat ik dit wilde verduidelijken ben ik erachter gekomen dat er een SQL standaard is. Dit houdt in dat de naamgeving voor elk attribuut van een tabel aan het volgende moet voldoen: <afkorting van maximaal 4 letters van de tabelnaam>_<attribuut>. In figuur 8.10 is het doorgevoerde SQL standaard te zien in de tabel “Objective”.



Figuur 8.10 De tabel "Objective" met het SQL standaard

Door aan te houden aan het SQL standaard ben ik in staat geweest een eenduidig ERD diagram op te stellen.

Om het versiebeheer van de SSKB op te nemen in het ERD diagram is er een tabel SSKB gedefinieerd. Deze tabel bevat een begin en een einddatum. Om de rest van de tabellen aan deze tabel te koppelen is voor alle tabellen een extra attribuut “sskb_Id” opgenomen als vreemde sleutel die verwijst naar de tabel “Sskb”. In figuur 9.11 is te zien hoe dit is geïmplementeerd.



Figuur 8.11 Versiebeheer toegepast in het ERD diagram

8.7 Het ontwikkelen van de use case “Advies verkrijgen”

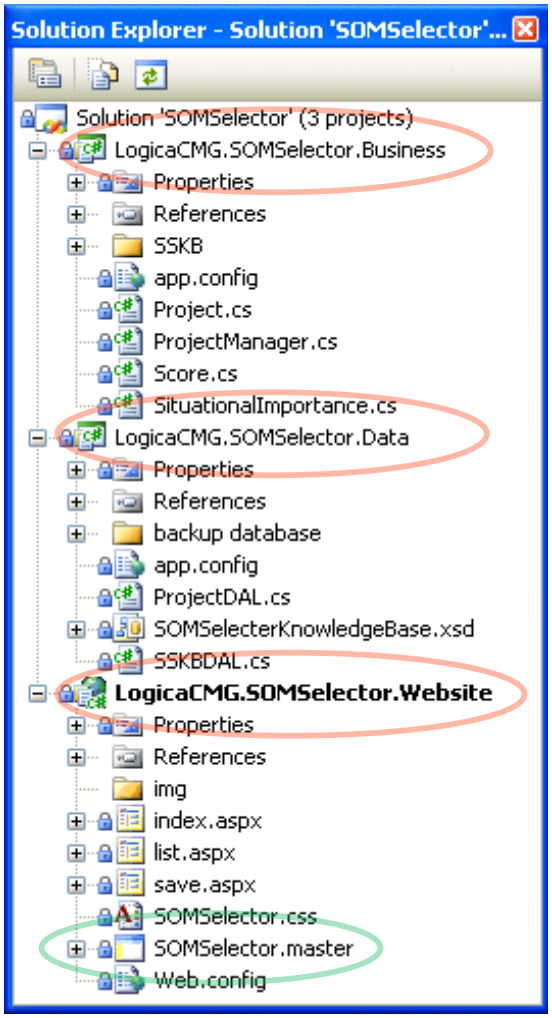
Nadat het ERD diagram is afgerond ben ik de use case “Advies verkrijgen” gaan ontwikkelen. Deze activiteit zal ik opdelen in een aantal delen. Allereerst zal ik uitleggen hoe ik de ontwikkelomgeving heb opgezet. Daarna zal ik beschrijven hoe ik de architectuur heb toegepast in de ontwikkelomgeving. Vervolgens zal ik vertellen welke problemen ik ben tegengekomen bij het programmeren. Aan het eind zal het resultaat van deze iteratie worden getoond.

Voor de ontwikkeling van de use case “Advies verkrijgen” heb ik gebruik gemaakt van Microsoft Visual Studio 2005 Express Editie. Deze versie week af van de versie waarmee ik heb gewerkt. Ik heb dit besproken met mijn begeleider en die heeft mij uitgelegd wat er nieuw is in de versie. De Express

Editie installeert ook direct een SQL Server. Door middel van Microsoft Visual SourceSafe 2005 heb ik versiebeheer toegepast. Working Tomorrow heeft een SourceSafe server staan waar de sourcecode kan worden opgeslagen.

Omdat de SSKB bestaat uit een vaste gegevensverzameling moesten deze gegevens ook in de database worden gezet. Ik heb gekozen om dit handmatig te doen omdat het niet mogelijk is om alle gegevens uit de Excel sheet met behulp van een script te kopiëren. Voor de SSKB zal in een volgend project ook de beheerfunctionaliteit moeten worden ontwikkeld zodat de beheerder een nieuwe versie niet op databaseniveau hoeft in te voeren.

Voor het implementeren van de architectuur heb ik gekozen om gebruik te maken van de technieken die Visual Studio biedt. Dit bied voldoende Hieronder ziet u in figuur 8.12 hoe de architectuur wordt weergegeven in Visual Studio.



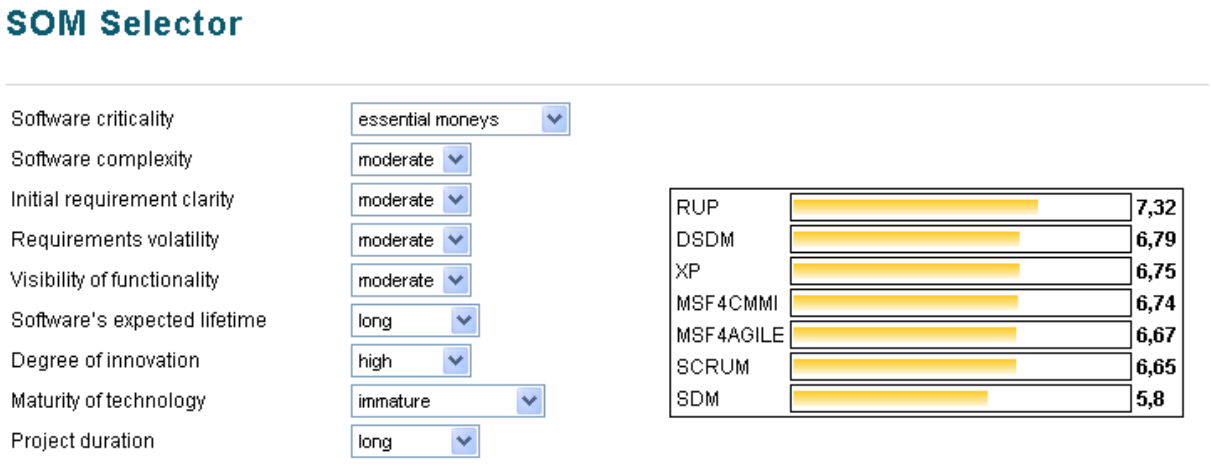
Figuur 8.12 De architectuur weergegeven in Visual Studio

Door middel van projecten kun je duidelijk de architectuur aangeven. De rood omcirkelde namen zijn 3 verschillende projecten die ook herleiden naar de verschillende lagen in de architectuur. Ook heb ik gekozen voor het gebruik van een master page (groen omcirkeld). Dit zorgt voor een bovenliggende laag over de andere pagina's. Hierdoor kan de lay-out die voor elke pagina hetzelfde is op 1 plaats worden beheerd.

Omdat in deze iteratie niet alle functionaliteit wordt gebouwd, heb ik besloten om de irrelevante klassen nog niet te programmeren. Hierdoor wordt er tijd bespaard met het programmeren. De irrelevante klassen zullen worden geprogrammeerd zodra de functionaliteit waarin ze worden gebruikt wordt ontwikkeld.

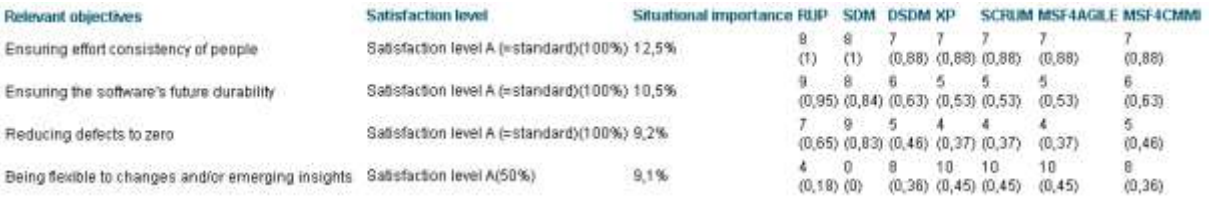
Tijdens het programmeren kwam ik er achter dat je in Visual Studio gebruik kunt maakt van TableAdapters. Een TableAdapter maakt een functie aan die gekoppeld kan worden aan een query. Deze functie kan worden aangeroepen in de programmeercode. Omdat er hierdoor een scheiding is tussen de business laag en de data laag heb ik besloten om de TableAdapters te gebruiken als data access layer.

Het uiteindelijke resultaat wat ik heb geprogrammeerd is hieronder te zien:



Figuur 8.13 Het invoeren van een projectprofiel in de SOM Selector

In figuur 8.13 is te zien hoe het projectprofiel kan worden ingevoerd. Daarnaast staat het resultaat van het aangevraagde advies. Het projectprofiel is overigens niet compleet weergegeven.



Relevant objectives	Satisfaction level	Situational importance	RUP	SDM	DSDM	XP	SCRUM	MSF4AGILE	MSF4CMMI
Ensuring effort consistency of people	Satisfaction level A (=standard)(100%)	12,5%	8 (1)	8 (1)	7 (0,88)	7 (0,88)	7 (0,88)	7 (0,88)	7 (0,88)
Ensuring the software's future durability	Satisfaction level A (=standard)(100%)	10,5%	9 (0,95)	8 (0,84)	6 (0,63)	5 (0,53)	5 (0,53)	5 (0,53)	6 (0,63)
Reducing defects to zero	Satisfaction level A (=standard)(100%)	9,2%	7 (0,65)	9 (0,83)	5 (0,46)	4 (0,37)	4 (0,37)	4 (0,37)	5 (0,46)
Being flexible to changes and/or emerging insights	Satisfaction level A(50%)	9,1%	4 (0,18)	0 (0)	8 (0,36)	10 (0,45)	10 (0,45)	10 (0,45)	8 (0,36)

Figuur 8.14 Opbouwing van het advies

In figuur 8.14 is te zien hoe het advies wordt opgebouwd. Dit is gesorteerd op de belangrijkheid van de doelstelling. Met behulp van de getallen tussen de haakjes zal per methode het eindcijfer worden berekend.

8.8 Het testen van de ontwikkelde software

Bij het testen van de ontwikkelde use case ben ik tegen een aantal problemen gelopen. Hieronder zal ik beschrijven hoe ik tot het testrapport ben gekomen en welke testtechnieken ik heb gebruikt.

Voor het opstellen van het testrapport heb ik gebruik gemaakt van TMap. Hierin wordt duidelijk beschreven wat elke testtechniek inhoud. Ik heb er voor gekozen om het testen kleinschalig te houden. Het was naar mijn mening niet relevant om een zeer uitgebreid master test plan te schrijven. De ontwikkelde use case was op functioneel gebied “te klein” om te testen.

Omdat het advies op basis van een algoritme werd berekend, heb ik gekozen voor een algoritmetest. Deze test bewijst of alle mogelijke paden worden doorlopen in je programma. Een pad kan bijvoorbeeld 2 kanten op gaan bij een IF-statement, dit noemen we een beslispunt. Als het statement waar is ga naar pad 1, als het statement onwaar is ga naar pad 2. Om de algoritmetest uit te voeren op het algoritme moesten eerst alle testpaden worden beschreven.

Bij het beschrijven van de testpaden werd duidelijk dat het algoritme maar één beslispunt had. Het algoritme wordt grotendeels altijd op dezelfde manier doorgelopen. Hierdoor kon de algoritmetest niet als bruikbare test worden gebruikt. Omdat de Excel sheet ook gebruik maakt van het algoritme heb ik besloten om te testen of de Excel sheet overeen komt met de SOM Selector.

Door middel van deze vergelijkingstest kon ik zien of de Excel-sheet dezelfde uitkomsten geeft bij het advies verkrijgen als de SOM Selector. Als ik overigens alle verschillende mogelijkheden van uitkomsten zou willen testen dan zou ik veel te lang bezig zijn. Ik heb hierom besloten om per type doelstelling te testen. Er zijn 4 typen doelstellingen in de SSKB. Door voor elk van deze typen een testcase te specificeren kan er met behulp van 4 testcases het gehele systeem worden getest. In tabel 8.2 ziet u een voorbeeld van een testcase.

Reducing defects to zero

Softw. criticality	life	essential moneys	discretionary moneys	comfort
Relative importance	1,0	0,7	0,0	0,0
Satisfaction level A (=standard)	100%	100%	0%	0%

Tabel 8.2 Voorbeeld van een testcase

Hierboven ziet u een doelstelling (Reducing defects to zero) met 1 projecteigenschap genaamd “Software criticality” en één bevredigingslevel namelijk level A. Ik heb er voor gekozen om voor alle

irrelevante projecteigenschappen bij deze testcase altijd de eerste optie te selecteren. Deze lijst is terug te vinden in het testrapport (externe bijlage). Om deze testcase een positieve uitkomst te laten zien moeten dus de uitkomsten voor elke methode bij een bepaalde optie gelijk zijn voor de SOM Selector en de Excel sheet. In tabel 8.3 ziet u het resultaat van de testcase.

Reducing defects to zero	**Uitkomst Excel sheet**							**Uitkomst SOM Selector**						
Software Criticality	**RUP**	**SDM**	**DSDM**	**XP**	**Scrum**	**MSF4A**	**MSF4C**	**RUP**	**SDM**	**DSDM**	**XP**	**Scrum**	**MSF4A**	**MSF4C**
Life	6.88	5.86	6.70	6.86	6.74	6.76	6.64	6.88	5.86	6.70	6.86	6.74	6.76	6.64
Essential moneys	6.88	5.70	6.81	7.03	6.90	6.92	6.74	6.88	5.70	6.81	7.03	6.90	6.92	6.74
Descretionary moneys	6.81	5.18	7.13	7.55	7.41	7.43	7.06	6.81	5.18	7.13	7.55	7.41	7.43	7.06
Comfort	6.69	5.03	7.19	7.69	7.54	7.56	7.11	6.69	5.03	7.19	7.69	7.54	7.56	7.11

Tabel 8.3 Het resultaat van een testcase

In dit voorbeeld komen alle waarden overeen waardoor de uitkomst van de test de correcte werking van de software bevestigt.

9 Construction fase – 1e iteratie

De construction fase ging om het ontwikkelen van de functionaliteiten “Advies opslaan” en “Opgeslagen adviezen inzien”. Binnen deze iteratie heb ik een aantal activiteiten uitgevoerd. Ik zal per activiteit beschrijven wat ik heb gedaan en welke keuzes ik heb gemaakt. De bedoeling was eigenlijk om twee iteraties te doen. Hier ben ik uiteindelijk in overleg met de opdrachtgever vanaf gestapt omdat de prioriteit van de opdrachtgever bij deze functionaliteiten lag.

9.1 De use case: inzien adviezen opstellen

Omdat ik in de tweede iteratie van de elaboration fase use case “Advies verkrijgen” heb ontwikkeld, kon ik mij in deze iteratie richten op de use case “Inzien adviezen” en het alternatieve verloop “Advies opslaan” van de use case “Advies verkrijgen”. Dit betekent ook dat de use case beschrijving ook in deze iteratie komt. Hierdoor worden alleen de functionaliteiten beschreven die relevant zijn voor deze iteratie.

De use case “inzien adviezen” zorgt ervoor dat je opgeslagen adviezen kan inzien. In tabel 9.1 zal de use case beschrijving te zien zijn.

Actors

- Project manager

Pre-condities

- Er is minimaal 1 advies opgeslagen

Standaard verloop

1. De actor kiest voor “opgeslagen adviezen weergeven”.
2. Het systeem laat een lijst zien met alle opgeslagen adviezen.
3. De actor klikt op een advies uit de lijst.
4. Het systeem toont het advies op het scherm.

Uitzonderingen

- Na stap 3 kan de volgende uitzondering in werking worden gesteld:
  - Er zijn geen adviezen opgeslagen, er zal een melding worden gegeven dat er geen adviezen zijn.

Post-condities

- Het advies is getoond op het scherm

Aanvullende eisen

Er zal een lijst komen met de naam van de projectmanager, de projectnaam en de projectomschrijving. Zodra daarop geklikt wordt zal het advies worden weergegeven.

De lijst zal automatisch worden gesorteerd op Project manager. De actor kan daarna nog kiezen om te sorteren op Project.

Tabel 9.1 De use case beschrijving van "Inzien adviezen"

In de eerste versie van deze use case had ik in het standaard verloop de actor-naam opgenomen. Dit heb ik later vervangen door het woord actor, omdat hierdoor de actor op 1 plek kon worden gewijzigd. Dit verbeterd de leesbaarheid van de use case beschrijving.

In samenspraak met de opdrachtgever is er voor gekozen om de adviezen voor iedereen openbaar te maken. Hierdoor kunnen gebruikers zien welke projecten er zijn ingevoerd en wat het advies was voor dat project. Hier is voor gekozen omdat de opdrachtgever meer prioriteit geeft aan het opslaan en het inzien dan aan autorisatie en authenticatie.

9.2 De use case: advies verkrijgen aanpassen

De use case “Advies verkrijgen” is in de tweede iteratie van de elaboration fase opgesteld. In deze iteratie is er extra functionaliteit bij gekomen waardoor de use case “Advies verkrijgen” moest worden aangepast. Hieronder zal ik beschrijven wat er is veranderd en waarom.

Alternatief verloop

- **Adviesonderbouwing verkrijgen**

Dit alternatieve verloop kan vanaf actie 4 in werking worden gesteld.

1. De actor selecteerd een score van een doelstelling en methode.
2. Het systeem zal de succeselementen laten zien per objective en methode. Dit zal te zien zijn als een ballon met tekst zodra de actor de score selecteert
3. Use case vervolgt bij stap 4.

- **Projectprofiel opslaan**

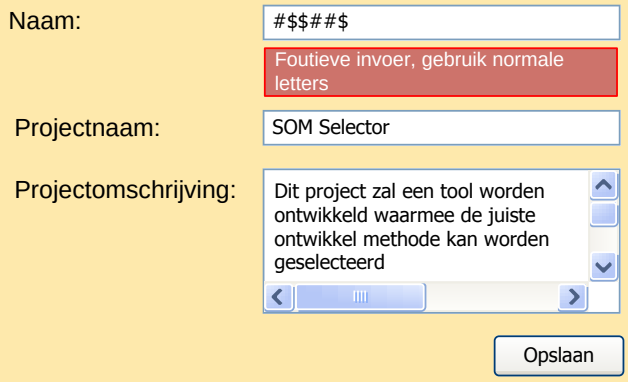
Dit alternatieve verloop kan vanaf actie 4 in werking worden gesteld.

1. De actor kiest voor projectprofiel opslaan.
2. Het systeem toont de invoervelden.
3. De actor vult zijn naam, email, projectnaam en projectbeschrijving in en drukt op de knop advies opslaan,
4. Het systeem slaat het projectprofiel op.
5. Use case vervolgt bij stap 4.

Uitzonderingen

In stap 3 kan de volgende uitzondering in werking worden gesteld:

- Een of meerdere invoervelden zijn leeg of met rare tekens ingevoerd. Er zal een foutmelding worden weergegeven en de actor kan het opnieuw proberen. Hieronder zal een voorbeeld worden gegeven hoe een foutmelding eruit ziet.



In stap 4 kan de volgende uitzondering in werken worden gesteld:

- Het advies wordt met dezelfde gegevens opgeslagen. Er zal een melding worden gegeven dat het advies niet kan worden opgeslagen.

Tabel 9.2 Het alternatieve verloop in de use case beschrijving van "Advies verkrijgen"

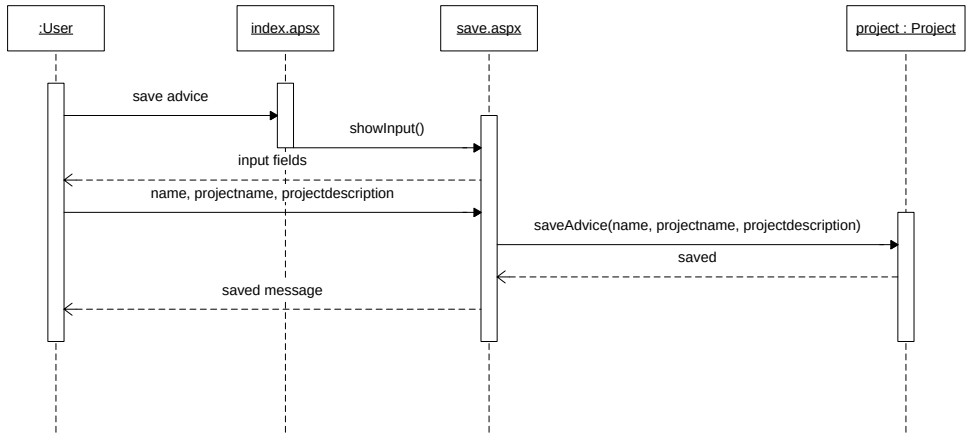
In tabel 9.2 ziet u het alternatieve verloop van de use case “Advies verkrijgen”. Een alternatief verloop is wanneer je afwijkt van het standaard verloop in de use case. Ik heb gekozen om dit als alternatieve verloop op te nemen omdat dit een onderdeel is van het verkrijgen van een advies. Naar mijn idee hoort het opvragen van adviesonderbouwing en het opslaan van het projectprofiel bij het advies verkrijgen. Hierdoor is het direct duidelijk wat de verschillende paden zijn binnen deze use case.

In overleg met de opdrachtgever is gekozen om het inloggen niet op te nemen in het systeem. De opdrachtgever gaf hier minder prioriteit aan. Ook omdat de implementatie van een inlogsysteem meer tijd zou kosten heb ik dit niet opgenomen, omdat je dan ook direct een registratie pagina en een wachtwoord vergeten pagina moet ontwikkelen.

Het email adres heb ik ook niet opgenomen in het systeem. Ik heb dit besproken met de opdrachtgever en hij heeft dit goedgekeurd. De naam van de projectleider zal voldoende zijn voor deze implementatie van de use case.

9.3 Sequence diagrammen voor de use cases

De sequence diagrammen zijn voor de use case “Inzien adviezen” en voor de alternatieve verlopen van de use case “Advies verkrijgen” gemaakt. Ik heb gekozen om voor de alternatieve verlopen aparte sequence diagrammen te maken omdat dit beter inzicht toonde in het verloop de alternatieve verlopen. In figuur 9.1 is het sequence diagram te zien voor het alternatieve verloop “Projectprofiel opslaan” van de use case “Advies verkrijgen”.



Figuur 9.1 Het sequence diagram van het alternatieve verloop "Projectprofiel opslaan"

In dit sequence diagram is te zien hoe een projectprofiel wordt opgeslagen. Het verschil tussen een standaard verloop en een alternatief verloop is niet terug te zien in het sequence diagram. De relatie naar het standaard verloop ook niet. Hierdoor is het sequence diagram wat in de eerste iteratie is ontwikkeld onveranderd gebleven. Het sequence diagram voor “Inzien adviezen” is terug te vinden in het design model in de externe bijlagen.

9.4 Implementatie Class diagram aanpassen

Hier is het implementatie class diagram aangepast met de methodes die uit de sequence diagrammen komen. In Microsoft Visio is dit allemaal automatisch gegaan. Het uiteindelijke implementatie class diagram is terug te vinden in de SRS in de externe bijlage.

9.5 Ontwikkelen van advies inzien en advies verkrijgen (meerdere verlopen)

De use case “Adviezen inzien” en de alternatieve verlopen van de use case “Advies verkrijgen” heb ik het geprogrammeerd zoals beschreven staat in de use case beschrijving en de sequence diagrammen. Om het alternatieve verloop “adviesonderbouwing inzien” te ontwikkelen, was het nodig om de succeselementen per ontwikkelmethode toe te voegen in de SSKB. De succeselementen waren niet direct beschikbaar in de thesis en de opdrachtgever gaf ook meer prioriteit aan het opslaan van het advies. In samenspraak met de opdrachtgever is toen besloten om de adviesonderbouwing niet te ontwikkelen.

9.6 Testen

Voor het testen van de ontwikkelde software binnen deze fase is er uiteindelijk geen tijd meer voor geweest. De bedoeling was om de testtechniek “use case test” uit te voeren over de ontwikkelde use cases. Dit zal na het inleveren van het verslag alsnog worden uitgevoerd.

10 Transition fase

Deze fase zal grotendeels na het inleveren van mijn afstudeerverslag plaatsvinden, omdat dan de overdracht plaats zal vinden. Het afstudeerverslag kan worden gezien als het lesson's learned document. Het onderzoek zal hieronder worden beschreven.

10.1 Onderzoek naar de presentatie van informatie uit de SSKB

Het onderzoek wat in de opdrachtomschrijving is beschreven is uiteindelijk niet uitgevoerd. De ideeën die ik heb opgedaan tijdens dit project waren voldoende om een advies te geven over welke informatie uit de SSKB kan worden getoond. De ingeplande week is hierdoor ook verkeerd geschat. Ik heb dit met de opdrachtgever besproken en heb afgesproken om in het afstudeerverslag een aantal aanbevelingen te doen. Een overzicht van alle aanbevelingen zijn terug te vinden in de externe bijlagen

Hieronder zullen de aanbevelingen worden getoond.

Method	**Omschrijving:** Oftewel software ontwikkel methode. Dit zijn de methodes waar een score voor wordt berekend. Op dit moment zijn er 7 methodes.
	Aanbeveling: *Dit is zeer interessant om op te nemen op de content website. Door een beschrijving erbij te voegen kan hiermee een handig overzicht worden getoond van ontwikkelmethoden.*

Tabel 10.1 voorbeeld van een aanbeveling

Objective	**Omschrijving:** Oftewel doelstelling. Deze doelstellingen verschillen per ingevoerd projectprofiel. Bij een doelstelling hoort altijd 1 of meer ContingencyFactors. Op basis van de gekozen ContingencyFactorOptions zullen met behulp van het algoritme de belangrijkste doelstellingen worden uitgekozen. Op dit moment zijn er 26 Objectives.
	Aanbeveling: *Dit is naar mijn mening zeer handig voor een content website. De Objectives bieden inzicht in wat nou precies doelstellingen zijn voor een project. Door dit inzicht te tonen op een website kunnen gebruikers zien welke doelstellingen er relevant zijn voor projecten. Dit is naar mijn mening interessant om te lezen.*

Tabel 10.2 Voorbeeld van een aanbeveling

11 Evaluatie

Hier zal ik gaan evalueren over het proces wat ik heb doorlopen en de producten die zijn ontwikkeld binnen dat proces. Ik zal in mijn eigen woorden vertellen wat ik er van vond. Wat er fout was gegaan en wat juist goed? Wat had ik beter kunnen doen? Dat zijn vragen die in dit hoofdstuk worden beantwoord.

11.1 Procesevaluatie

Het proces wat ik heb doorlopen om tot de uiteindelijke eindproducten te komen was achteraf gezien goed. Hieronder zal ik er dieper op in gaan.

Voordat het project begon heb ik veel tijd gestoken in het opstellen van het plan van aanpak. Het probleem op school hierbij was namelijk altijd dat zodra het plan van aanpak was gemaakt er niet meer naar werd gekeken. Hierdoor werd het plan van aanpak nutteloos. Het plan van aanpak moet namelijk een weerspiegeling zijn van je project. En dat was op school nooit het geval. Daarom wilde ik een goed plan van aanpak opstellen, ook al duurde dit langer dan verwacht. Bij het opstellen van het plan van aanpak kwam ik ook tot de conclusie dat ik RUP nooit goed heb toegepast binnen de projecten op school. Op school hadden wij namelijk altijd het idee dat je bij een iteratie alle fasen moest doorlopen. Dus als je 3 iteraties had, dan zag het er zo uit: Inception, Elaboration, Construction Transition, Inception, Elaboration, Construction Transition, Inception, Elaboration, Construction Transition. Zoals blijkt uit het voorbeeld is dit onlogisch. Om dus de juiste aanpak te beschrijven moest ik eerst begrijpen hoe RUP in elkaar zat. Hoe de iteraties werken. Doordat ik nu één cyclus doorloop (Inception, Elaboration, Construction Transition) en binnen een fase itereer hoef ik bijvoorbeeld de Inception fase niet meerder keren te doorlopen.

Ook heb ik gebruik gemaakt van iteratieplannen. Deze documenten worden aan het begin van een iteratie gemaakt en bestaan uit: doelstelling voor de iteratie, risico's die worden aangepakt in de iteratie, wijzigingen die moeten worden doorgevoerd in de iteratie, evaluatie van de vorige iteratie en een planning voor de iteratie. Met behulp van deze documenten kon ik mijn iteraties een stuk beter managen. Vooral de planning per iteratie was handig omdat ik hierdoor gedetailleerd kon plannen.

De iteraties waren opgebouwd door middel van de belangrijkste risico's zo snel mogelijk te beheersen. Dit is uiteindelijk een goede manier geweest omdat hierdoor de belangrijkste functionaliteit als eerst ontwikkeld kon worden en ik in ieder geval dat kon opleveren mocht het project mislukken.

Vooraf had ik een aantal iteraties beschreven in het plan van aanpak. Namelijk 2 in de elaboration fase, minimaal 2 construction iteraties en 2 transition iteraties. Dit was achteraf veel te veel omdat een iteratie dan maximaal 2,5 week kan duren. Dit kwam ook omdat ik de opdracht een stuk eenvoudiger

schatte dan het daadwerkelijk was. Door de uitloop van de iteraties en het te makkelijk denken over het afstudeerverslag is het uiteindelijk dus niet verlopen zoals ik me vooraf had ingesteld.

Bij de uitvoering heb ik het proces doorlopen wat ik vooraf had opgesteld. De keuze om als eerst het class diagram af te ronden is achteraf een goede keuze geweest. Dit diagram was zeer complex en daar ben ik ook ongeveer 3 weken mee bezig geweest. Een stuk langer dan vooraf was ingepland. Achteraf had ik dit niet sneller kunnen doen. Alleen de inschatting van de tijd had wellicht wat beter gekund. Het diagram is overigens wel onveranderd gebleven.

Doordat ik uiteindelijk tijd te kort kwam ben ik dus niet meer toe gekomen aan het ontwikkelen van de functionaliteit “advies onderbouwing”. Hieruit blijkt hoe belangrijk een betere schatting van de tijd is. Al had ik vooraf beter geschat was ik er wellicht achter gekomen dat de planning niet reëel was.

11.2 Productevaluatie

Hier zal per eindproduct (en evt. per tussenproduct) een evaluatie worden geschreven. Zijn de uiteindelijke producten geworden wat de opdrachtgever verwachtte en wat had ik beter kunnen doen.

Over het algemeen ben ik tevreden met de eindproducten. De producten geven een goed inzicht in de SOM Selector. Ik ben ook van mening dat de eindproducten een goede basis vormen voor het verdere verloop van het project.

11.2.1 Software Requirements Specification (SRS)

De SRS vormt de eisen die gesteld zijn aan de SOM Selector. Ik ben hier zeer tevreden over. De use cases zijn duidelijk beschreven en dit heeft een goede basis gevormd voor het systeem. Ook door het iteratief ontwikkelen is de kwaliteit van de SRS steeds beter geworden naar mate het project vorderde.

11.2.2 Design model

Het design model is de technische beschrijving van het systeem. Omdat het systeem redelijk complex is was het lastig te beschrijven. Uiteindelijk vind ik wel dat het design model een duidelijk beeld geeft van de technische achtergrond van het systeem. Ik zal het class diagram apart evalueren omdat dat een belangrijk onderdeel is van de SOM Selector.

11.2.3 Class diagram

Het class diagram is de basis voor de SOM Selector. Ik heb hier dan ook veel tijd aan besteed. Voordat het diagram was goedgekeurd moest er veel worden gecommuniceerd met de opdrachtgever. Achteraf ben ik blij dat ik het zo heb gedaan omdat hierdoor een goed class diagram ontstond. Het class diagram is ook onveranderd gebleven wat bij projecten op school nooit voor kwam. Dan merk je ook hoe belangrijk het is als er goed over wordt nagedacht en het veel wordt besproken met de opdrachtgever. In de toekomst zou ik deze aanpak weer gebruiken.

11.2.4 Testrapport

Het testrapport is het resultaat van het testen van de ontwikkelde SOM Selector. Ik heb me bij dit rapport vooral gericht of de SOM Selector overeen kwam met de Excel sheet. Door dit uitvoerig te testen is er een rapport uit gekomen waaruit blijkt dat er geen afwijking zit in de vergelijking van het Excel prototype en de SOM Selector. Achteraf ben ik ook blij dat ik het zo getest heb omdat hiermee kon worden aangetoond de ontwikkelde software eenduidig en correct de juiste functionaliteit aanbied.

11.2.5 SOM Selector

De SOM Selector is het uiteindelijke programma wat ik heb gemaakt. Over het algemeen ben ik hier tevreden over. De functionaliteit die ik beschreven heb is ook geïmplementeerd. Alleen had ik liever gehad dat er meer functionaliteit kon worden ontwikkeld. Het programma doet wat het moet doen maar er zou veel meer mee kunnen. Er was alleen niet voldoende tijd om de rest te implementeren. Ik ben ook wel van mening dat de SOM Selector wel zou kunnen helpen bij het kiezen van een ontwikkelmethode maar dat de echte kracht van het systeem pas duidelijk wordt als het gehele beslismodel van Klaas-Jan is geïmplementeerd.

11.2.6 Adviesrapport

Het opstellen van het advies rapport is heel anders verlopen dan verwacht. Het onderzoek hoefde uiteindelijk niet te worden uitgevoerd omdat ik al genoeg inzicht had in de SSKB. Ik heb daardoor afgesproken met de opdrachtgever dat ik per class heb aangegeven of deze in de toekomst kan worden gebruikt op een website over ontwikkelmethoden. Ik denk dat dit niet uitgebreider had moeten zijn omdat dit vooral bedoeld was om mijn zicht op het systeem vast te leggen.

Literatuurlijst

Boeken

Testen volgens TMap, Martin Pol / Ruud Teunissen / Erik van Veenendaal, 2002

Onderzoeksrapport

Matching project situation and software development methods, Klaas-Jan Molendijk, 2007

Bijlage A: opdrachtomschrijving

Opdrachtsomschrijving

Implementeren van het beslismodel in de SOM Selector bij LogicaCMG

LogicaCMG is een internationale ICT-dienstverlener met wereldwijd circa 40.000 medewerkers. Door haar uitgebreide staat van dienst en branchekennis helpt LogicaCMG haar klanten een leiderschapspositie in te nemen. LogicaCMG levert diensten op tal van terreinen, zoals management- en ICT-consultancy, systeemontwikkeling en integratie en neemt voor klanten complete bedrijfsprocessen in beheer. Haar klanten zijn actief in diverse markten, zoals telecommunicatie, bank- en verzekeringswezen, energie en utilities, industrie, distributie, transport en de overheid.

Het succes van een softwareontwikkelingsproject is in grote mate afhankelijk van de gekozen aanpak. De laatste tijd is de aandacht voor de selectie van software ontwikkel methoden (SOM) meer en meer in de aandacht komen te staan. Dit komt onder andere door de introductie van de Microsoft Solution Framework implementaties (voor ASD en CMMI) en Ivar Jacobson's EssUP, als antwoord op het langer bestaande Rational Unified Process (RUP) van IBM en Agile methodieken als Extreme Programming, Scrum en Crystal. De projectleider heeft een moeilijke keuze te maken, niet alleen omdat de methoden hun eigen sterke en zwakke punten kennen, maar ook omdat elk project zijn eigen karakter heeft. Factoren als grootte, doorlooptijd, tooling, duidelijkheid van de requirements, eindgebruikerbetrokkenheid, betrouwbaarheid van de software en offshore development bepalen de aanpak en daarmee de keuze voor de toe te passen SOM.

In opdracht van het Competence Centre Software Ontwikkel Methoden van LogicaCMG (CCSOM) heeft een student van de Universiteit Leiden de kennis en ervaring omtrent SOMmen beschreven en gekwantificeerd en een beslismodel beschreven waarmee de meest geschikte SOM kan worden bepaald o.b.v. een projectprofiel.
Het afstudeerverslag van de Leidse student is een belangrijk uitgangspunt. Op basis van het afstudeerverslag zal de SOMSelector worden geïmplementeerd.
Een student van de HHS heeft deze opdracht eerder gedaan, maar de uitgangssituatie verschilt met deze opdracht, want het onderzoek van de Leidse student was ten tijde van de opdracht nog gaande. Bij de aanvang van deze opdracht is er wel een afgerond en geaccepteerd afstudeerverslag van de Leidse student. De student van de HHS heeft wel de bruikbaarheid en haalbaarheid aangetoond van SOM Selector.

Het doel van de opdracht is om het beslismodel te implementeren in een decision support tool, de zogenaamde SOM Selector. De SOM Selector zal geïntegreerd worden in een website over Software

Ontwikkel Methoden. De SOM Selector zal gebruik maken van een kennisdatabase. Deze database bevat de informatie over SOMmen zoals beschreven in het afstudeerverslag van de Leidse student. Voor de SOM Selector bestaat een functionele afbakening in de vorm van een MoSCoW-lijst van functionaliteiten. Voor sommige van deze is reeds een functionele specificatie beschikbaar, die eventueel aangepast moet worden. De SOM Selector zal op basis van deze functionele specificaties worden ontwikkeld. Door de grootte van MoSCoW-lijst is met de opdrachtgever afgesproken om alleen de belangrijkste functionele eisen te realiseren. De belangrijkste functionele eisen zullen iteratief worden gerealiseerd. De belangrijkste functionele eisen zijn:

- Het kunnen invoeren van een projectprofiel, door middel van ruim 20 projectkarakteristieken.
- Berekenen advies dat is gebaseerd op het beslismodel en de inhoud van de kennisdatabase. Dit advies zal bestaan uit een gesorteerde lijst van meest geschikte SOMen, waarbij iedere SOM een score heeft tussen 0,0 en 10,0.
- Presenteren van de adviesonderbouwing. Deze onderbouwing bestaat uit de kwantificatie van na te streven doelstelling, eventueel aangevuld met risico's in een projectsituatie en de proceselementen van een SOM.
- Opslaan van het projectprofiel en (onderbouwing van) het bijbehorende advies. Dit zorgt ervoor dat de gebruikers hun advies later kunnen inzien.

De volgende software wordt gebruikt:

- Microsoft Visual Studio 2005
- Microsoft SQL Server
- Microsoft Visio

De volgende rapporten zijn beschikbaar:

- Afstudeerverslag student Leiden
- Productdocumentatie en implementatie student HHS

In het kader van de afstudeeropdracht worden de volgende activiteiten verricht:

- Maken Plan van aanpak
- SOM-keuze voor eigen opdracht
- Herprioriteren functionele specificaties
- Bestuderen afstudeerverslag Leidse student
- Bestuderen productdocumentatie en implementatie student HHS, daarnaast het beoordelen van herbruikbaarheid (van de onderdelen), waarschijnlijk niet herbruikbaar, zoals vernomen van de opdrachtgever
- Domeinmodel (UML-Class diagram) maken op basis van het beslismodel
- Sequencediagram maken voor het algoritme
- ERD diagram maken. (database ontwerp)
- Implementeren belangrijkste functionaliteiten van de SOM Selector
- Testen geïmplementeerd beslismodel. Aantonen dat het algoritme overeen komt met het afstudeerverslag van de Leidse student
- Onderzoeken hoe kennis in de database gepresenteerd kan worden, op een andere manier dan op basis van een projectprofiel. Dit levert een adviesrapport op

De volgende technieken worden gebruikt:

- MoSCoW analyse
- Use case diagram
- Sequentie diagram
- Klasse diagram

- Gestructureerd testen.
- C# 2.0
- ASP .NET

De volgende producten worden opgeleverd:

- Ontwerpdocumentatie SOM Selector en kennisdatabase. Hieronder vallen:
    - Domein model
    - Ontwerp algoritme
    - Testrapport en testcases
    - Aangepast functioneel ontwerp (specificatie) student HHS
- Adviesrapport over het onderzoek hoe de kennis in de database gepresenteerd kan worden.
- Geïmplementeerde en geteste SOM Selector