2010

Process Control in Model Driven Software Development

Student: Anne Buit

Student #: 1504690

Bedrijf: Thinkwise Software

1e Examinator: Bernard Markus

Voorwoord

Ter afronding van de studie Informatica aan de Hogeschool Utrecht ben ik het afgelopen half jaar bezig geweest met een afstudeertraject bij Thinkwise Software.

Tijdens het afstudeertraject heb ik veel ervaring opgedaan, zowel kennis over modelgedreven softwarearchitectuur, softwareontwikkeling en de verschillende aspecten die hierbij komen kijken. Hierbij wil ik alle medewerkers van Thinkwise Software bedanken voor de ondersteuning die ze hebben geboden in het afgelopen half jaar; ik heb nooit voor een dichte deur gestaan.

Hiernaast zou het niet mogelijk zijn geweest om dit afstudeertraject tot een succesvol resultaat te brengen zonder de kennis die ik bij de Hogeschool Utrecht heb opgedaan, zowel bij de opleiding Informatica met het onderzoekssemester rond softwarearchitectuur als bij de minor Charismatisch Leiderschap. Voor deze kennis wil ik alle docenten, collega-studenten, stafdiensten en andere betrokkenen van harte bedanken.

Het afstudeerproject is zonder twijfel het meest intrigerende project geweest waar ik tot op heden aan heb gewerkt. Naast een hoog niveau van abstractie en technische complexiteit dat gemoeid was met de architectuur, het ontwerp en de implementatie, is het onderzoek dat nodig bleek om tot een eenduidige definitie van requirements erg interessant geweest.

De uitdaging van dit project heeft ervoor gezorgd dat ik me in het afgelopen half jaar geen moment verveeld heb. Ik kijk met veel plezier terug op het afgelopen half jaar en ik zie uit naar soortgelijke projecten in de toekomst.

Anne Buit

Managementsamenvatting

De kwaliteit van een product is één van de belangrijkste factoren die het succes van een software ontwikkelingstraject bepalen. De inspanningen voor het repareren van kwaliteitsproblemen lopen op naar mate het ontwikkel- en implementatietraject vordert. Alle betrokkenen zijn er dus bij gebaat om in een vroeg stadium afwijkingen te identificeren.

Voor de Thinkwise Software Factory, de modelgedreven softwareontwikkelingstool van Thinkwise Software, zijn enkele uitbreidingen geïdentificeerd waardoor de Thinkwise Software Factory ondersteuning kan bieden tijdens het ontwikkelingsproces, om de kwaliteit van eindproducten te verbeteren.

ISO 9000, de serie standaarden voor het waarborgen van kwaliteit van het ISO instituut, hanteert de volgende definitie voor kwaliteit: '*The degree to which a set of inherent characteristics fulfills requirements*'[1]. Deze definitie geeft de sterke band tussen de kwaliteit van een product en de requirements van een product aan. Het is dan ook vanzelfsprekend dat van de verschillende potentiële uitbreidingen één onderwerp naar voren gekomen als de meest cruciale uitbreiding, namelijk de integratie van requirements en applicatielogica specificaties.

Requirements zijn de eisen die aan een product worden gesteld door de belanghebbenden. Op basis van de requirements worden producten gebouwd en wordt het eindresultaat gevalideerd. Door de integratie van de requirements in de Thinkwise Software Factory wordt het proces om de requirements van een product te bepalen en te onderhouden ondersteund. Bovendien biedt de Thinkwise Software Factory expliciete traceerbaarheid tussen de requirements en de verschillende modelobjecten van het product, waardoor de requirements worden gerealiseerd.

De integratie van applicatielogica specificaties biedt soortgelijke voordelen. Applicatielogica specificaties beschrijven op een technische, maar platformonafhankelijke wijze onderdelen *business logica* die in het eindproduct geïntegreerd worden. Expliciete traceerbaarheid wordt geboden tussen delen broncode en applicatielogica specificaties, die op hun beurt terug verwijzen naar requirements.

Naast procesondersteuning en traceerbaarheid bieden deze uitbreidingen een vermindering in de spreiding van de documentatie. Hiernaast wordt het projectmanagement ondersteund, doordat de voortgang van een project meetbaar wordt door de vordering van requirement implementatie.

De set requirements mag gebruikt worden als een contract tussen belanghebbenden en het projectmanagement, het projectmanagement kan de set requirements gebruiken voor het inschatten van de kosten voor het realiseren van het product en voor het maken van een projectplanning.

Wijzigingen aan de eisen van een product resulteren in een wijziging van de set requirements. Door de aanpassing van de initiële set requirements is het verantwoordbaar dat het projectmanagement de kosteninschatting en de planning van het project voor deze wijzigingen aanpast. De impact van de wijzigingen aan de eisen van een product wordt op deze manier de verantwoordelijkheid van de belanghebbenden die de wijzigingen voorstellen.

Al deze voordelen zorgen voor een kwalitatief beter eindproduct, doordat meer beheersbaarheid en ondersteuning wordt geboden bij het ontwikkelingstraject. De impact van wijzigingen aan de requirements van een bestaand product kunnen direct worden doorgerekend in de implementatie, door de traceerbaarheid van de requirements die door de Thinkwise Software Factory wordt geboden.

Het resultaat is een stabiel en beheersbaar ontwikkeltraject met een voorspelbaar en gewenst resultaat, met een product gebouwd op basis van de eisen en criteria die voor het product gesteld zijn.

Dat is kwaliteit.

Inhoudsopgave

1 Inleiding

Voor het afronden van de studie Informatica aan de Hogeschool Utrecht is dit project uitgevoerd als afstudeertraject. Het project is begonnen op 1 februari 2010 en zal op 1 juli 2010 eindigen.

Het afstudeerproject is aangeboden door Thinkwise Software, specialist in computergestuurde softwarebouw. Thinkwise Software actief bezig met onderzoek en innovatie rond software ontwikkeling. Het bedrijf is in 2002 opgericht en telt rond de 30 medewerkers.

Qua klanten ligt de focus vooral op bedrijven met een eigen ICT afdeling. Thinkwise Software kan momenteel grote organisaties als Sligro Food Group, Koninklijke Zeelandia, KS Profiel en Royal Huisman Shipyard tot haar klanten rekenen.

Figuur 1 - Thinkwise Software

Naast partnerships met verschillende software- en databaseplatform aanbieders kan Thinkwise Software sinds kort ook AIA Software, de producent van het ITP Document Platform, tot haar partners rekenen.

Thinkwise Software is de producent van de Thinkwise Software Factory. De Thinkwise Software Factory is een softwareontwikkelingstool waarbij de softwarebouw volledig modelgedreven wordt aangepakt. Informatiesystemen bestaande uit gebruikersinterfaces, business functionaliteit, documentatie en een database, worden volledig modelgedreven ontwikkeld. Het afstudeerproject is onderdeel van de doorontwikkeling van de Thinkwise Software Factory naar versie 1.60.

Deze scriptie gaat allereerst in op het bouwproces met de Thinkwise Software Factory en de architectuur van applicaties gebouwd met de Thinkwise Software Factory. Vervolgens wordt het project en de projectaanpak beschreven. Het onderzoek dat vooraf is gegaan aan de implementatie en een omschrijving van deze implementatie worden gevolgd door een terugblik op het project en een afsluitend advies over verschillende aspecten van de Thinkwise Software Factory en Thinkwise Software.

2 De Thinkwise Software Factory

Om het project in context te kunnen plaatsen, is het belangrijk om bekend te zijn met de Thinkwise Software Factory. Dit hoofdstuk zal de verschillende aspecten van deze modelgedreven software ontwikkelingstool belichten. Aanpassingen die tijdens het project zijn gedaan aan de Thinkwise Software Factory worden in dit hoofdstuk niet meegenomen, de situatie wordt besproken zoals deze is vóór de uitvoering van het project.

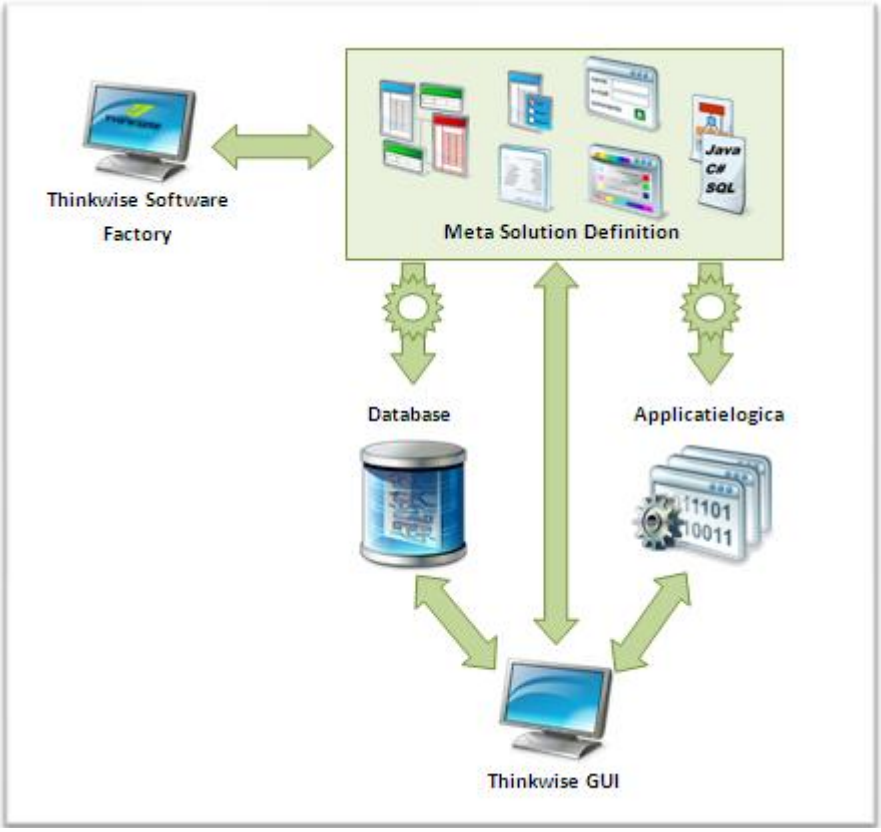
De Thinkwise Software Factory is een modelgedreven software ontwikkelingstool, gericht op het computergestuurd ontwikkelen van zakelijke, administratieve applicaties.

Alle functionele aspecten van een product worden door middel van modellen gespecificeerd in de Meta Solution Definition (MSD), een elektronische bouwtekening. Denk hierbij aan het datamodel, processtromen, datamanipulatie schermen, *Business Intelligence* schermen, stijl, processtromen menustructuren etc.

Applicatielogica, de specifieke bedrijfsregels, wordt als broncode op een aspectgeoriënteerde manier in het model geweven. Dit zorgt voor een enorme reductie in broncode. Een eenheid applicatielogica met enkele tientallen regels broncode kan door één weefconditie op honderden plekken in het model worden toegepast.

De database en applicatielogica waar het eindproduct gebruik van zal maken worden door de Thinkwise Software Factory gegenereerd. Anders dan conventionele modelgedreven software ontwikkelingstools wordt de grafische gebruikersinterface niet gegenereerd aan de hand van de elektronische bouwtekening. In plaats daarvan worden grafische gebruikersinterfaces geboden die de MSD *runtime* interpreteren en zich dynamisch vormen naar dit model.

Dit hoofdstuk gaat niet in op de ingebruikname van eindproducten, de autorisatie van eindproducten en verschillende andere aspecten die voor de ingebruikname van eindproducten naar voren komen. Dit wordt buiten de Thinkwise Software Factory ingericht door middel van de IAM, de Intelligent Application Manager.



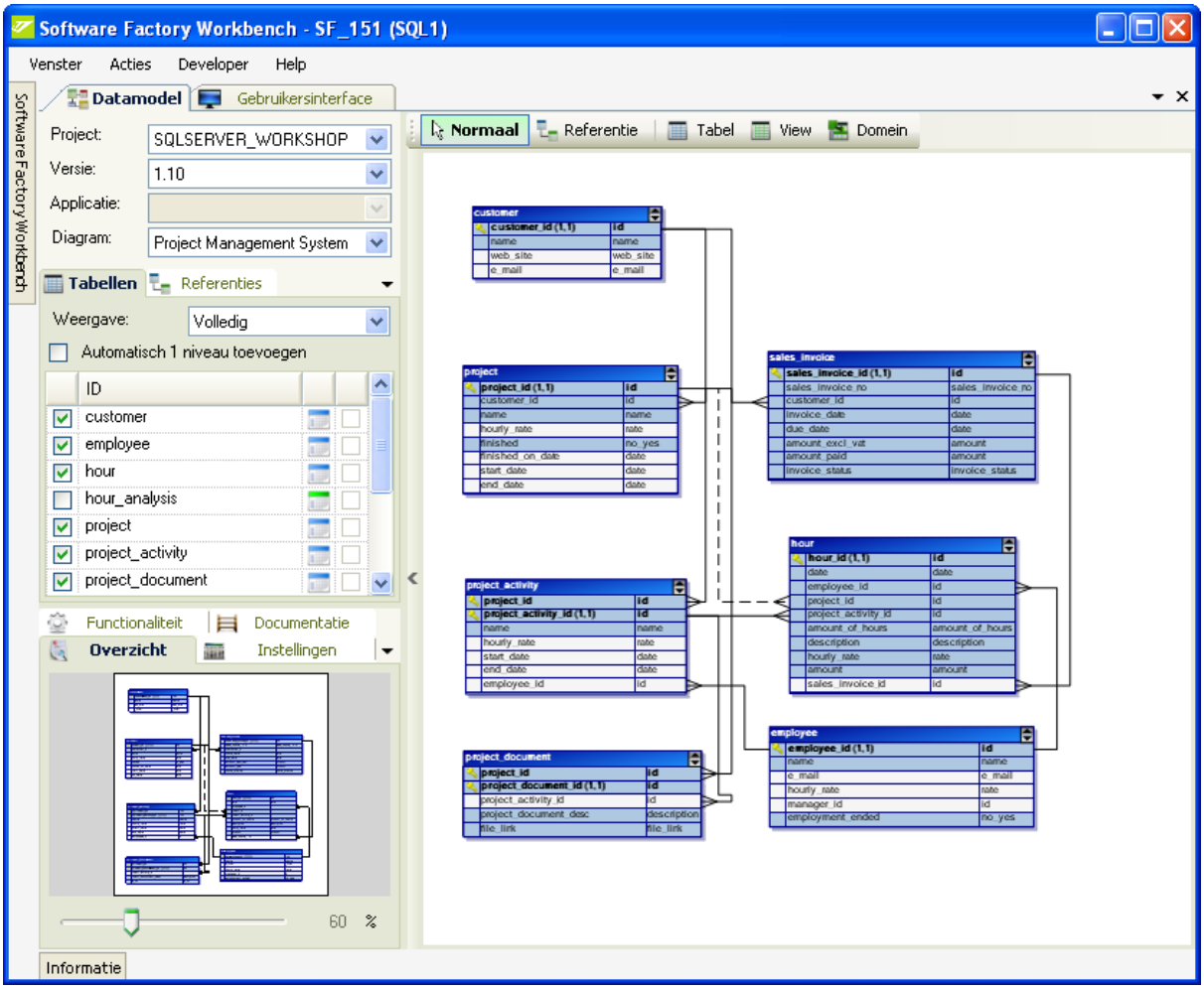
Figuur 2 - Runtime MSD interpretatie door de Thinkwise GUI

2.1 De modelleerschermen

De Thinkwise Software Factory biedt verschillende modelleerschermen om het model vast te leggen van een product. Om een volledig beeld te vormen van de functionaliteit van de Thinkwise Software Factory worden de verschillende modelleerschermen in deze paragraaf toegelicht.

2.1.1 Datamodel

Allereerst biedt de Thinkwise Software Factory een modelleerscherm om het datamodel in te richten. In dit scherm kunnen de verschillende data entiteiten, attributen en referenties worden gemodelleerd. Dit is de eerste stap in het modelleertraject, op basis van de data entiteiten zal het modelleertraject worden voortgezet[1].



Figuur 3 – Het modelleerscherm waarin het datamodel wordt vastgelegd.

[1] Een ontwikkeltraject is bezig om ook op basis van workflow diagrammen het modelleertraject te starten.

2.1.2 GUI model

Het gebruikersinterface modelleerscherm biedt de mogelijkheid om de weergave van data entiteiten en attributen in het eindproduct te modelleren.

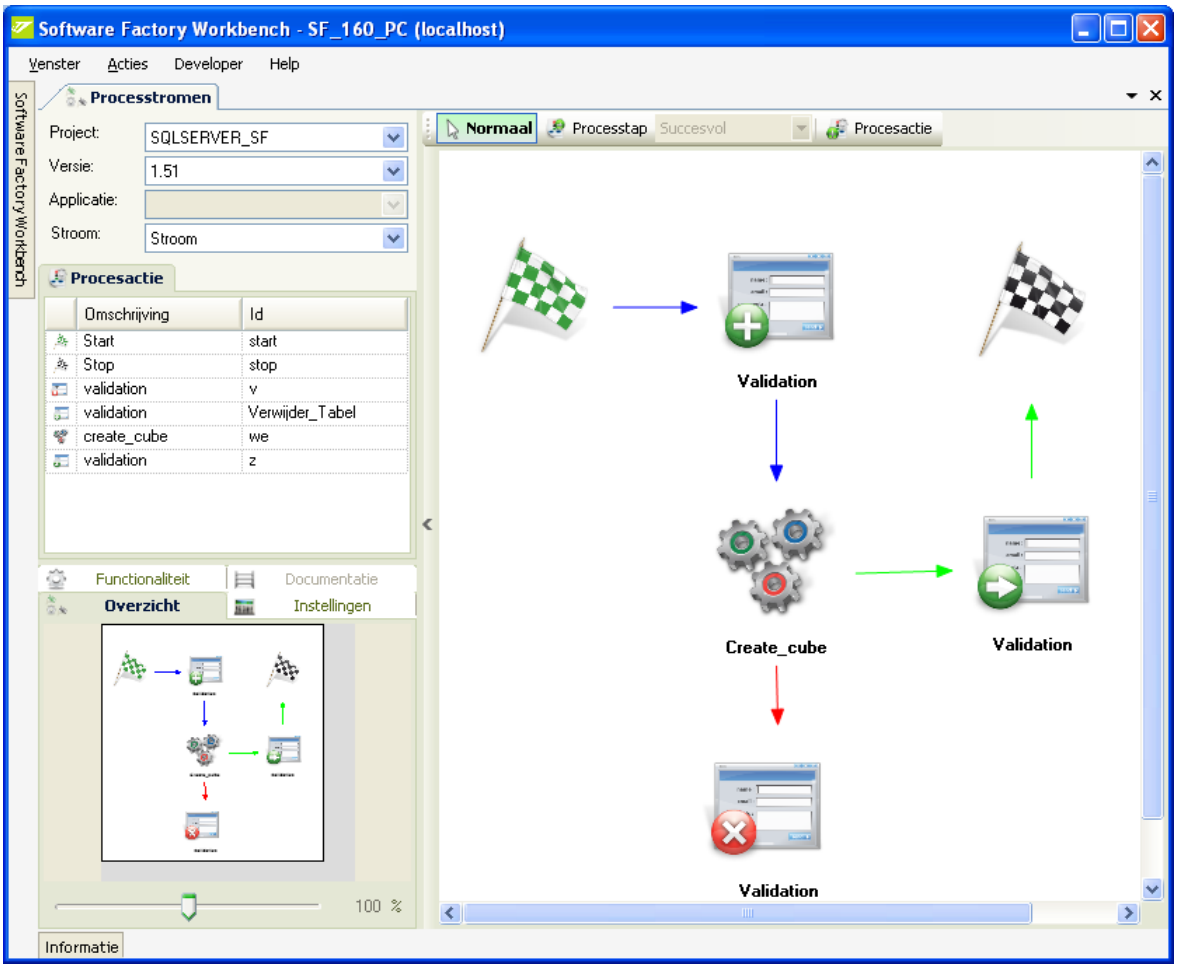
Hierbij kan gebruik worden gemaakt van de componenten die de Thinkwise GUI's bieden om data entiteiten, attributen en referenties weer te geven, zoals een lijstcomponent, een formuliercomponent en een tabbladencomponent. Verschillende opties worden gemodelleerd zoals veldweergave opties, stijlattributen, tabblad weergave, standaard sorteringen en filters enzovoorts.

Het ontwerpen van menu's, rapportages, vertalingen en business intelligence schermen en andere onderwerpen wordt ook door dit modelleerscherm gefaciliteerd.

In het GUI model worden weergaveopties enkel statisch gedefinieerd. Dynamische of conditionele weergaveopties worden door middel van applicatielogica aangestuurd, gedefinieerd in de Functionality Weaver.

2.1.3 Processtromen

Door het modelleren van processtromen kan de gebruiker worden begeleid bij het uitvoeren van opeenvolgende acties in het eindproduct. Hierbij kan de ontwerper gebruik maken van verschillende standaard DML acties en bepaalde GUI acties.



Figuur 4 - Het processtromen modelleerscherm

2.1.4 Functionality Weaver

Specifieke regels met betrekking tot dynamische weergaveopties, berekeningen en datamanipulatie kunnen op een aspectgeoriënteerde manier door het model worden geweven. De Functionality Weaver biedt de mogelijkheid om deze regels, binnen Thinkwise Software beter bekend als *applicatielogica*, als broncode templates vast te leggen en in het model te weven.

Dit gaat verder dan aspectgeoriënteerd programmeren (AOP), aangezien de developer de mogelijkheid heeft om een dynamische toewijzing voor een broncode template te definiëren.

2.1.5 Documentatie

Het documentatie scherm biedt de mogelijkheid om helpteksten te definiëren voor de gemodelleerde objecten. De ontwikkelaar heeft de mogelijkheid om deze helpteksten tot in het diepste detail van eindproduct objecten vast te leggen, bijvoorbeeld per element van een keuzelijst.

Deze helpteksten zijn beschikbaar in de eindproducten. De Thinkwise GUI's zijn gebouwd om deze helpteksten te laten reageren op het gebruik van de GUI. Wanneer een gebruiker het helpscherm activeert wanneer de cursor zich in een veld bevind, zal het helpvenster openen met de helptekst van het veld.

2.1.6 Generator

Door middel van de generator worden scripts gegenereerd om de database van het eindproduct aan te maken of te upgraden. Deze scripts worden door de Thinkwise Software Factory gegenereerd, afhankelijk van het geselecteerde database platform.

Het GUI model en processtromen worden niet omgezet in scripts of andere gegenereerde objecten. Dit model wordt runtime door de Thinkwise GUI's ingeladen. Hierover meer in het volgende hoofdstuk.

De generator genereert wel scripts voor de logica laag. Hierbij wordt de geweven broncode geplaatst in een gegenereerd platform. Het is mogelijk om een database als platform te gebruiken voor de applicatielogica. Hiernaast kan ook gekozen worden voor een middle-tier platform van een bepaalde taal, zodat het product kan aansluiten bij externe interfaces, zoals een bestaande service bus.

2.2 Thinkwise GUI's

Om met een eindproduct te werken worden Thinkwise GUI's ingezet. De Thinkwise GUI's interpreteren de Meta Solution Definition en vormen zich runtime volgens dit model. Schermen bevatten componenten die zich runtime initialiseren op de definitie van een object.

Ter illustratie, hetzelfde generieke formuliercomponent wordt geïnitialiseerd op de definitie van het dataobject *Klant* en *Orderregel*. Het formuliercomponent interpreteert de MSD en toont de velden op de manier zoals dit is gedefinieerd in de MSD, het component gebruikt de definitie van de volgorde, veldweergave, verplichtheid, etc.

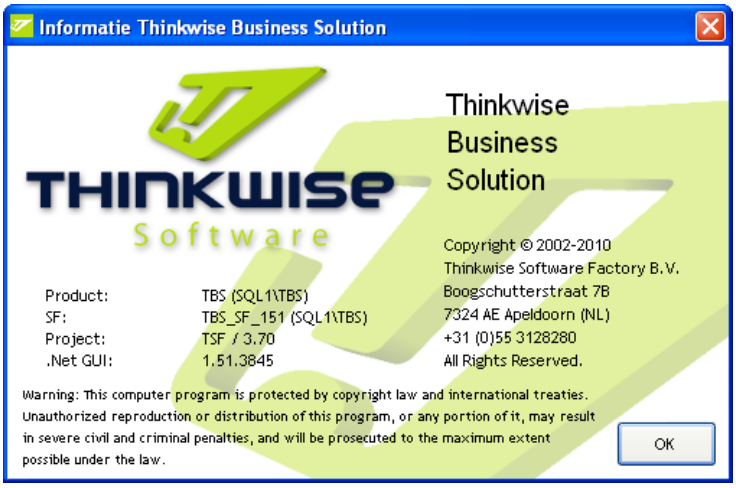
Runtime interpretatie geeft de garantie dat alle formulieren uniform werken. Niet elk scherm hoeft opnieuw getest te worden, de generieke aard van de componenten biedt de garantie dat elk component dezelfde werking vertoond in elk scherm.

Thinkwise GUI's zijn beschikbaar in enkele platformen, namelijk Java SE, .NET, Java EE (web). Nieuwe platformen komen ook beschikbaar, er wordt gewerkt aan een ASP .NET (web) GUI.

Thinkwise Software onderhoudt deze GUI's. Wanneer nieuwe opties aan het MSD worden toegevoegd, worden de GUI's aangepast zodat ze deze opties kunnen interpreteren.

Door de runtime modelinterpretatie architectuur is het voor bedrijven makkelijker om over te gaan op een nieuw GUI platform wanneer het huidige platform verouderd is. Ook zijn er geen aanpassingen nodig wanneer het GUI platform verandert. Een webgebaseerde GUI interpreteert exact hetzelfde model als een stand-alone GUI en maakt gebruik van dezelfde applicatielogica laag.

Hiernaast biedt deze architectuur het voordeel dat wijzigingen in de Meta Solution Definition direct doorgevoerd in de grafische gebruikersinterface. Ook kan in een vroeg stadium, na het opstellen van het datamodel, direct een grafische gebruikersinterface worden getoond als prototype, waarbij de standaardwaarden voor grafische weergaveopties worden gehanteerd.



Figuur 5 - Het informatiescherm van een Thinkwise GUI.
Product: Eindproduct database naam en server
SF: Model database naam en server
Project: Naam / versie van het MSD model

3 Projectbeschrijving

3.1 Probleemstelling

Het bouwproces met de Thinkwise Software Factory wordt op het gebied van kwaliteit ondersteund door middel van model validaties en invoercontroles. Dit garandeert echter enkel de integriteit van het model.

De Thinkwise Software Factory ondersteunt het ontwikkelproces onvoldoende op het gebied van functionaliteit. Traceerbaarheid met de doelen van de belanghebbenden wordt niet door de Thinkwise Software Factory ondersteund, met als gevolg dat wijzigingen van de eisen met modelwijzigingen tot gevolg een onbekende impact hebben op andere eisen die geraakt worden door deze modelwijziging.

Het software ontwikkelingsproces is productgericht en biedt vooral ondersteuning in het gebied van modelleren, niet op andere onderdelen zoals de initiële analyse van de wensen en het testen van de eindproducten.

Dit heeft direct invloed op de kwaliteit van eindproducten en dus op de waardering van de Thinkwise Software Factory als software ontwikkelingstool.

3.2 Opdrachtomschrijving

De hoofdvraag luidt als volgt:

'*Op welke manieren kan het bouwproces worden ondersteund om de kwaliteit van eindproducten van de Thinkwise Software Factory te verbeteren?*'

Deze hoofdvraag is onderverdeeld in enkele onderdelen. Één van deze onderdelen is naar voren gekomen als een uitbreiding van strategisch belang voor Thinkwise Software.

Om de onderdelen in context te plaatsen, is het belangrijk om een goede definitie van kwaliteit vast te stellen. ISO 9000, de serie standaarden voor het waarborgen van kwaliteit van het ISO instituut, definieert kwaliteit als volgt: '*The degree to which a set of inherent characteristics fulfills requirements*' [(1)].

Na inventarisatie zijn de volgende vijf onderdelen naar voren gekomen waar de Thinkwise Software Factory het ontwikkelproces kan verbeteren.

- Requirements en applicatielogica specificaties
- Broncode versiebeheer en meta-informatie
- Broncode review- en controlemogelijkheden
- Impactanalyses
- Geweven broncode testmogelijkheden

Hoewel de laatste vier uitbreidingen in eerste instantie sterker naar voren kwamen in het project mandaat, zijn deze uitbreidingen buiten de scope van het project komen te liggen, door het strategisch belang van het integreren van de requirements en applicatielogica.

Paragraaf 4.3, waar de afwijking van de initiële project fasen verdeling wordt besproken, gaat dieper in op deze scopewijziging.

3.2.1 Requirements & Applicatielogica specificaties

De eisen die aan producten worden gesteld, de requirements, staan op het moment los van de Thinkwise Software Factory. Hiernaast worden de applicatielogica specificaties, de tekstuele regels die in de Functionality Weaver worden uitgewerkt tot broncode templates en broncode template toewijzingen, beschreven in documenten buiten de Thinkwise Software Factory.

Door deze te integreren in de Thinkwise Software Factory worden de volgende voordelen behaald:

- Reductie van de spreiding van documentatie.
- Mogelijkheid tot ondersteuning bij de specificatie van requirements en applicatielogica.
- Mogelijkheid tot ondersteuning bij het gebruik van de gespecificeerde requirements en applicatielogica.
- Traceerbaarheid tussen requirements en model, applicatielogica specificaties en broncode.
- Analysemogelijkheden over het implementatieproces van de requirements

3.2.2 Broncode versiebeheer en meta-informatie

Referenties naar broncode bestanden worden opgenomen in het model. Het doel is om broncode op te nemen in het MSD model, in plaats van de referenties naar losse bestanden. Dit biedt mogelijkheden met betrekking tot analyse en het aanbieden van meta-informatie over de broncode, bijvoorbeeld versiehistorie.

Hiernaast kan de Thinkwise Software Factory het ontwikkelproces ondersteunen door meta-informatie op te slaan over de *lifecycle* van een deel broncode, en hier inzicht in te bieden. Bijvoorbeeld, veel wijzigingen in de code of een grote toename in toewijzingen door model-wijzigingen geeft aan dat een deel broncode meer kritiek is en wellicht meer controle nodig heeft.

Deze aanpassingen bieden meer controle over de broncode, wat resulteert in hogere kwaliteit van de broncode.

3.2.3 Broncode review- en controlemogelijkheden

De Thinkwise Software Factory kan de kwaliteit van de broncode templates verbeteren door review mogelijkheden en controlemogelijkheden te bieden voor deze. Hierdoor komt de verantwoordelijkheid voor de kwaliteit van de broncode bij meerdere personen te liggen. Op deze manier is de kans dat fouten worden ontdekt vóór de ingebruikname van het product groter.

3.2.1 Impactanalyses

De Thinkwise Software Factory zal de mogelijkheid kunnen bieden om impactanalyses op bepaalde modificaties van het model weer te geven op andere delen van het model. Op deze manier wordt de kennisdrempel voor nieuwe ontwikkelaars met de Thinkwise Software Factory verlaagd en komen ontwikkelaars minder snel voor verrassingen te staan.

Deze uitbreiding bestaat uit twee delen, het bieden van impactanalyses op de gemodelleerde onderdelen van de MSD en het bieden van impactanalyses met betrekking tot de broncode en toewijzingen van broncode. Voor het tweede deel is het van belang dat de expliciete integriteit tussen broncode en het model wordt verbeterd.

3.2.1 Geweven broncode testmogelijkheden

Het gestructureerd (en mogelijk geautomatiseerd) testen van de geweven broncode wordt niet ondersteund door de Thinkwise Software Factory. De broncode is echter het onderdeel van de Thinkwise Software Factory waar de functionaliteit een beduidend grotere rol speelt dan de integriteit.

Hiernaast zal deze uitbreiding aan het projectmanagement de mogelijkheid bieden om een afweging tussen de kwaliteit van het eindproduct en de kosten die het testen van de applicatielogica met zich mee brengt.

4 Projectaanpak

Om het project beheersbaar te houden is gebruik gemaakt van de PRINCE2 projectmanagement methode. Binnen Thinkwise Software wordt voor bepaalde projecten al gebruik gemaakt van deze methode en verschillende templates voor documenten waren daarom reeds beschikbaar.

4.1 Projectmanagement methode

PRINCE2 is een productgeoriënteerde en flexibele methode. Aspecten van de methode die niet van toepassing zijn kunnen worden genegeerd. Een project wordt opgedeeld in fasen, waarbij de deliverables, de producten van een fase, de opbrengst van een fase vertegenwoordigen.

Belangrijk bij PRINCE2 is de *Business Case*, de redenatie van de opdrachtgever om het project uit te voeren. Per fase zal de opbrengst van de fase, de deliverables, aan de business case worden getoetst.

De aansturing van het project wordt uitgevoerd door de stuurgroep. Deze groep wordt door middel van *Management by Exception* op de hoogte gehouden van de status van het project. Tijdens het project worden op voorgedefinieerde momenten rapporten opgeleverd aan de stuurgroep. Wanneer kleine scope veranderingen nodig zijn wordt dit overlegd met de *Change Authority*, ook onderdeel van de stuurgroep. Grote scope veranderingen of uitloop in de planning wordt aan de stuurgroep voorgelegd in de vorm van een *Exception Report*, waarbij de stuurgroep een beslissing neemt over het vervolg van een project.

De stuurgroep is verdeeld in vier rollen:

De **Business Executive** is onderdeel van de stuurgroep. Deze rol is vervuld door Victor Klaren (CTO). Als eigenaar van de business case is Victor Klaren de eindverantwoordelijke voor het project en is bevoegd alle beslissingen te nemen.

De **Senior User** is als onderdeel van de stuurgroep verantwoordelijk voor het representeren van de gebruikers die met de resultaten van het project zullen werken. De Senior User zal de rol van koppeling tussen de projectontwikkeling en eindgebruikers realiseren en kan worden benaderd met vragen over de functionele specificaties. Deze rol is vervuld door Jasper Kloost (Senior Ontwikkelaar).

De **Senior Supplier** is ook onderdeel van de stuurgroep en verantwoordelijk voor het vrijgeven van de resources benodigd om het project uit te voeren. Hiermee is hij verantwoordelijk voor de allocatie van de projectmedewerkers aan het project, de beschikbaarheid van een projectomgeving en eventuele benodigde hard- en software. Aangezien er voor dit project geen medewerkers beschikbaar zullen worden gesteld, zal de rol vooral het beschikbaar stellen van benodigde hard- en software zijn. Frank Wijnhout (Senior Ontwikkelaar) heeft deze rol vervuld.

De **Change Authority** is per definitie niet onderdeel van de stuurgroep. Echter, de Change Authority heeft de mogelijkheid om kleine scope wijzigingen toe te staan, zonder een Exception Report op te leveren bij de stuurgroep. Hiervoor zal een Change Request moeten worden ingediend bij de Change Authority. Deze rol is vervuld door Frank Wijnhout.

4.2 Fasen verdeling

De initiële fasen verdeling is tijdens de initiatie fase van het project als volgt opgesteld:

Start fase
Het project is begonnen met de *Start Fase*. In deze fase is het Project Management Team aangesteld. Vervolgens is aan de hand van het Project Mandaat een Project Voorstel opgesteld. Ook is in deze fase een planning opgesteld voor de Initiatie Fase.

Initiatie fase
Het Project Initiatie Document is het belangrijkste product van deze fase. Op basis hiervan kunnen de vervolgfasen van het project worden uitgevoerd. In deze fase zal ook een inventarisatie worden gedaan van de verschillende potentiële procescontrole uitbreidingen.

Ontwikkelfase 1 – Requirements en applicatielogica specificaties
De deliverables van de eerste ontwikkelfase dienen antwoord te geven op de deelvraag *'Hoe kan de brug tussen functionele specificaties en de Thinkwise Software Factory worden geslagen?'*.

Ontwikkelfase 2 – Broncode versiebeheer en meta-informatie
De deliverables van de ontwikkelfase voor broncode uitbreidingen dienen antwoord te geven op de deelvraag *'Op welke manier kan versiebeheer en meta-informatie behoud worden toegepast op broncode binnen de Thinkwise Software Factory?'*.

Ontwikkelfase 3 – Geweven Applicatielogica Uitbreidingen
De deliverables van deze fase dienen antwoord te geven op de deelvraag *'Welke mogelijkheden kan de Thinkwise Software Factory bieden met betrekking tot controle en review van geweven applicatielogica?'*.

Ontwikkelfase 4 – Impactanalyses
De deliverables van de vierde ontwikkelfase dienen antwoord te geven op de deelvraag *'Hoe kan de Thinkwise Software Factory de ontwikkelaar op de hoogte worden gesteld van de impact van modelwijzigingen?'*.

Ontwikkelfase 5 – Geweven broncode testmogelijkheden
De deliverables van de vijfde ontwikkelfase moeten antwoord te geven op de deelvraag *'Hoe kan het gestructureerd testen van applicatielogica worden gerealiseerd?'*. Deze fase kan naast programmatuur ook documentatie in de vorm van een handleiding opleveren om de ontwikkelaar instructies te geven met betrekking tot het testen.

Scriptie fase
De Scriptie fase is toegewijd aan het schrijven van de scriptie. Deze fase zal overlappen met de derde en vierde ontwikkelingsfasen van het project. Dit is echter niet het enige moment waarop aan de scriptie wordt gewerkt, tijdens alle fasen door kan hier tijd aan worden besteed.

Transitie Fase
Deze fase zal overlappen met de vijfde ontwikkelingsfase van het project. Deze fase heeft geen deliverables of managementproducten, maar bestaat uit kennisoverdracht en afronding.

4.3 Afwijkingen

Tijdens het onderzoek van de eerste ontwikkelfase, requirements en applicatielogica, zijn de verschillende ontwikkelfasen opnieuw aan de business case getoetst. Hierbij kwam naar voren dat het integreren van de requirements vele malen belangrijker was, zelfs van strategisch belang voor Thinkwise Software, dan de overige deliverables.

De scope van de eerste ontwikkelfase, die eerst enkel applicatielogica behelsde, is daardoor flink uitgebreid. Het gevolgd hiervan was een Exception Report, opgesteld op 30 maart, om de ontwikkelfasen vier en vijf, *Impactanalyses* en *Geweven Broncode Testmogelijkheden*, buiten de scope van het project te plaatsen. De stuurgroep is hiermee akkoord gegaan. Dit Exception Report is opgenomen als bijlage (Bijlage 1).

Begin mei is een tweede Exception Report opgesteld. De reden hiervoor was de beslissing dat in plaats van prototypen een volledig werkend product met betrekking tot *requirements* en *applicatielogica* medio juli opgeleverd moet zijn, als één van de speerpunten van de nieuwe release van de Thinkwise Software Factory. Ook de integratie met de huidige modelleerschermen moet voor deze release afgerond zijn. Hierbij zijn de andere deliverables, met uitzondering van de scriptie, door de stuurgroep buiten de scope van het project geplaatst.

5 Onderzoek

Voordat requirements en applicatielogica specificaties geïntegreerd kunnen worden in de Thinkwise Software Factory, is het van belang om een heldere context te scheppen rond deze concepten. Voordat de ontwerpfase is gestart is uitgebreid onderzoek gedaan naar requirements en applicatielogica. Dit onderzoek heeft de volgende vragen beantwoord:

- Wat zijn requirements en aan welke eisen moeten requirement specificaties voldoen?
- Wat is applicatielogica en hoe wordt applicatielogica gespecificeerd?
- Waar bevinden de requirements en de applicatielogica specificaties zich in de architectuur van Thinkwise Software Factory producten?
- Welke actoren zijn betrokken bij het opstellen en het gebruik van requirement specificaties en applicatielogica specificaties?
- Hoe ziet het software ontwikkelingstraject er uit bij het gebruik van requirement specificaties en applicatielogica specificaties?

In dit hoofdstuk staan de onderzoeksresultaten beschreven. Deze resultaten zijn de basis waarop de uitbreidingen zijn ontworpen en gerealiseerd.

5.1 Requirements definitie

Requirements zijn de basis van softwareontwikkelingstrajecten, de requirements beschrijven de eisen die door de belanghebbenden aan het product worden gesteld. Door middel van requirements kan het doel, het gebruik en de functionaliteit van een software systeem worden beschreven.

IEEE 830 stelt dat door het opstellen van requirement specificaties de volgende doelen worden bereikt [(2)].

- De specificaties dienen als basis waarop een overeenkomst kan worden bereikt tussen de klant en de software leverancier over wat het product moet doen.
- Door het opstellen van de specificaties wordt de moeite die het ontwikkelen van de software met zich meebrengt verminderd, doordat in een vroeg stadium eventuele onduidelijkheden worden geïdentificeerd.
- Op basis van de specificaties kan een inschatting worden gemaakt over de omvang van het ontwikkeltraject.
- Hiernaast bieden de specificaties een ijkpunt voor validatie en verificatie van het eindproduct.
- Door de platformonafhankelijke aard van requirement specificaties kunnen deze worden hergebruikt in andere omgevingen.
- Hiernaast dienen de specificaties als basis voor aanpassingen aan het systeem.

Requirements worden in de theorie opgesplitst in drie niveaus [(3)], business-, gebruikers- en systeemrequirements. Daarnaast worden requirements onderverdeeld in twee soorten requirements, functionele en niet-functionele requirements.

Businessrequirements
De businessrequirements beschrijven de *waarom* vraag. Deze requirements vormen de basis van visie- en scopedocumenten voor het project, bijvoorbeeld het projectplan. Hiernaast kunnen veel gebruikersrequirements worden teruggeleid naar businessrequirements. *De bestellinggegevens zullen via internet beschikbaar zijn om de klanten op de hoogte te houden van de status van hun bestelling. Op deze manier zal de klanttevredenheid te verbeteren.*

Gebruikersrequirements
De gebruikersrequirements representeren de *wat* vraag. De term *gebruiker* staat hier voor elke mogelijke stakeholder, elke mogelijke belanghebbende. Gebruikersrequirements staan los van de functies van het systeem, ze beschrijven enkel wat de actoren (met het systeem) doen.
De medewerkers van de verkoop afdeling stellen bestellingen samen aan de hand van telefonisch contact met een klant.

Systeemrequirements
De systeemrequirements representeren de *hoe* vraag. *Hoe* gaat het systeem de gebruiker ondersteunen bij het uitvoeren van een bepaalde taak. Deze requirements zijn implementatie-onafhankelijk, termen zoals tabellen, schermen en knoppen horen hier niet thuis. Systeem-requirements beschrijven dus niet hoe het systeem de ondersteuning zal realiseren.
Het systeem controleert na het invoeren van een bestelling de kredietwaardigheid van de klant waar de bestelling voor wordt ingevoerd. Wanneer een klant niet kredietwaardig is wordt de bestelling door het systeem gemarkeerd als afgekeurd.

Functionele- en niet-functionele requirements
Functionele requirements beschrijven de functionaliteit die het systeem aanbiedt voor de belanghebbenden [(3)]. Denk hierbij aan diensten, taken en functies die het systeem zal uitvoeren [(4)].

Functionele specificaties kunnen gerealiseerd worden in de MSD door middel van modelobjecten en applicatielogica specificaties.

Niet-functionele requirements beschrijven eigenschappen van het systeem zoals de prestaties, veiligheid en onderhoudbaarheid [(4)]. Hierbij kan gerefereerd worden aan de kwaliteitsaspecten van producten gedefinieerd door ISO 9126, zoals *Reliability, Usability* en *Efficiency* [(5)].

5.2 Requirements specificatie

Door de kritieke positionering van requirements in het software ontwikkelingsproces is het belangrijk dat requirements aan een aantal regels voldoen. Arendsen stelt dat een correcte requirement aan de volgende regels voldoet [(3)]:

Uniek identificeerbaar
Een requirement krijgt een uniek nummer of referentiecode. Deze code mag vervolgens niet gewijzigd worden. De code wordt gebruikt om naar dit requirement te kunnen verwijzen.

Atomair
Een requirement mag niet meer dan één eis of mogelijkheid bevatten. Als de requirement nog is op te splitsen moet dat gebeuren en ontstaan er dus twee requirements. IEEE stelt dat er wel meerdere condities of beperkingen in een requirement worden gespecificeerd [(6)].

Eenduidig
Een requirement mag maar op één manier geïnterpreteerd kunnen worden. Dit is een regel waarvan de meetbaarheid soms moeilijk is vast te stellen. De volgende vuistregels kunnen in acht worden genomen:

- Gebruik geen formuleringen die zaken open laten;
- Gebruik geen termen die meer dan één betekenis hebben;
- Beschrijf altijd wie een actie uitvoert

Hiernaast is het aangeraden om ontkenningen te vermijden. '*Het systeem zal niet adresgegevens opslaan van een klant*' biedt oneindig veel mogelijkheden voor wat wél zal worden opgeslagen in het systeem.

Geen implementatiedetails
Requirements beschrijven het probleem of de behoefte van de belanghebbende en daarmee de buitenkant van het systeem (wat gaat erin en wat komt eruit?). Requirements mogen dan ook geen oplossingen beschrijven.

Traceerbaar
Bij bedrijven met een dynamische business komt het vaak voor dat de requirements veranderen in de loop van de tijd. Door aanpassingen in wetgeving, dynamiek van de markt en wensen van gebruikers komen er requirements bij, veranderen requirements en worden andere requirements verouderd [(2)]. Om deze reden is het belangrijk dat de requirements een hoge mate van *traceability*, ofwel traceerbaarheid, hebben.

Op elk moment moet achterhaald kunnen worden door wie, waarom en wanneer een requirement is opgesteld en veranderd. Dit is de *backwards traceability* van een requirement. Een hiërarchische indeling van requirements helpt ook bij de *backwards traceability*. Systeemrequirements worden gekoppeld aan de gebruikersrequirements die ze ondersteunen. Gebruikersrequirements worden op hun beurt gekoppeld aan de businessrequirements die het doel van de requirement beschrijven.

Wanneer een requirement verandert of veroudert, moet achterhaald kunnen worden op welke manier de requirement is geïmplementeerd in het huidige systeem. Op basis daarvan kan de impact van een verandering worden bepaald en kan een plan worden opgesteld voor de aanpassing van het systeem. Dit wordt de *forward traceability* genoemd.

Testbaar / verifieerbaar
De uiteindelijke test van het geïmplementeerde systeem vindt plaats op basis van de requirements. Als is vastgesteld dat aan alle requirements is voldaan, wordt het systeem als afgerond beschouwd. Dit betekent dat het mogelijk moet zijn om voor elke requirement vast te stellen of deze is vervuld, dus meetbaar en begrensd.

Vooral bij niet-functionele requirements is het belangrijk dat de acceptatiecriteria wordt opgenomen, aangezien deze requirements niet een *wat* vraag beantwoorden, maar gericht zijn op prestaties, performance en gebruikersgemak. Als voorbeeld, de niet-functionele requirement '*De nieuwe applicatie moet snel zijn met het uitvoeren van de MRP-run'* bevat geen acceptatiecriteria. '*De nieuwe applicatie moet de MRP-run uitvoeren binnen 10 minuten'* bevat deze wel.

Prioriteit bepaald
Voor elk requirement moet vastgelegd zijn wat de prioriteit is. Hiervoor kan bijvoorbeeld de MoSCoW prioritering van gebruikt worden. Deze is gericht op *incrementen* en is daardoor uitermate geschikt voor het werken met projectversies zoals dit binnen de Thinkwise Software Factory wordt gedaan. De MoSCoW prioritering bestaat uit de volgende aanduidingen[2]:

- Must have – De requirement is van essentieel belang. De belanghebbenden zullen niet tevreden zijn wanneer deze requirement niet is geïmplementeerd.
- Should have – De requirement is van belang, maar er is een tijdelijke oplossing mogelijk waardoor het acceptabel is als deze requirement op een later moment wordt geïmplementeerd.
- Could have – De requirement is leuk om te hebben en kan gezien het tijdsbestek binnen de scope gerealiseerd worden, maar zal buiten de scope vallen wanneer het tijdsbestek is onderschat.
- Won't have – Deze prioriteit staat voor 'we willen deze requirement wel implementeren, maar nu niet'. Dit zijn wensen die in eerste instantie niet worden geïmplementeerd.

[2] Deze beschrijving van de MoSCoW prioritering is een vrije vertaling van een beschrijving opgesteld door het DSDM Consortium[(7)].

5.3 Applicatielogica definitie

De Thinkwise Software Factory heeft een serie concepten geïntroduceerd waarmee regels die op het model van toepassing zijn als broncode in het model te weven. Deze regels variëren van berekeningen, controles, dynamisch gevulde standaardwaarden tot dynamische weergaveopties[3]. De functionaliteit die hiermee wordt gerealiseerd wordt applicatielogica genoemd. Onderstaande tabel geeft een lijst weer van de meest voorkomende applicatielogica concepten en hun functie.

Concept	Functie
Default	Biedt de mogelijkheid om de waarde van velden te beïnvloeden. Hiernaast biedt het de mogelijkheid om de cursor naar een veld te verplaatsen.
Lay-out	Biedt de mogelijkheid om de beschikbaarheid van velden (zichtbaarheid en muteerbaarheid) te wijzigen.
Context	Bepaalt de beschikbaarheid van detailtabbladen, taken en rapporten. Hiernaast biedt het de mogelijkheid om het actieve detailtabblad te wijzigen.
Proces	Bepaalt na het uitvoeren van een processtroom actie welke vervolgstappen moeten worden uitgevoerd, en de volgorde van deze.
Taak	Voorziet de applicatie van specifieke functionaliteit ter ondersteuning van diverse zaken.
View	Biedt de mogelijkheid om gegevens te denormaliseren door kolommen uit verschillende tabellen te combineren in een logische tabel.
Trigger/Event	Biedt de mogelijkheid om gegevens bij te werken en de transactie terug te draaien na het opslaan, wijzigen of verwijderen van een rij.

Hiernaast zijn de modelobjecten waar deze concepten op van toepassing zijn beperkt. Voor de designer is het van belang om op de hoogte te zijn van deze mogelijkheden en beperkingen. Onderstaande tabel geeft een overzicht van de ondersteuning van applicatielogica concepten door modelobjecten.

	Tabel	View	Taak	Rapport	Proces Actie
Default	☑	☑	☑	☑	
Lay-out	☑	☑	☑	☑	
Context	☑	☑			
Proces					☑
Taak			☑		
View		☑			
Trigger/Event	☑	☑			

De Thinkwise GUI's maken gebruik van de applicatielogica concepten en geven daarbij de context van de applicatie mee, bijvoorbeeld de waarden van de huidig geselecteerde rij, de resultaten van een procesactie of de parameters van een rapport. Het is van belang dat de designer op de hoogte is van de momenten waarop de Thinkwise GUI's gebruik maakt van een bepaald applicatielogica concept en welke context parameters bij het concept beschikbaar zijn.

[3] The Business Rules Group definieert deze regels als '*Derivation business logic*' en *'Action Assertion business logic'* van het type '*Condition*' [(9)].

5.4 Applicatielogica specificaties

Applicatielogica specificaties zijn technischer van aard dan systeemrequirements. De applicatielogica specificaties beschrijven op technisch niveau *op welke manier* het systeem moet reageren in bepaalde situaties en *op welke manier* bepaalde berekeningen moeten worden uitgevoerd.

De applicatielogica specificaties worden op basis van requirements opgesteld door de designer en dienen voor de developer als basis voor het schrijven en weven van broncode.

Logica concept bepaling.
Het logica concept, de codegroep, wordt vastgesteld voor de applicatielogica specificatie. De codegroep representeert het logicaconcept waar de developer de applicatielogica in zal schrijven. De mogelijkheden op dit gebied worden bepaald door de projectversie.

Applicatielogica beschrijving
Dit onderdeel van de specificatie beschrijft welke logica door het systeem zal worden uitgevoerd. Dit zullen veelal conditionele bepalingen zijn waar de GUI door wordt aangestuurd of berekeningen die worden uitgevoerd op records.

Deze beschrijving is de basis voor geschreven broncode en moet voldoen aan de specificatie regels die zijn opgesteld voor requirement specificaties, beschreven paragraaf 5.2. Uitzondering hierop is dat applicatielogica specificaties uiteraard wél implementatiedetails beschrijven. Hiernaast wordt de prioriteit bepaald door de requirement(s) waar de applicatielogica specificatie voor is gedefinieerd.

Toewijzingstype bepaling (statisch/dynamisch)
De Thinkwise Software Factory biedt verschillende mogelijkheden om logica op een aspectgeoriënteerde manier in het systeem te plaatsen. Hierbij wordt het onderscheid gemaakt tussen statische toewijzingen (een bepaald aantal modelobjecten) en dynamische toewijzingen (een conditionele toewijzing aan de hand van attributen of structuren van modelobjecten). De designer zal bepalen welk type toewijzing van toepassing is op de specificatie.

In het geval van een dynamische toewijzing: Toewijzingsspecificatie
Wanneer de designer bepaalt dat de specificatie dynamisch wordt toegewezen, zal de designer een toewijzingspecificatie opgeven, waarin de condities voor de toewijzing worden beschreven. Dit is een tekstuele specificatie.

Het voordeel van een dynamische toewijzing is dat de logica mee zal groeien met uitbreidingen van het model. Ter illustratie, een dynamische toewijzingspecificatie: *Deze regel wordt toegepast op alle taken die worden uitgevoerd op de data-entiteit 'Factuur'.* Wanneer een nieuwe taak wordt toegewezen aan de data-entiteit *Factuur* zal de applicatielogica zonder enige aanpassing direct op deze taak worden geweven.

In het geval van een statische toewijzing: Modelobject toewijzing(en).
Een statische toewijzing geeft aan dat deze rechtstreeks aan één of meerdere modelobjecten wordt toegewezen. De designer stelt vast op welke modelobjecten de specificatie van toepassing is.

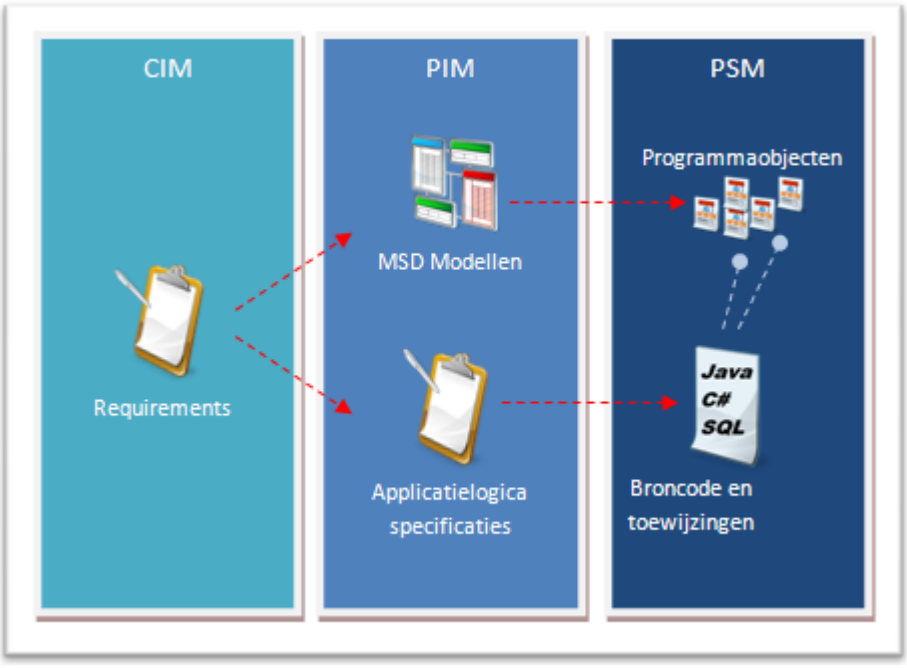
5.5 Architecturale positionering

De architectuur van eindproducten gebouwd met de Thinkwise Software Factory zijn het best weer te geven door gebruik te maken van concepten van de modelgedreven architectuur (MDA), opgesteld door de Object Management Group. Bij de modelgedreven architectuur wordt gebruik gemaakt van 3 *viewpoints*[8].

Het eerste *viewpoint* is het Computation Independent Model (CIM). Dit model vertegenwoordigd het systeem en de context van het systeem op een manier los van het digitaliseren. De requirement specificaties zijn gepositioneerd in het CIM, de requirements beschrijven het systeem en de context van het systeem op een digitaliseringsonafhankelijke wijze. Door de positionering in het CIM *viewpoint* is het mogelijk om de requirements te communiceren met partijen die niet bekend zijn met softwarebouw en de terminologie die daarbij gebruikt wordt.

Het tweede *viewpoint* is het Platform Independent Model (PIM). Hierbij wordt een oplossing geboden voor de requirements van het CIM in de vorm en structuur van een systeem, maar op een platformonafhankelijke manier. Applicatielogica specificaties bevinden zich, net als de meeste modelobjecten van de Meta Solution Definition, binnen het PIM. De aard van applicatielogica specificaties maakt deze positionering mogelijk, de specificaties zijn niet gericht op een bepaald platform, maar worden gespecificeerd met behulp van PIM modelobjecten en platform-onafhankelijke tekst.

Het derde *viewpoint* is het Platform Specific Model (PSM). Dit model biedt een platformspecifieke implementatie van het PIM. Waar in het PIM over entiteiten, attributen en events wordt gepraat, worden in het PSM termen als tabellen, methoden en triggers gebruikt. *Programmaobjecten*, worden door de Thinkwise Software Factory gegenereerd aan de hand van het *Platform Model* (PM) van het gewenste platform. Denk hierbij aan de programmaobjecten met code om de database te genereren, maar ook aan programmaobjecten die de applicatielogica concepten van een modelobject vertegenwoordigen. Applicatielogica broncode wordt toegewezen aan deze programmaobjecten.



Figuur 6 - De positionering van requirement specificaties en applicatielogica specificaties binnen de modelgedreven architectuur van eindproducten van de Thinkwise Software Factory.

5.6 Actoren betrokken bij het ontwikkelingstraject

Bij het opstellen en uitwerken van requirements zijn verschillende (groepen) mensen betrokken. Welke partijen betrokken zijn bij een project hangt sterk af van de software ontwikkelingsmethode die wordt gebruikt.

In het onderstaande overzicht van actoren wordt rekening gehouden met de Thinkwise Software Factory als software ontwikkelomgeving. Hiernaast worden ook de actoren beschreven die betrokken zijn bij de specificatie en het gebruik van de applicatielogica van een eindproduct.

Analist
De analist is de actor die de belanghebbenden identificeert en vervolgens de requirements opstelt voor de business en andere belanghebbenden.

Hij maakt op basis van documentstudies, een analyse van het bestaande systeem, interviews, workshops of andere technieken[3] een analyse van de requirements. De methoden die een analist gebruikt om kennis op te doen worden elicitaties genoemd.

Vervolgens specificeert hij de requirements volgens een aantal regels waardoor deze helder communiceerbaar zijn naar de belanghebbenden. De requirement specificaties worden gecontroleerd door de belanghebbenden. De analist zal op basis van feedback op afgekeurde requirements de specificaties verbeteren.

De systeemrequirement specificaties worden ook gecontroleerd door de designer, aangezien de designer op basis van deze specificaties verder zal werken. De analist zal op basis van feedback op afgekeurde requirements de specificaties verbeteren.

Op deze manier worden de requirements door de analist op iteratieve wijze gespecificeerd.

Designer
De designer (ontwerper) gebruikt de gespecificeerde systeemrequirements om het systeem te modelleren in de Meta Solution Definition van de Thinkwise Software Factory.

De designer valideert de juistheid van het model door middel van terugkoppeling met de belanghebbenden. Dit gebeurt door middel van demo's met de abstracte GUI's. Hiernaast kan de ontwerper gebruik maken van de modellen, zoals processtromen, ERD modellen en GUI modellen om zijn ontwerpkeuzes terug te koppelen.

Naast het modelleren zal de ontwerper op basis van systeemrequirements de specificaties van de applicatielogica opstellen. Dit zijn implementatiespecifieke beschrijvingen van logica en zullen als basis dienen van de geweven applicatielogica.

Developer
De developer (ontwikkelaar) zal de broncode in templates schrijven en de templates weven op objecten van het model, door middel van control procedures. Hierbij maakt hij gebruik van de systeemrequirements, die in de vorm van applicatielogica specificaties zijn opgesteld.

De developer stemt de juistheid van de applicatielogica specificaties met de designer af, net zoals de designer de systeemrequirements met de analist afstemt.

Belanghebbenden

De belanghebbenden (stakeholders) zijn alle partijen die met het systeem te maken hebben[3]. Denk hierbij aan de directie, verkopers, leveranciers, ICT afdelingen, applicatiebeheerders en helpdeskmedewerkers.

De belanghebbenden spelen een rol bij het eliciteren van gebruikers- en systeemrequirements. Hiernaast worden de belanghebbenden betrokken bij het valideren van de gespecificeerde requirements.

De Business

De business is de aanduiding die wordt gebruikt voor speciaal type belanghebbende dat bij elk project aanwezig is. De business is de eigenaar van het project. De bedrijfsdoelstellingen die gerealiseerd zullen worden met het project worden door de business vertegenwoordigd. Hiernaast bepaalt de business de scope van het project.

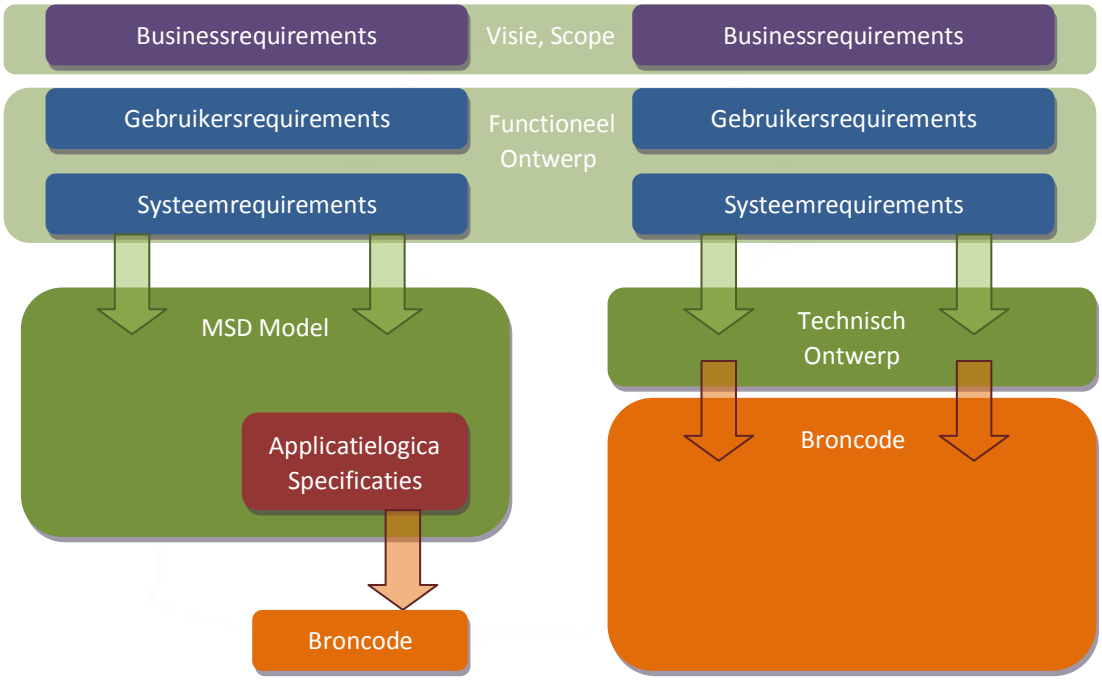
De business is de eigenaar van de businessrequirements. Bij de identificatie en validatie van deze requirement specificaties zal de business worden betrokken. De kwalificatie ‘business’ betekent niet dat voor deze belanghebbende enkel businessrequirements worden vastgelegd, de business mag ook gebruik maken van het systeem. In dit geval zal de business ook de eigenaar van gebruikers- en systeemrequirements zijn.

5.7 Positionering requirements en applicatielogica ontwikkelingstrajecten

In paragraaf 5.5 zijn de requirements gepositioneerd binnen het CIM van de modelgedreven architectuur. Dit hoofdstuk gaat in op de concrete positionering van requirements binnen het ontwikkelingstraject met de Thinkwise Software Factory, ook in vergelijking met traditionele software ontwikkelingtrajecten. Hiernaast worden applicatielogica specificaties in context geplaatst.

Het ontwikkelproces van de requirements is in grote mate afhankelijk van de projectaanpak. Bij een watervalmethode zal niet worden begonnen met het uitwerken van de systeemrequirements voordat alle specificaties compleet, gevalideerd en goedgekeurd zijn. Bij een iteratieve methode kan er voor gekozen worden om de systeemrequirements uit te werken voor (een groep) goedgekeurde business- en gebruikersrequirements. Deze keuze zal op een andere wijze tot stand komen dan bij *agile* systeemontwikkeling methoden. Het boek *'Succes de met requirements!'* beschrijft in een bijlage de impact van de verschillende projectaanpak methoden op het specificatieproces van de requirements[(3)].

In de volgende secties wordt in grote lijnen het ontwikkelproces van en met requirements en applicatielogica beschreven, los van de projectaanpak methode, maar wel gericht op de modelgedreven softwareontwikkeling van de Thinkwise Software Factory.



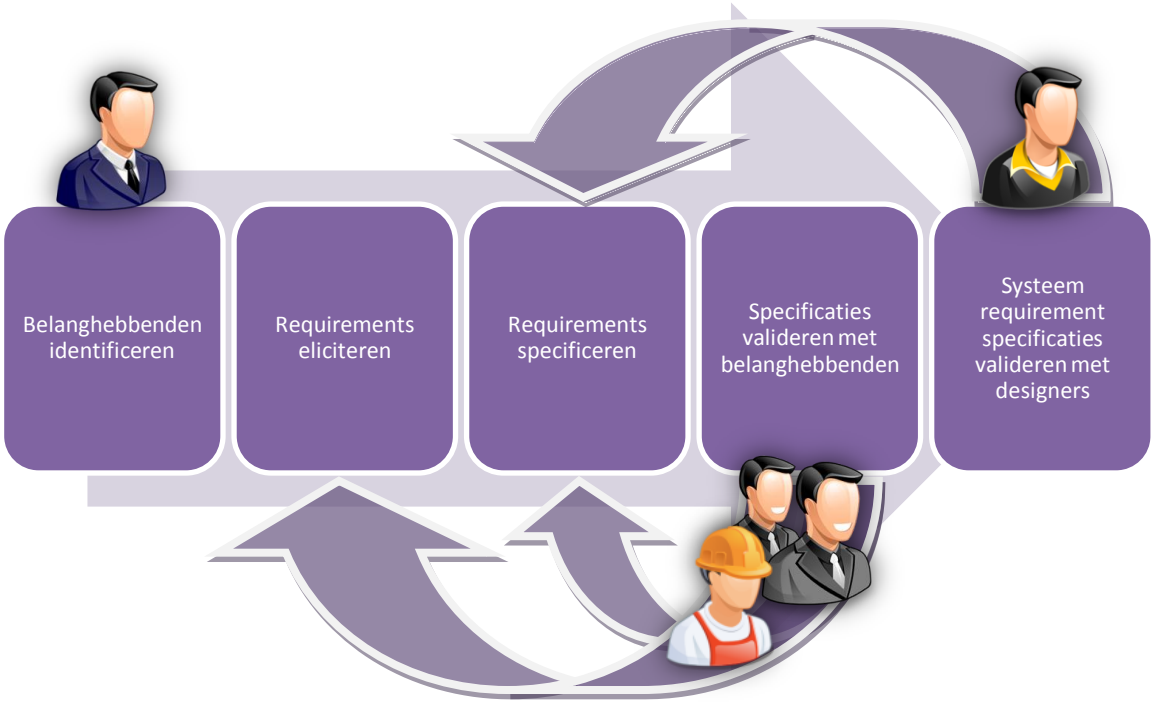
Figuur 7 - Het ontwikkelproces aan de hand van requirement in de traditionele softwareontwikkeling (rechts) en de Thinkwise Software Factory (links)

In de traditionele softwareontwikkeling worden de systeemrequirements als tekst en modellen op een implementatiespecifieke manier uitgewerkt in een technisch ontwerp.- Op basis daarvan wordt de code van de eindapplicatie geschreven.

Bij de Thinkwise Software Factory wordt het technisch ontwerp gemodelleerd in de MSD. De applicatielogica specificaties worden als tekst gespecificeerd en dienen als basis voor de broncode. Doordat alle tussenproducten in de Thinkwise Software Factory worden opgeslagen, is de traceerbaarheid tussen de modellen en ontwerpen vele malen groter. Hiernaast wordt de hoeveelheid broncode gereduceerd, niet alleen door de generatie van het eindproduct en de interpretatie van het model, maar ook door het aspectgeoriënteerd weven van de applicatielogica. Dit geeft namelijk de mogelijkheid voor hergebruik van broncode.

De eerste actor die bij het ontwikkelproces naar voren komt is de analist. De eerste actie die de analist normaal gesproken uitvoert is het identificeren van de belanghebbenden. Dit is een zeer belangrijke stap bij het ontwikkelproces, wanneer een belanghebbende niet wordt betrokken bij de elicitatie is de kans groot dat eisen over het hoofd worden gezien.

Op basis van de belanghebbenden zal de analist de elicitatietechnieken bepalen. Dit zijn de technieken die de analist toepast om de requirements te achterhalen. Denk hierbij aan workshops en interviews.



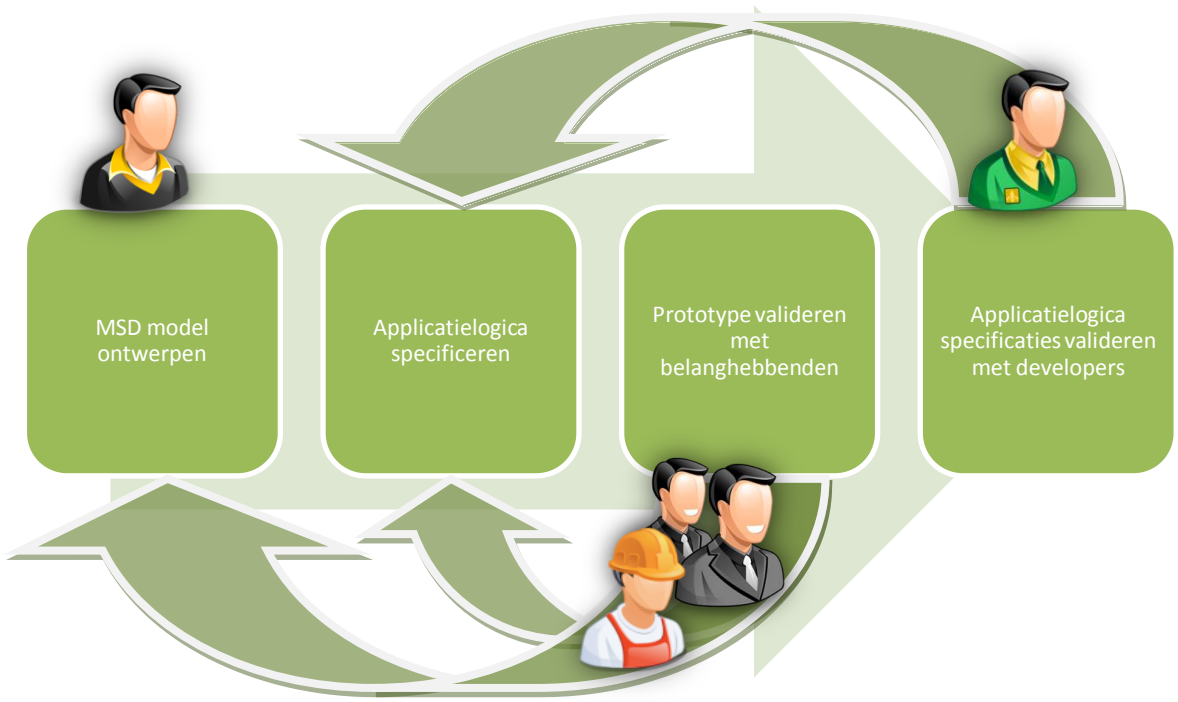
Figuur 8 – Het specificatie proces van requirements door de analist

Door middel van de vooraf gedefinieerde technieken zoals interviews, workshops, analyse van bestaande systemen en/of andere technieken zal de analist de kennis elicitaties uitvoeren. Een veelgebruikte aanpak voor interviews is het gebruik van usecase modellen en scenario beschrijvingen. De analist zal de resultaten van een elicitatie vastleggen in bijvoorbeeld interviewrapporten.

Door het analyseren van de kennis elicitatie resultaten zal de analist de business-, gebruikers-, en systeemrequirements specificeren. Door het specificeren van de businessrequirements wordt de visie en scope van het project duidelijk. De gewenste functionaliteit van het projectresultaat zal blijken uit de gebruikersrequirements. Veruit de meeste requirements zijn functionele systeemrequirements. Om van geëliciteerde kennis tot correcte requirements te komen, wordt een bepaald proces gevolgd, waarbij de requirements steeds verder wordt uitgesplitst en gedetailleerder wordt beschreven, tot het atomaire niveau wordt bereikt.

De analist biedt de gespecificeerde requirements ter validatie aan bij de belanghebbenden. Wanneer deze niet voldoende zijn zullen de specificaties worden verbeterd. Mogelijk begint het proces opnieuw vanaf de elicitatie fase. Vervolgens worden de gespecificeerde systeemrequirements ter validatie aangeboden bij de designers. De goedkeuring van de designer is nodig, aangezien de designers op basis van deze specificaties het model zal bouwen.- Wanneer deze niet voldoende zijn zal de analist de specificaties verbeteren aan de hand van de feedback van de developers.

Nadat de systeemrequirements geaccepteerd zijn, gaat de designer aan de slag met het bouwen van een initieel model. Een datamodel wordt opgesteld, processtromen worden gedefinieerd, het GUI model wordt vastgelegd, de processtromen worden gedefinieerd en helpteksten worden geschreven.



Figuur 9 – Het uitwerken van systeemrequirements door de designer

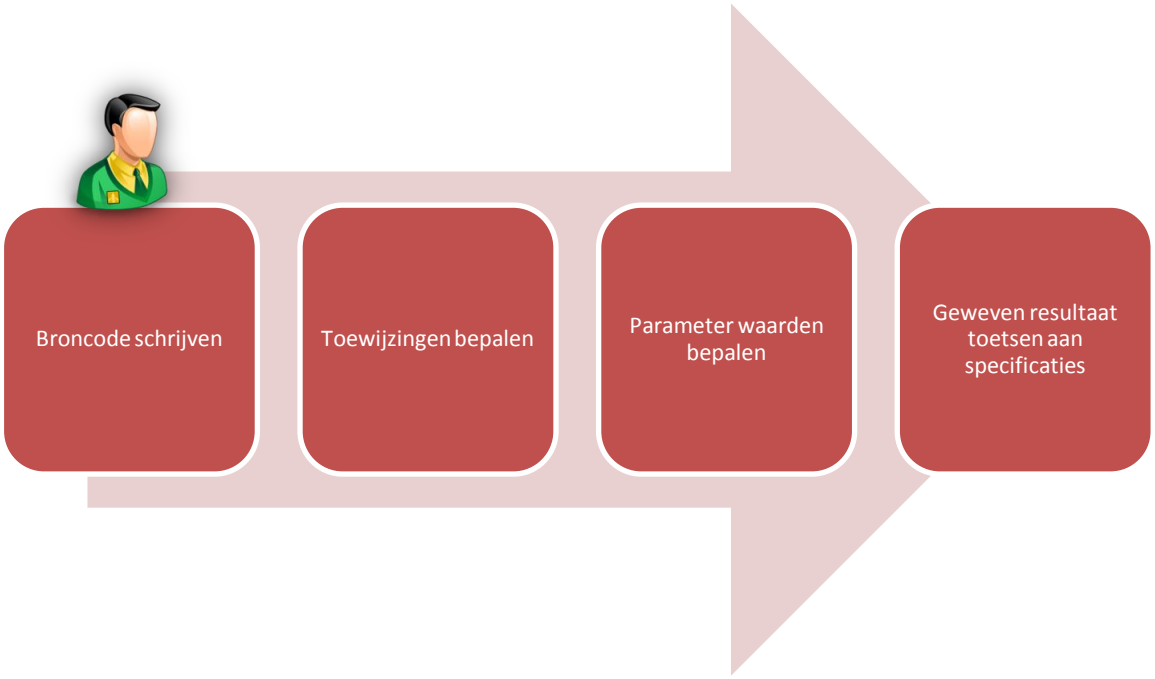
Niet alle systeemrequirements worden (volledig) door middel van modellen worden gespecificeerd. Wanneer bepaalde applicatielogica in het model moet worden opgenomen, zal de designer een tekstuele specificatie opstellen van een regel applicatielogica. Deze dient als basis voor het weven van broncode door de developer. Hierbij wordt aangeraden om te voldoen aan de eisen die zijn opgesteld voor requirement specificaties. Daarnaast moet rekening worden gehouden met de Thinkwise concepten voor applicatielogica.

Door de unieke architectuur van de Thinkwise Software Factory, door middel van de runtime interpretatie van het model, kan op dit moment in tijd een prototype van het eindproduct worden getoond. Dit product bevat nog geen applicatielogica, maar is wel uitgerust met alle gegevens van het data- en GUI model. Op basis hiervan kunnen de belanghebbenden het eindproduct beoordelen en feedback geven op het product. De designer kan deze feedback verwerken in het model. De applicatielogica specificaties worden in dit stadium gebruikt om de belanghebbenden een idee te geven van de functionaliteit waarmee het prototype zal worden uitgebreid.

Uiteindelijk zullen de applicatielogica specificaties ter validatie worden aangeboden aan de developer. De developer zal deze beoordelen op juistheid en correctheid.- Wanneer de applicatielogica specificaties niet voldoende zijn om de broncode te schrijven en te weven, zullen de specificaties aan de hand van de feedback van de developer worden verbeterd.

Nadat de belanghebbenden het product en de tekstuele applicatielogica specificaties hebben goedgekeurd, en de developer de applicatielogica specificaties hebben goedgekeurd, kan de programmeer fase beginnen. Hoe strak de grens ligt tussen deze fasen hangt af van de methode die wordt gebruikt voor de softwareontwikkeling.

Nadat applicatielogica specificaties zijn geaccepteerd, gaat de developer aan het werk om de specificaties in code om te zetten en in het eindproduct te positioneren. Dit onderdeel is vrij technisch van aard en is daarom niet uitgebreid beschreven.



Figuur 10 - Het schrijven en weven van applicatielogica door de developer

Per applicatielogica specificatie maakt de developer een vertaalslag van specificatie naar code. Wanneer meerdere specificaties structureel op elkaar lijken, heeft de developer de mogelijkheid om abstractie aan te brengen in de code. Op deze manier wordt het systeem beter onderhoudbaar, omdat er minder broncode is. De developer kan hierbij gebruik maken van parameters in de betreffende onderdelen broncode.

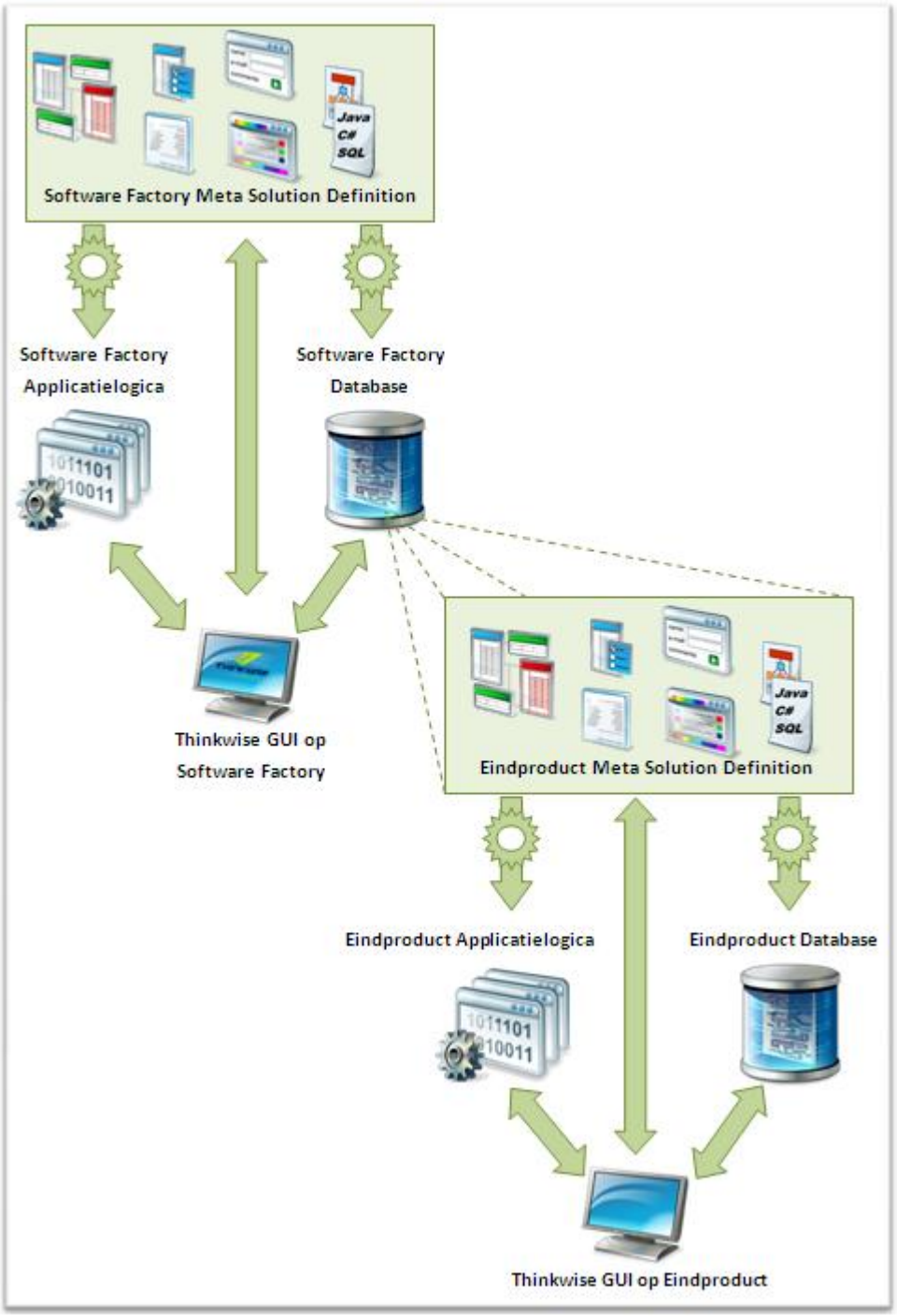
Aan de hand van de weefbeschrijving of statische toewijzingsspecificatie zal de developer de code in het model weven. Wanneer de developer gebruik maakt van een abstractie door middel van parameters, zal de developer de parameters vullen per toewijzing.

De developer zal het resultaat van het weven van de code toetsen aan de specificaties. Hierbij kan ook gebruik worden gemaakt van de applicatielogica specificaties.

6 Implementatie

6.1 Architectuur Thinkwise Software Factory

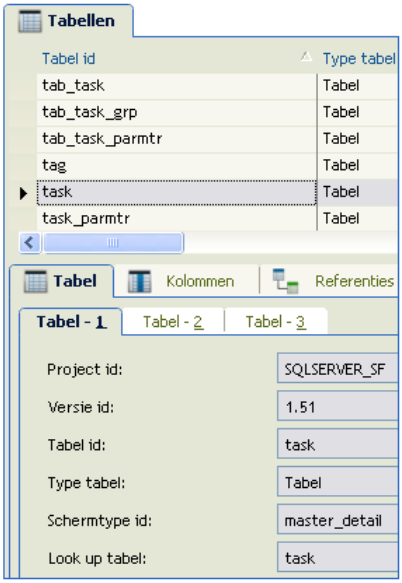
Om de realisatie van de uitbreiding in context te zetten is het van belang om kennis te hebben van de architectuur van de Thinkwise Software Factory. Hoofdstuk 2 beschrijft de architectuur van eindproducten van de Thinkwise Software Factory. Hierop voortbordurend, de huidige versie van de Thinkwise Software Factory is een eindproduct van de voorgaande versie van de Thinkwise Software Factory. De architectuur van de Thinkwise Software Factory is dus gelijk aan de architectuur van eindproducten gemaakt met de Thinkwise Software Factory.



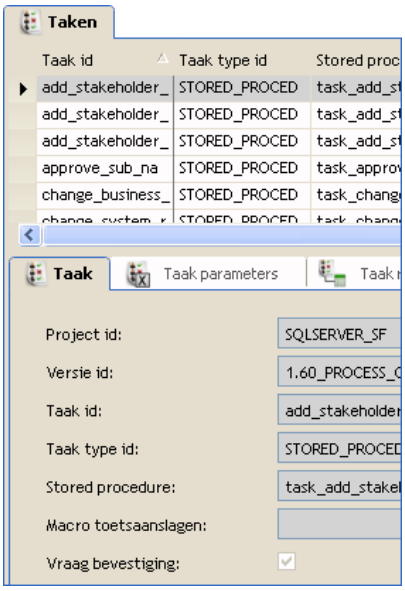
Figuur 11 - De architectuur van de Thinkwise Software Factory. De Thinkwise Software Factory is zelf een eindproduct van de Thinkwise Software Factory en gebruikt een Thinkwise GUI om zijn MSD model te interpreteren.

Ter illustratie, de taak '*Voeg belanghebbende toe'* in een eindproduct is een record in de tabel *Taken* van de MSD van het eindproduct. In dit record en zijn detailrecords staan alle meta-gegevens van de taak '*Voeg belanghebbende toe'* gedefinieerd, alle eigenschappen van deze specifieke taak.

De tabel '*Taken'* in de MSD van het eindproduct is op zijn beurt een record in de tabel *Tabellen* van de MSD van de Thinkwise Software Factory. In dit record staan alle meta-gegevens van de tabel '*Taken'* beschreven, de attributen die voor taken gespecificeerd kunnen worden via de Thinkwise Software Factory in de MSD van een eindproduct.



Figuur 12 - Het *tabel* record 'task' in de MSD van de Thinkwise Software Factory



Figuur 13 - Het *taak* record 'add-stakeholder' in de MSD van het eindproduct

Figuur 14 – De taak 'Voeg belanghebbende toe' in het eindproduct.

Wanneer aanpassingen worden gedaan aan de Thinkwise Software Factory, zal een nieuwe versie worden aangemaakt van model van de Thinkwise Software Factory. In deze nieuwe versie van het Thinkwise Software Factory model worden de nieuwe uitbreidingen gemodelleerd. Deze uitbreidingen kunnen vervolgens worden toegepast op eindproducten wanneer deze versie van de Thinkwise Software Factory in gebruik wordt genomen.

Voor de uitbreidingen van de Thinkwise Software Factory is versie 1.60 van de MSD van de Thinkwise Software Factory aangemaakt. Deze MSD wordt bewerkt door middel van versie 1.52 van de Thinkwise Software Factory. In de toekomst, wanneer versie 1.60 in gebruik is genomen, wordt het mogelijk om requirements te gebruiken bij het ontwikkelen van eindproducten, dus ook bij het ontwikkelen van de volgende versies van de Thinkwise Software Factory.

6.2 Gegevensstructuur

Zoals in de vorige paragraaf staat beschreven, alle uitbreidingen die nodig zijn voor een nieuwe versie van de Thinkwise Software Factory worden gemodelleerd door middel van de Thinkwise Software Factory, in een nieuwe versie van de MSD van de Thinkwise Software Factory.

Wanneer de nieuwe versie van de Thinkwise Software Factory beschikbaar wordt gesteld, zijn de nieuw geïntroduceerde modelobjecten beschikbaar voor gebruik bij de bouw van eindproducten.

Bij alle gegevensobject definities is als onderdeel van de *primary key* een *foreign key* naar een project en een projectversie opgenomen, tenzij anders aangegeven. Een projectversie representeert een instantie van het MSD. Welke requirements zijn gespecificeerd, wat hun prioriteit en implementatiestatus is, is afhankelijk van de versie waar naar gekeken wordt. Over een lange periode van tijd veranderden de requirements, komen nieuwe requirements naar voren en verouderen bepaalde requirements. Een projectversie geeft de requirement situatie weer van een bepaald moment in de *lifecycle* van een product.

Veel van de entiteiten en attributen voor requirements zijn ontworpen om het proces te ondersteunen dat wordt doorlopen bij softwareontwikkeling aan de hand van requirements. Daarnaast wordt de opbouw van het datamodel bepaald door de eisen en richtlijnen die door Arendsen en IEEE zijn gesteld. Deze onderwerpen zijn terug te vinden in paragraaf 4.

Hiernaast is onderzoek gedaan naar andere requirement beheertools, onder andere Rational® RequisitePro®, OSRMT en eRequirements™. Ook is feedback verwerkt van gebruikers van de Thinkwise Software Factory, zowel intern bij Thinkwise Software als bij bedrijven die gebruik maken van de Thinkwise Software Factory.

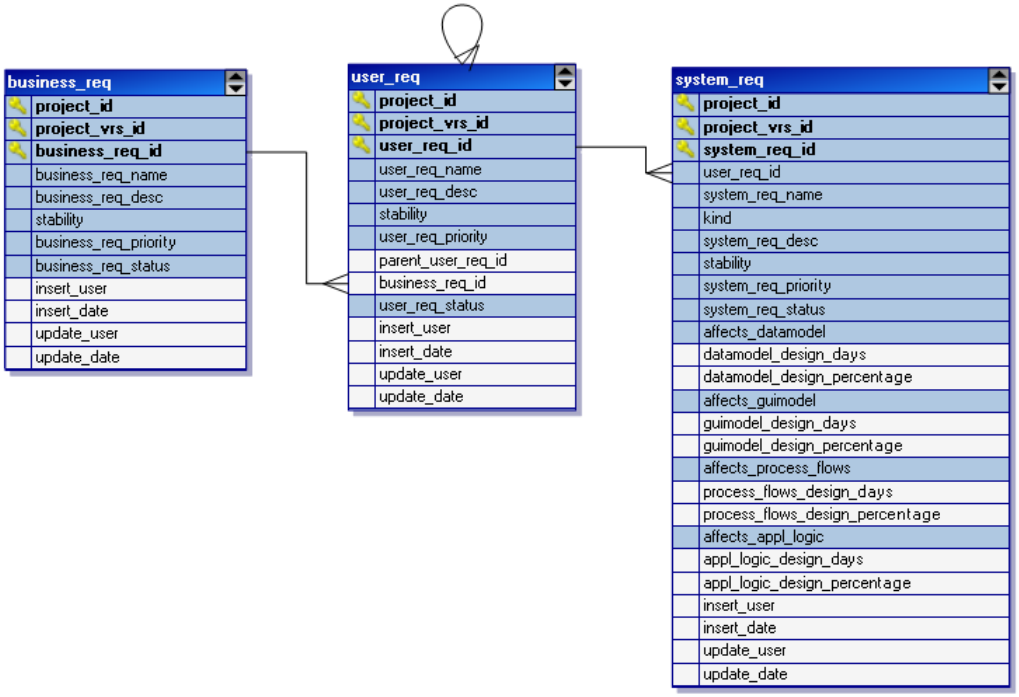
De definitie van applicatielogica specificaties is gebaseerd op de huidige werkwijze van Thinkwise Software en bevat daarmee alle relevante attributen. Dit is vooral gebaseerd op de beschikbare Microsoft Word templates die door Thinkwise Software worden aangeboden voor het specificeren van applicatielogica.

Figuur 15 - Het volledige datamodel van requirements en applicatielogica. In deze paragraaf worden alleen de belangrijkste gegevensentiteiten besproken.

Requirements

Requirements in de Thinkwise Software Factory zijn onderverdeeld in 3 niveaus, business-requirements, gebruikersrequirements en systeemrequirements. Deze zijn hiërarchisch ingedeeld, een gebruikersrequirement kan verwijzen naar een businessrequirement en/of een bovenliggende gebruikersrequirement. Een systeemrequirement moet verwijzen naar een gebruikersrequirement.

Hiernaast kunnen gebruikersrequirements ook onderling hiërarchisch worden ingedeeld. Op deze manier worden de systeemrequirements, die veruit het grootste deel van de requirements vormen, ingedeeld op de functie die ze ondersteunen. Hierdoor worden de requirements beheersbaar.



Figuur 16 - De hiërarchische indeling van de requirements is terug te zien in het datamodel.

Zoals uit het model blijkt zijn requirements niet ondergebracht in één tabel met een typebepaling. De reden hiervoor is het verschil in attributen van de verschillende typen requirements. Door per type een tabel te gebruiken wordt de applicatielogica die nodig is om de data correct te houden en juist te presenteren flink verminderd.

Zoals uit het datamodel blijkt hebben alle requirements enkele attributen gemeen. Dit is de *identifier*, de naam, een omschrijving, een stabiliteitsindicatie, een prioriteitsindicatie en een status.

Hiernaast kan bij een systeemrequirement een implementatieschatting worden gedaan en kan een implementatiestatus per modeltype[4] worden opgegeven.

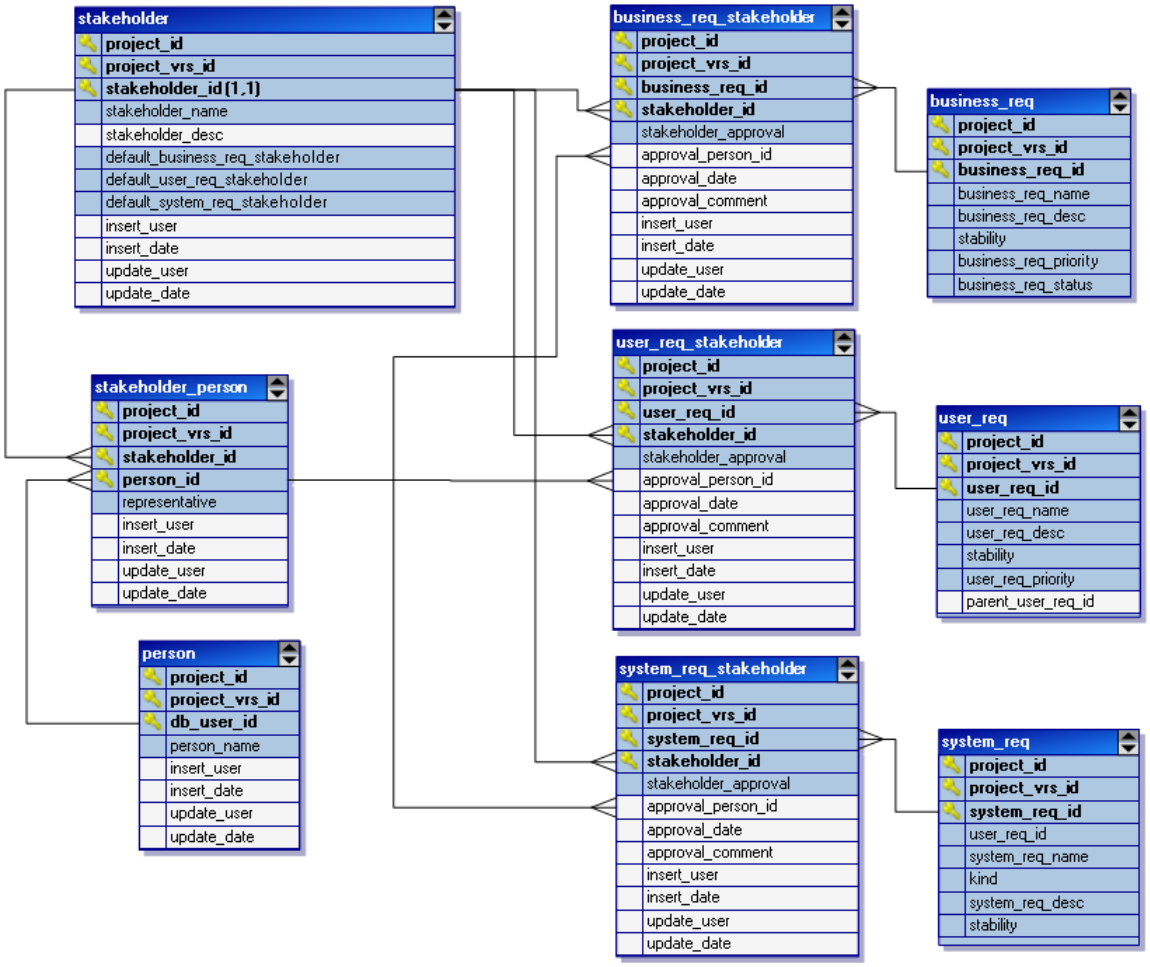
Alleen bij systeemrequirements wordt de soort opgegeven (functioneel/niet-functioneel). De reden hiervoor is dat alleen op het niveau van systeemrequirements eigenschappen van het systeem worden beschreven. Pas op dit niveau kan onderscheid worden gemaakt tussen functioneel en niet-functioneel.

[4] Datamodel, GUI model, Processtromen model en Applicatielogica Specificaties

Belanghebbenden

De belanghebbenden worden geregistreerd in een aparte tabel. Personen kunnen 'lid' zijn van meerdere belanghebbenden. Per 'lidmaatschap', geregistreerd in de koppeltabel tussen persoon en belanghebbende, wordt aangegeven of deze persoon de belanghebbende vertegenwoordigt. Alleen vertegenwoordigers mogen hun oordeel uitspreken over de requirements.

Hiernaast kan bij een belanghebbende worden aangegeven of deze automatisch bij nieuwe requirements van een bepaald type gekoppeld moet worden. Dit zal vaak het geval zijn bij de *Designers* belanghebbenden, die alle systeemrequirements moeten goedkeuren, aangezien zij de vertaalslag tussen de systeemrequirements en de PIM modellen maken.



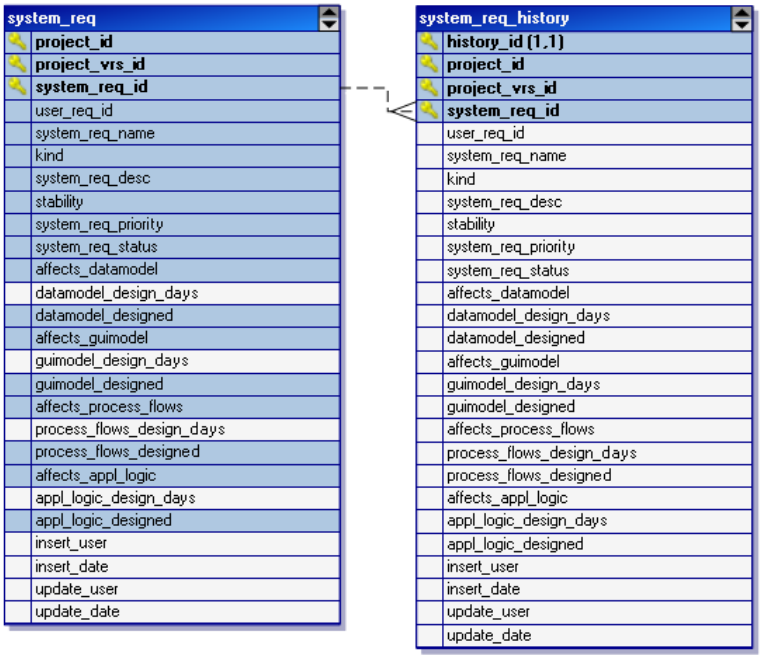
Figuur 17 - De datastructuur van belanghebbenden

Goedkeuring over een requirement wordt geregistreerd in de koppeltabel tussen requirements en belanghebbenden. Applicatielogica is aanwezig om de goedkeuringsstatus te veranderen wanneer er veranderingen zijn in de goedgekeurde requirements. Goedkeurig wordt verleend namens een persoon van de betreffende belanghebbende.

Requirement historie

De historie tabeldefinities zijn dynamisch gegenereerd. Hiervoor is een procedure geschreven.

Deze procedure maakt een kopie van de brontabel definitie, de brontabel kolom definities en de brontabel referentiedefinities. De referentiedefinities, die een database referentie en/of detailtabblad/lookup knop in een GUI aangeven, worden echter niet integer gemaakt. Op deze manier zal de historie niet problemen opleveren het verwijderen van records of het wijzigen van de primary key. Hiernaast wordt een identitiy veld aan de primary key toegevoegd, om zo meerdere historie records toe te staan.



Figuur 18 - De historie tabel van de systeemrequirements tabel

Vervolgens weeft de procedure enkele stukken broncode in de triggers van de tabellen, die er voor zorgen dat de historie tabel bij wijzigingen wordt bijgewerkt

Deze constructie zorgt er voor dat wijzigingen aan de modelobjecten waar een historie modelobject voor wordt gegenereerd, direct door de procedure worden doorgevoerd naar het historie modelobject en de bijbehorende triggers.

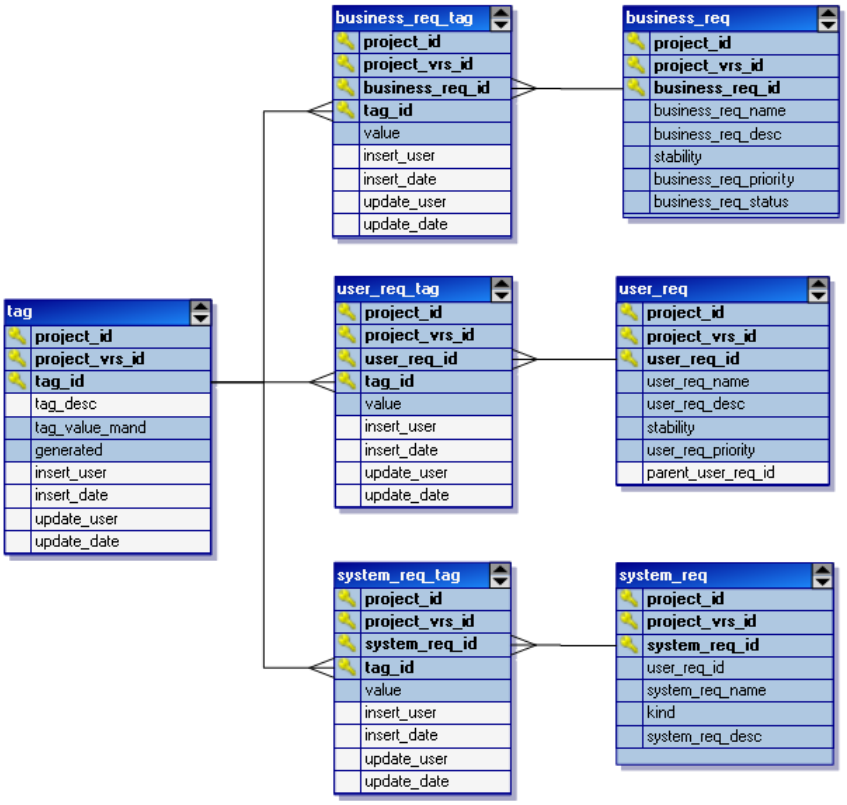
Requirement tags

Tags zijn labels, handmatig gedefinieerde velden die aan modelobjecten worden gekoppeld.

Tags worden vaak gebruikt om bepaalde functies in het model te simuleren tot een permanente oplossing is geïntroduceerd in de Thinkwise Software Factory. Op record niveau kunnen door middel van tags gebruikersgedefinieerde velden en veldwaarden worden vastgelegd.

Hiernaast kunnen tags worden gebruikt door procedures die het model dynamisch uitbreiden. Bijvoorbeeld, de modelobjecten *business_req*, *user_req*, *system_req* en *appl_logic* zijn gekoppeld aan de tag '*generate_history*'. Deze tag wordt gebruikt door een procedure die automatisch historietabellen genereert voor deze modelobjecten. .

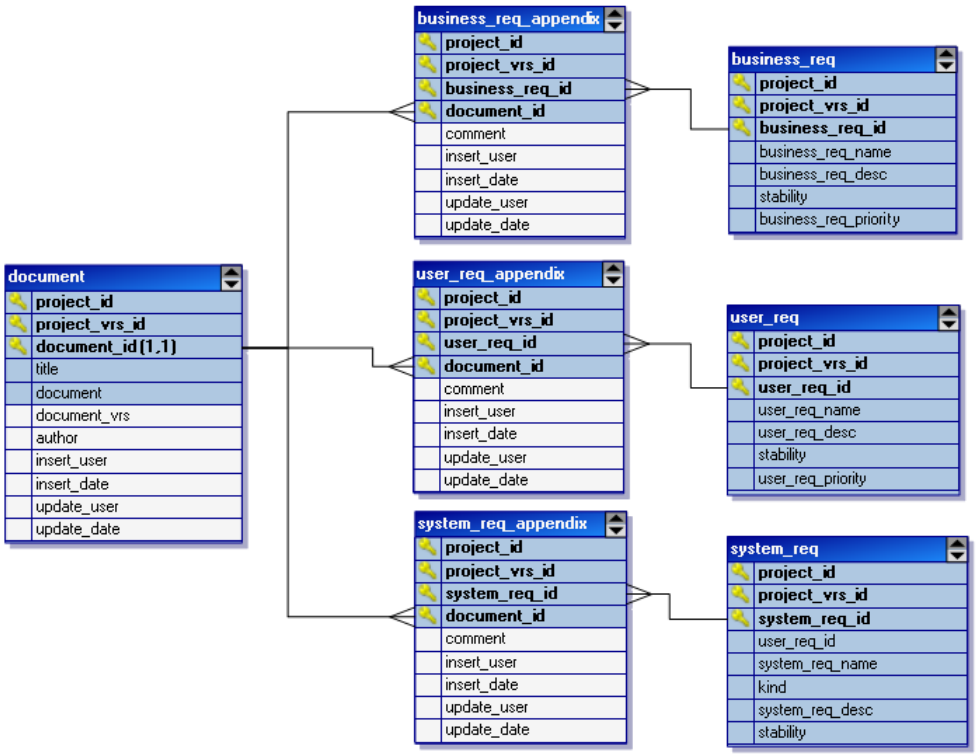
Ook kunnen tags voor *Business Intelligence* views of componenten worden gebruikt, om in deze componenten bepaalde records snel te kunnen identificeren en groeperen.



Figuur 19 - Requirement tags

Requirement bijlagen

De documenten tabel verzorgt de bijlagen van de verschillende requirements. In de koppeltabel tussen de requirements en de documenten kan commentaar worden opgenomen.



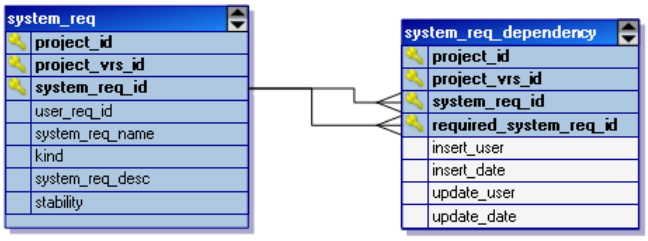
Figuur 20 - De datastructuur van requirement bijlagen

Van een document kunnen verschillende attributen worden opgeslagen ter ondersteuning van de *Backwards Traceability*, zoals de versie en de auteur.

Systeemrequirement afhankelijkheden

Wanneer een systeemrequirement wordt geïmplementeerd, kan de implementatie afhankelijk zijn van de implementatie van een andere systeemrequirement. Bijvoorbeeld, voor de implementatie van een systeemrequirement met betrekking tot *het berekenen van een productprijs* moet eerst de systeemrequirement *registreren van btw tarieven* worden geïmplementeerd.

De analist wordt de mogelijkheid geboden om deze afhankelijkheden te definiëren. Van de analist mag niet verwacht worden dat alle afhankelijkheden worden gemodelleerd, maar wanneer kritieke afhankelijkheden zijn aangegeven kan dit helpen bij de besluitvorming met betrekking tot de implementatievolgorde.

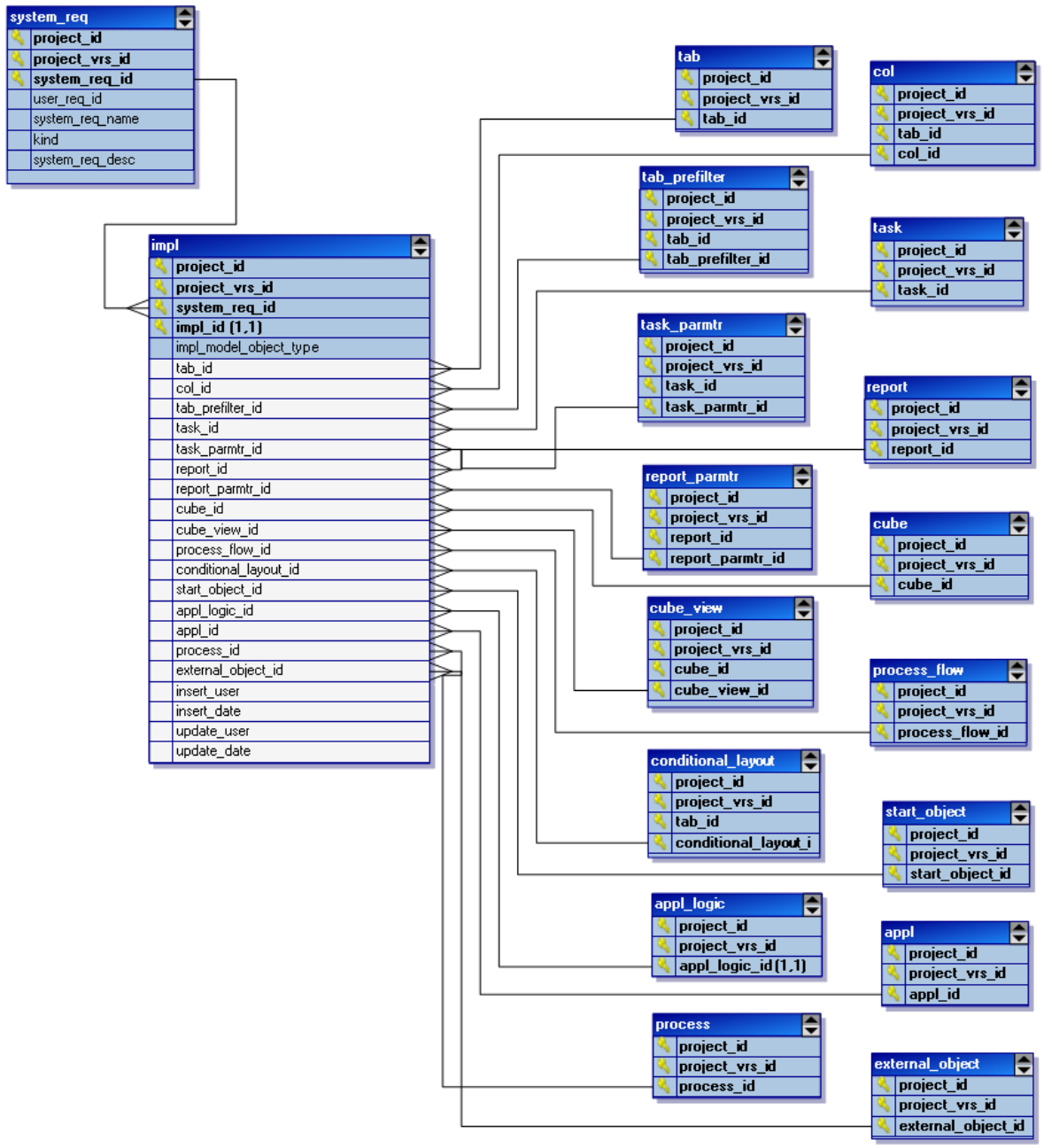


Figuur 21 - Systeemrequirement implementatieafhankelijkheden

Systeemrequirement implementaties

Wanneer de designer aan de slag gaat met het implementeren van de systeemrequirements in het PIM model, moet de traceerbaarheid tussen een requirement en de gemodelleerde objecten worden bijgehouden. Voor dit doeleinde is een tabel toegevoegd die referenties kan leggen tussen systeemrequirements en relevante modelobjecten. De lijst met modelobjecten is geselecteerd aan de hand van de vraag *'Welke modelobjecten vertegenwoordigen de functionaliteit van het eindproduct'*.

Views zijn beschikbaar gesteld om het koppelen van tabellen te vergemakkelijken.

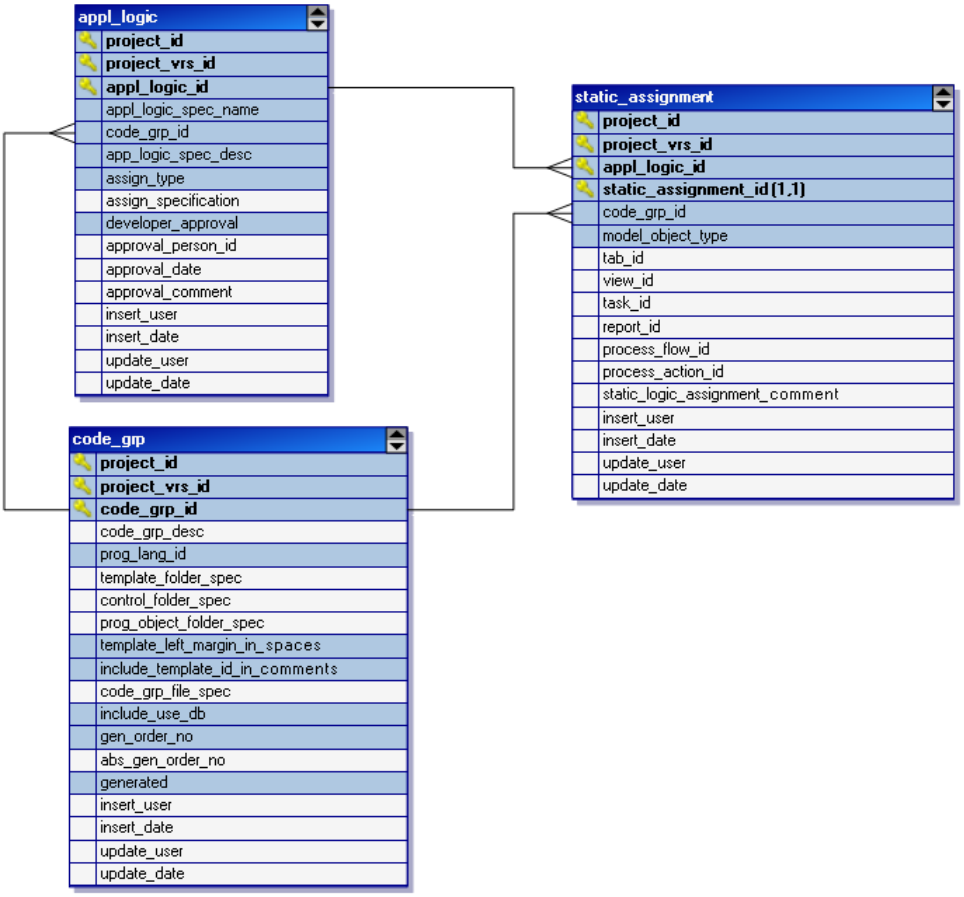


Figuur 22 - De traceerbaarheid tussen het CIM en het PIM, de systeemrequirement implementatie tabel

Een andere mogelijkheid bij het realiseren van de traceerbaarheid is een overervingrelatie creëren waarbij alle modelobjecten erven van dezelfde supertabel. Deze supertabel zal dan de modelobject sleutel en het modelobject type bevatten. Op deze manier is er slechts 1 referentie nodig tussen de implementatietabel en de supertabel. De impact hiervan is echter vooralsnog te groot.

Applicatielogica specificaties & statische toewijzingen

Applicatielogica specificaties zijn onder te verdelen in twee typen, statisch en dynamisch. Statische applicatielogica specificaties hebben betrekking op enkele voorgedefinieerde modelobjecten, dynamische applicatielogica specificaties worden geweven aan de hand van condities. Het datamodel biedt de mogelijkheid om meerdere statische toewijzingen te doen voor statische applicatielogica specificaties. Voor dynamische applicatielogica specificaties wordt een toewijzingsbeschrijving opgegeven.



Figuur 23 – Applicatielogica specificaties en statische toewijzingen

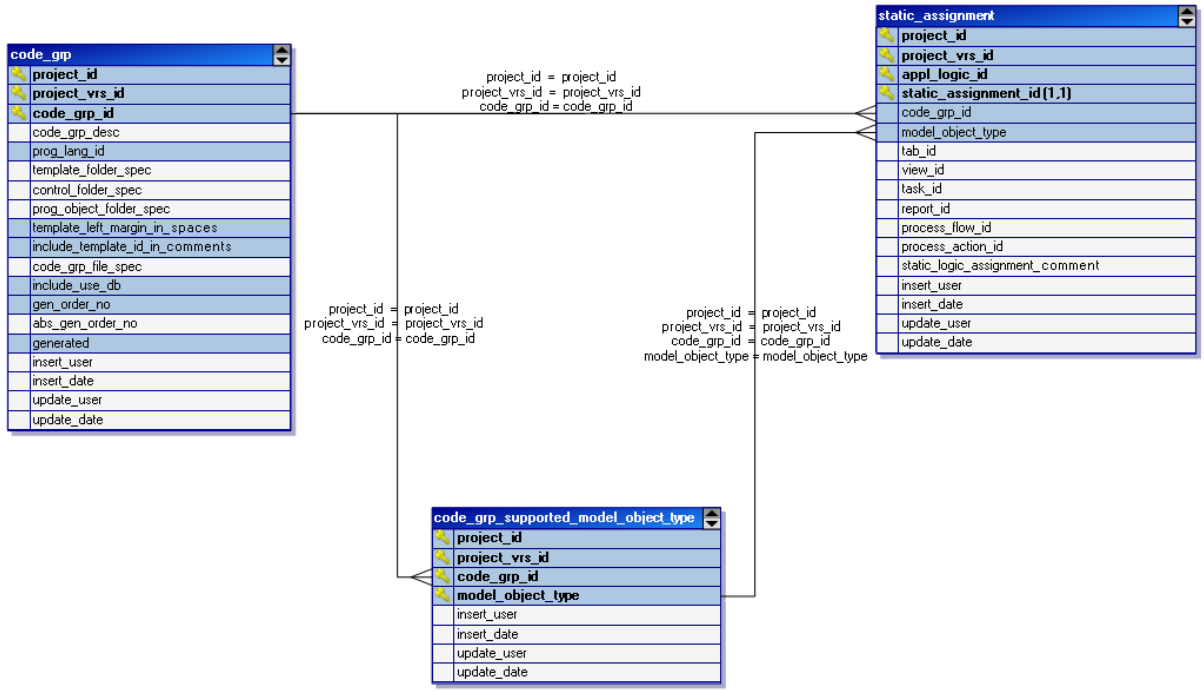
Een codegroep representeert een logica concept. Het logica concept van een statische toewijzing is uiteraard gelijk aan deze van een applicatielogica specificatie.

Ondersteunde applicatielogica concepten per modelobject type

De reden dat de codegroep ook is opgenomen in de statische toewijzing is om de beschikbare modelobject typen snel te filteren op de mogelijkheden die de codegroep heeft, met betrekking tot modelobjecten. Zoals eerder staat beschreven, logica concepten hebben betrekking op specifieke modelobjecten.

Onderstaand diagram geeft de relatie weer waardoor integriteit wordt afgedwongen tussen de mogelijke modelobject typen die kunnen worden gespecificeerd voor een bepaalde codegroep.

De *code_grp_supported_model_object_type* tabel is vooraf gevuld.



Door middel van *Lay-out* applicatielogica is het mogelijk om alleen het modelobject veld te laten zien van het geselecteerde type, bijvoorbeeld het veld *task_id* wanneer het model_object_type 'Taak' is gekozen. Deze velden verwijzen naar alle modelobjecten van het betreffende type, zie onderstaand diagram.

Overig

Naast bovenstaande tabellen is een koppeltabel tussen applicatielogica specificaties en *control procedures*, die applicatielogica broncode en broncode toewijzingen vertegenwoordigen. Deze koppeling wordt aangebracht wanneer op basis van een applicatielogica specificatie een control procedure wordt aangemaakt.

Ook is een historie tabel aanwezig voor de applicatielogica specificaties, opgebouwd volgens dezelfde structuur als de requirements historietabellen.

Hiernaast is een tabel aanwezig waarin dubbelzinnige uitdrukkingen worden gedefinieerd. De requirements en applicatielogica specificaties lichten op wanneer een dubbelzinnige uitdrukking voor komt in de beschrijving.

6.3 Gebruikersinterface

Deze paragraaf beschrijft de uitbreidingen aan de grafische gebruikersinterface van de Thinkwise Software Factory.

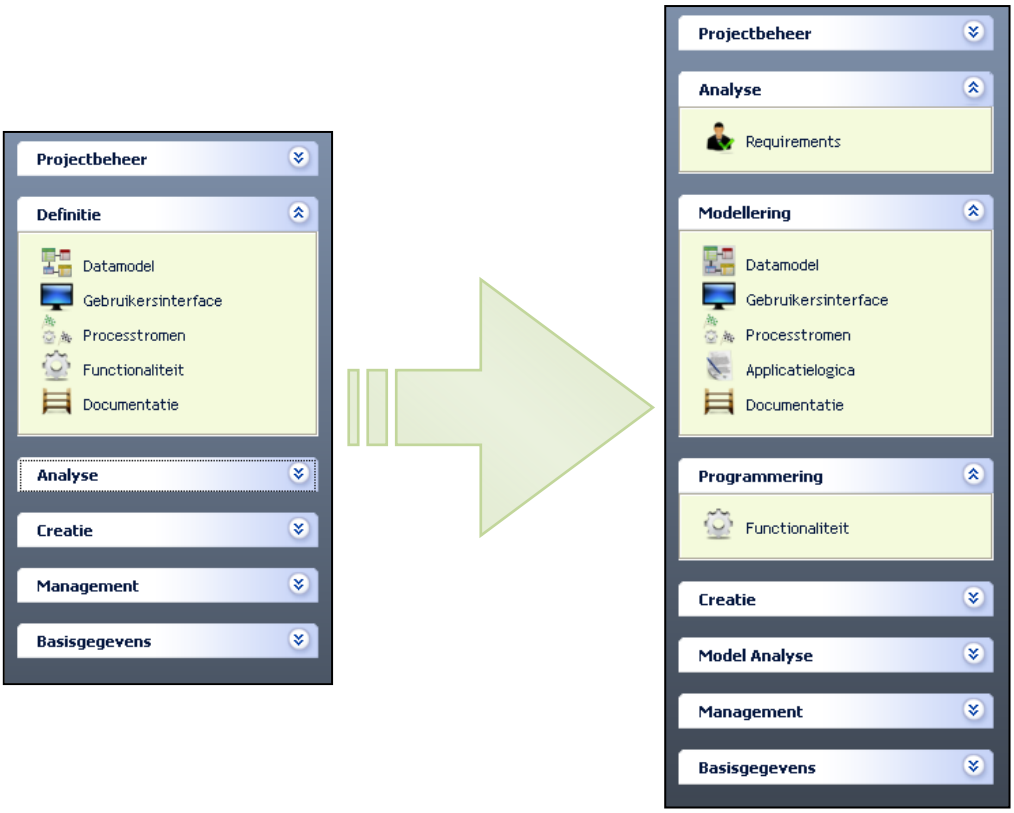
De meeste eigenschappen van de grafische gebruikersinterface zijn gemodelleerd in het metamodel van de Thinkwise Software Factory.

Één van de belangrijkste weergave concepten, die ook het meest is toegepast bij de implementatie, is het master-detail concept. Hierbij wordt een scherm op een bepaald onderwerp geïnitialiseerd en worden detailtabbladen aangemaakt voor detail referenties. Deze detailtabbladen zijn in feite een nieuw master-detail scherm en bevatten op hun beurt ook detailtabbladen.

Enkele specifieke onderdelen van bepaalde schermen moeten overeen komen met andere modelleerschermen van de Thinkwise Software Factory. Hierbij is gebruik gemaakt van het concept *Processcherm*. Een processcherm is een alternatief voor de standaardschermen, dat door middel van C# code naast het model interpreteren ook kan afwijken van de standaard.

6.3.1 Menustructuur

De menustructuur is gewijzigd om het analyseren van requirements een plek te geven binnen het proces. Hiernaast biedt de nieuwe menustructuur een betere schreiding tussen de verschillende modellen en de verschillende rollen van de actoren die met deze modellen werken. Hierbij is de menustructuur opgedeeld in de verschillende rollen (analist, designer, developer), die ook overeen komen met de verschillende modeltypen van de modelgedreven architectuur (CIM, PIM, PSM).



Figuur 24 - De oude (links) en de nieuwe (rechts) menustructuur van de Thinkwise Software Factory

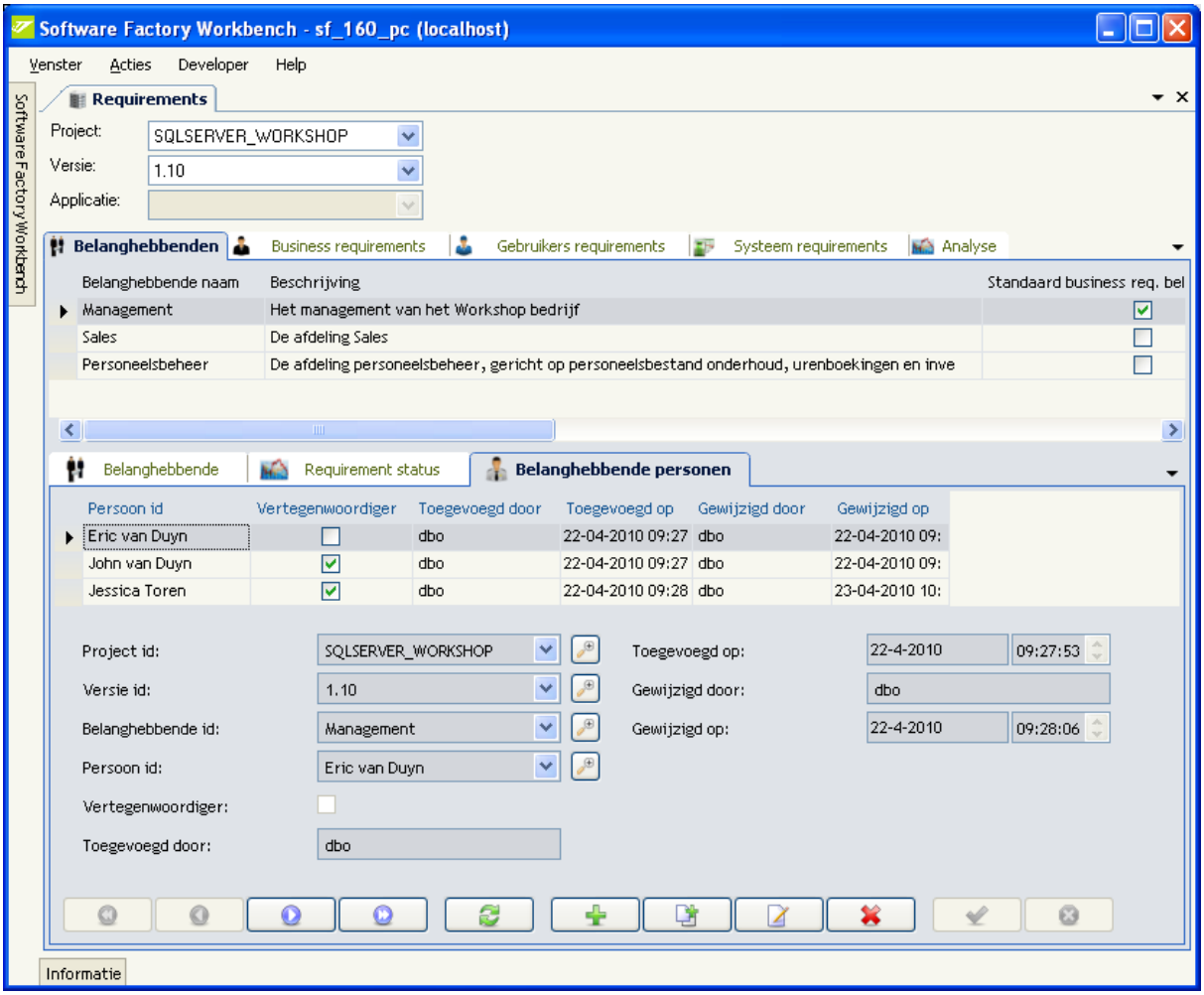
6.3.2 *Requirements Scherm*

In het MSD Model van de Thinkwise Software Factory is in de menustructuur een menu-item toegevoegd ter ondersteunen van het proces met betrekking tot de requirements. Voor dit doel is het requirements scherm geïntroduceerd. Dit scherm bestaat uit een project/projectversie selectie component en een vijftal tabbladen.

Het eerste tabblad is genaamd 'Belanghebbenden'. De analist gebruikt dit tabblad voor het overzicht van de belanghebbenden. In dit tabblad wordt een master-detail weergegeven, geïnitialiseerd op de belanghebbenden tabel. In het model zijn voor dit onderwerp enkele detailreferenties voor dit onderwerp geactiveerd.

- Het eerste detailtabblad bevat een formulier om datamanipulatie uit te voeren voor de tabbladen.
- Het tweede tabblad is een *Business Intelligence* scherm, dat doorsneden biedt met betrekking tot de requirements van de geselecteerde belanghebbende.
- Het derde tabblad biedt de mogelijkheid tot overzicht en datamanipulatie met betrekking tot de personen die gekoppeld zijn aan de belanghebbende. In onderstaand schermafdruk is dit tabblad geactiveerd.

Hiernaast kan vanuit het Belanghebbenden tabblad de status van de gekoppelde requirements via een taak uit het contextmenu (het rechtermuisknop menu) worden gewijzigd. Ook kan een rapportage op dezelfde wijze worden afgedrukt. Een voorbeeldrapportage is opgenomen als bijlage (Bijlage 2).



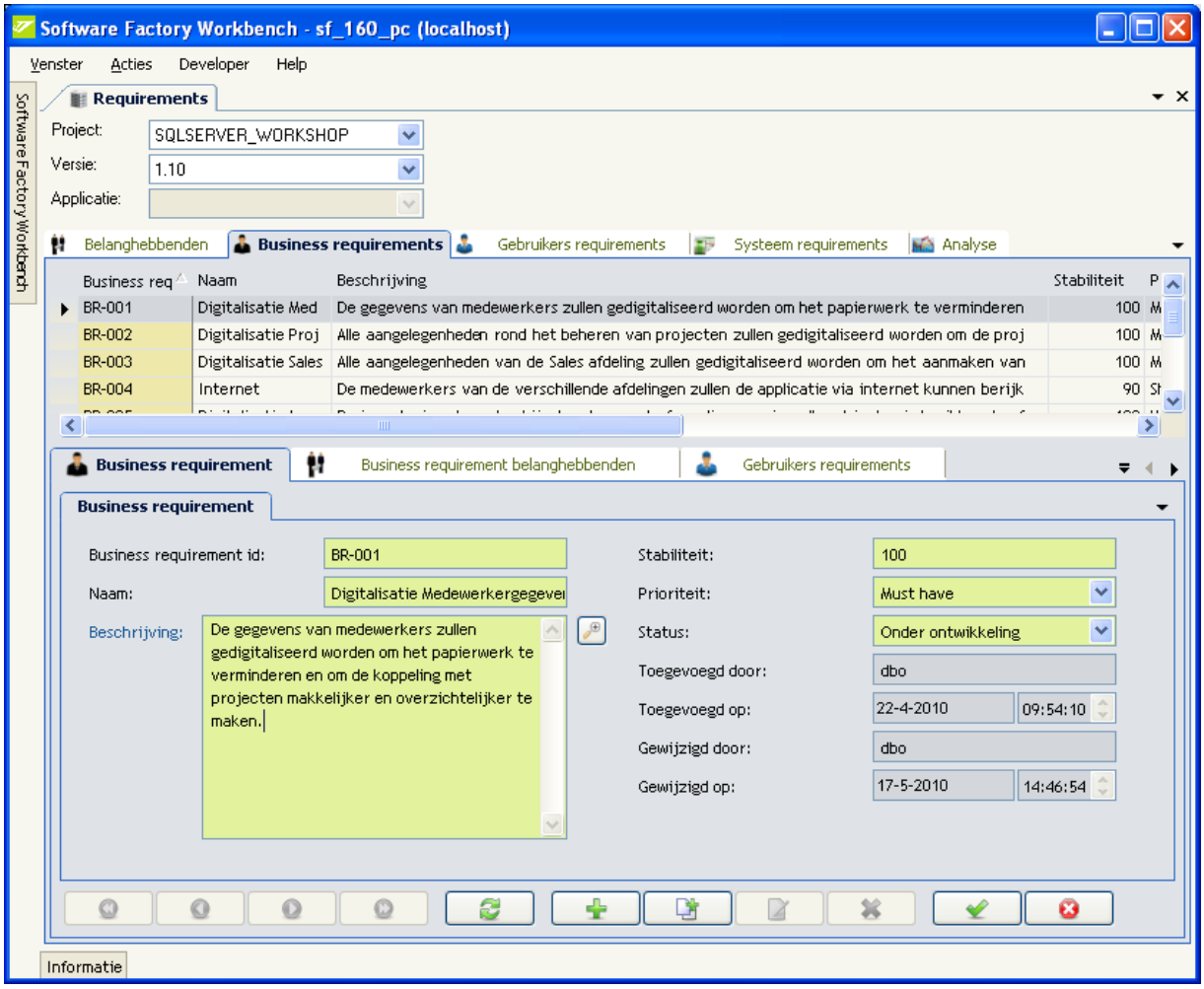
Figuur 25 - Het belanghebbenden tabblad van het requirements scherm

Het tweede tabblad is genaamd 'Businessrequirements'. Hier wordt voor de analist een master-detail geïnitialiseerd voor de businessrequirements. Door middel van een gemodelleerde *voorwaardelijke layout* worden kleurcodes uitgedeeld aan de verschillende records, afhankelijk van de status.

In dit master-detail scherm zijn 6 detailtabbladen beschikbaar gesteld.

- Het eerste detailtabblad bevat een formulier dat wordt gebruikt voor datamanipulatie. Onderstaand schermafdruk geeft hier een voorbeeld van.
- In het tweede detailtabblad worden belanghebbenden gekoppeld aan de geselecteerde businessrequirement.
- Het derde detailtabblad geeft een overzicht van alle gebruikersrequirements gekoppeld aan de geselecteerde businessrequirement.
- In het vierde detailtabblad kunnen bijlagen aan de geselecteerde businessrequirement worden gekoppeld.
- Het vijfde detailtabblad biedt de mogelijkheid om een tag te koppelen aan de geselecteerde businessrequirement.
- Het zesde detailtabblad biedt een historisch wijzigingsoverzicht van de geselecteerde businessrequirement.

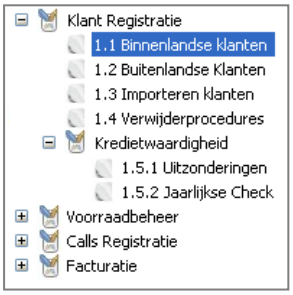
Hiernaast is in dit scherm een taak aanwezig om de status van de geselecteerde businessrequirements te wijzigen. Ook is een taak aanwezig om belanghebbenden te koppelen aan de geselecteerde businessrequirements.



Figuur 26 - Het businessrequirements master-detail in het requirements scherm

Het derde scherm, het 'Gebruikersrequirements' tabblad, geeft een overzicht van alle gebruikersrequirements. Voor het overzicht van de gebruikersrequirements wordt op het moment van schrijven een lijstcomponent gebruikt. Echter, een ontwerp voor een boomstructuur component is gemaakt, om de hiërarchie van gebruikersrequirements te visualiseren. Een ontwerpschets is opgenomen als bijlage (Bijlage 3).

In het gebruikersrequirement tabblad zijn 8 detailtabbladen beschikbaar.



Figuur 27 - Concept voor hiërarchische weergave van gebruikersrequirements

- Het eerste detailtabblad biedt een formulier dat wordt gebruikt voor datamanipulatie
- Het tweede detailtabblad biedt een overzicht van onderliggende gebruikersrequirements. Dit scherm is gelijk aan het gebruikersrequirement tabblad. Echter, alle gebruikersrequirement in dit detailtabblad staan in de context van de geselecteerde gebruikersrequirement.
- Het derde detailtabblad geeft een overzicht van belanghebbenden, en biedt de mogelijkheid om belanghebbenden te koppelen.
- Het vierde detailtabblad geeft een overzicht van de onderliggende systeemrequirements, biedt de mogelijkheid om deze aan te maken en biedt de mogelijkheid om datamanipulatie uit te voeren op de onderliggende systeemrequirements.'
- Het vijfde detailtabblad biedt de mogelijkheid om bijlagen te koppelen.
- Het zesde detailtabblad biedt een *Business Intelligence* component, waar doorsneden met betrekking tot implementatiegegevens getoond worden over de onderliggende systeemrequirements, en systeemrequirements van onderliggende gebruikersrequirements.
- Het zevende detailtabblad biedt de mogelijkheid om tags koppelen aan de geselecteerde gebruikersrequirement.
- Het achtste detailtabblad geeft een historisch overzicht van de geselecteerde gebruikersrequirement.

Net als bij het businessrequirements scherm is in dit scherm een taak aanwezig om de status van de geselecteerde businessrequirements te wijzigen. Ook is een taak aanwezig om belanghebbenden te koppelen aan de geselecteerde businessrequirements.

Figuur 28 - De taak voor het veranderen van de status van één of meerdere gebruikersrequirements

Het systeemrequirements tabblad biedt eveneens een master-detail scherm met alle systeemrequirements van de betreffende projectversie. Dit tabblad zal in de praktijk minder gebruikt worden, aangezien systeemrequirements meestal in de context van een gebruikersrequirement worden gedefinieerd, in het vorige tabblad. Dezelfde *voorwaardelijke opmaak* is toegepast als bij de businessrequirements, om snel de verschillende statussen te identificeren.

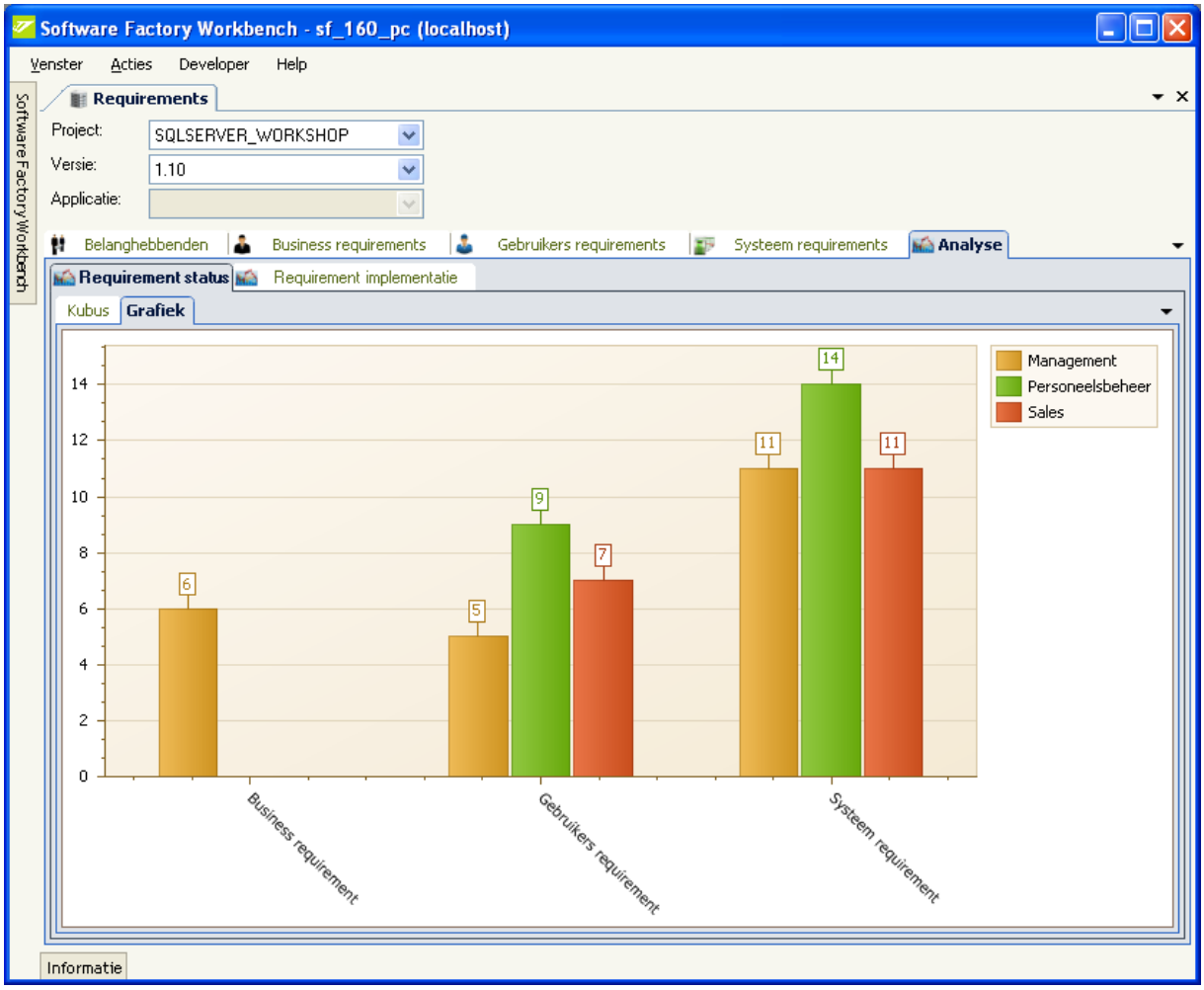
Systeemrequirements bieden 9 detailtabbladen.

- Het eerste detailtabblad biedt wederom een formulier voor het uitvoeren van datamanipulatie. In het formulier wordt de ontwerp inschatting in een apart tabblad getoond, aangezien deze inschatting waarschijnlijk door een andere actor zal worden gespecificeerd (de designer) dan de actor die de requirement specificeert (de analist).
- Het tweede detailtabblad biedt een scherm om belanghebbenden te koppelen aan de geselecteerde systeemrequirement.
- Het derde detailtabblad biedt de mogelijkheid om bijlagen te koppelen aan de geselecteerde systeemrequirement.
- Het vierde detailtabblad biedt de mogelijkheden om afhankelijkheden te definiëren van andere systeemrequirements.
- Het vijfde detailtabblad geeft een overzicht van systeemrequirements die afhankelijk zijn van de geselecteerde systeemrequirement.
- In het zesde detailtabblad is genaamd ‘Requirement implementatie’ en biedt een *Business Intelligence* component, waarin een grafisch overzicht wordt geboden van de modelobjecten die gekoppeld zijn aan de betreffende systeemrequirement.
- Het zevende detailtabblad geeft een overzicht van modelobject koppelingen in de vorm van een lijst component, waar nieuwe koppelingen snel aangemaakt kunnen worden en modelobjecten ontkoppeld kunnen worden.
- Het achtste detailtabblad biedt de mogelijkheid om tags koppelen aan de geselecteerde systeemrequirement.
- Het negende detailtabblad geeft een historisch overzicht van de geselecteerde gebruikersrequirement.

Hiernaast biedt ook het systeemrequirements scherm de twee taken, de eerste om de status van één of meerdere systeemrequirements aan te passen en de tweede om snel belanghebbenden toe te voegen aan één of meerdere systeemrequirements.

Het Analyse tabblad in het requirements scherm biedt twee *Business Intelligence* componenten. In dit scherm kunnen doorsneden in de requirements data worden gemaakt en worden grafische weergaven van deze doorsneden geboden.

Het eerste *Business Intelligence* component biedt de mogelijkheid om doorsneden in de requirements te maken op het gebied van de belanghebbenden, status en goedkeuringsstatus. Hiernaast kunnen andere attributen aan de doorsnede parameters worden toegevoegd, zoals prioriteit en type. Onderstaand schermafdruk geeft een voorbeeld van een doorsnede van de verschillende requirement typen van de verschillende belanghebbenden.

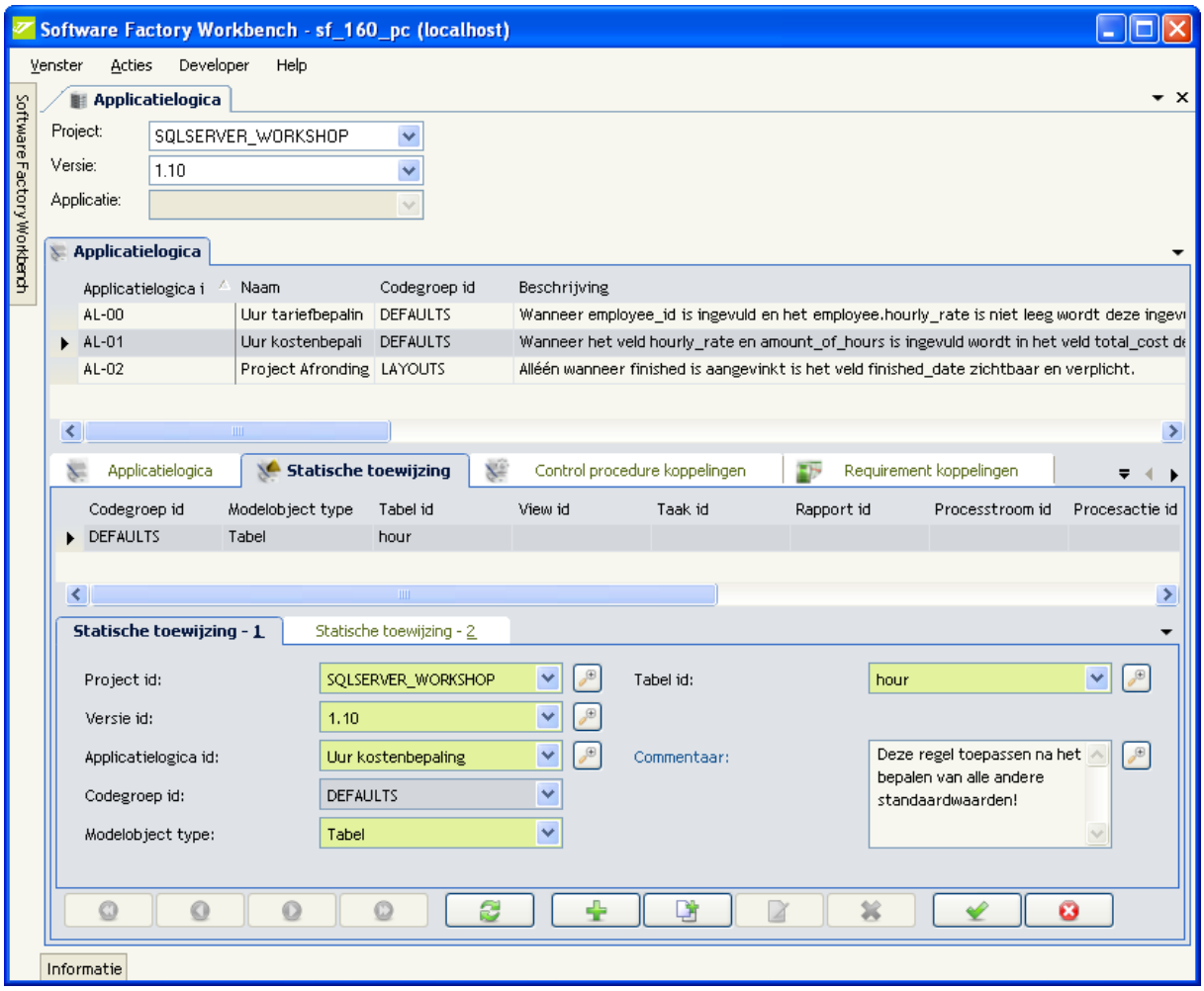


Figuur 29 - Het *Business Intelligence* component, geïnitialiseerd op requirements met betrekking tot de belanghebbenden, status en goedkeuring

6.3.3 *Applicatielogica Scherm*

Het applicatielogica scherm wordt gebruikt door de designers om de applicatielogica specificaties vast te leggen. Net als alle modelleerschermen is het applicatielogica scherm een processcherm, aangezien het project/projectversie selectiecomponent aanwezig moet zijn. Ten tijde van schrijven is in dit scherm één tabblad aanwezig, een master-detail scherm geïnitialiseerd voor applicatielogica specificaties. Een *Business Intelligence* component is nog niet ontworpen voor de applicatielogica specificaties. Voor applicatielogica specificaties zijn vijf detailtabbladen beschikbaar:

- In het eerste tabblad is een formulier component aanwezig om datamanipulatie uit te voeren op de applicatielogica specificaties.
- In het tweede tabblad kunnen statische toewijzingen worden gedefinieerd. Dit tabblad wordt dynamisch in- of uitgeschakeld aan de hand van het toewijzingstype van de geselecteerde applicatielogica specificatie, aangezien bij dynamische applicatielogica specificaties geen statische toewijzingen worden opgegeven.
- Het derde tabblad geeft een overzicht van *Control Procedures* die op de applicatielogica specificatie zijn gebaseerd; de *forward traceability*.
- Het vierde tabblad geef een overzicht van gekoppelde requirements aan de geselecteerde applicatielogica specificatie, de *backwards traceability*. Paragraaf 6.4.1 gaat dieper in op de techniek achter het automatisch koppelen van requirements aan modelobjecten.
- Het vijfde tabblad geeft een historisch wijzigingsoverzicht van de applicatielogica specificatie.



Figuur 30 - Het applicatielogica scherm, tijdens het definiëren van een statische toewijzing

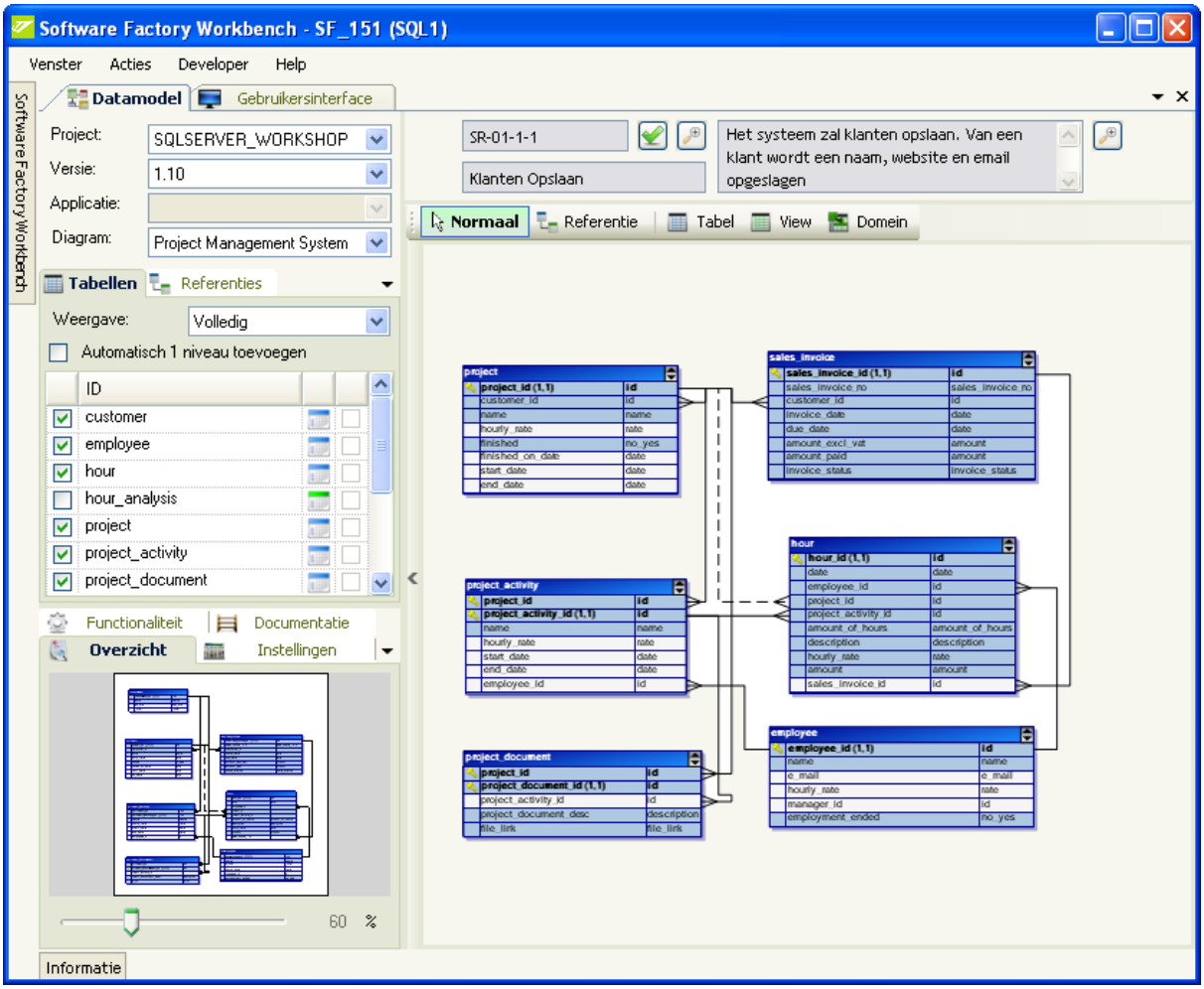
6.3.4 *Aanpassing modelleerschermen*

Om de traceerbaarheid tussen requirements expliciet te maken moeten gemodelleerde objecten aan systeemrequirements worden gekoppeld. Hierbij moet de designer echter niet in zijn werk gestoord worden; het koppelen van de systeemrequirements moet door de Thinkwise Software Factory worden uitgevoerd, niet door de gebruiker. Ten tijde van schrijven is de grafische uitbreiding nog niet gerealiseerd in de gewenste vorm, enkel een prototype is ontworpen, maar deze voldoet nog niet aan de eisen.

Om de designer te ondersteunen bij het modelleerproces op basis van requirements zijn de modelleerschermen uitgebreid met een requirements component.

Allereerst zal de designer een systeemrequirement te selecteren. Het selectiescherm is beschikbaar in een popup venster, die geopend wordt vanuit het modelleerscherm met een knop. Op de set met systeemrequirements in het modelleerscherm is een standaard filter aanwezig waarbij alleen systeemrequirements van het betreffende modeltype worden weergegeven (datamodel, GUI model, processtromen of applicatielogica) die nog niet zijn afgerond. Ook heeft de designer de mogelijkheid om *geen* requirement te selecteren.

Na het selecteren van een requirement wordt het ID, de naam en de omschrijving weergegeven in het requirements component. Alle acties die de designer uitvoert in het modelleerscherm resulteren vervolgens in koppelingen tussen modelobjecten en requirements. Wanneer de designer de modeltype specifieke aspecten van een requirement heeft gemodelleerd, zal hij de requirement afvinken. De requirement wordt op afgerond gezet en het selectiescherm wordt dan opnieuw getoond.



Figuur 31 - Ontwerp voor het requirement component in het data model scherm

6.4 Broncode

6.4.1 C# code

Alle modelleerschermen van de Thinkwise Software Factory zijn geschreven in C#. Hierbij is gebruik gemaakt van de designer van Microsoft Visual Studio. Aangezien vrijwel alle applicatielogica zich in de business laag (SQL) bevind, gaat het bij het schrijven van C# code grotendeels om het visualiseren van de data, waarbij gebruik gemaakt is van standaard componenten.

Echter, de code om modelobjecten te koppelen aan systeemrequirements is geïntegreerd in de modelleerschermen. Deze code kan niet worden uitgevoerd in de business laag, omdat daar de context niet bekend is (de geselecteerde systeemrequirement is niet bekend).

Voor het afvangen van de gebruikersacties is meerdere malen gebruik gemaakt van een *observer* design pattern.

Bij controllers[5] worden eventlisteners geregistreerd die alle deze eventlisteners registreren wanneer een childcontroller[6] wordt aangemaakt. Op deze manier worden de eventlisteners recursief geregistreerd. Onderstaande broncode laat deze eventlistener zien.

```
void controller_AfterSyncChildController(object sender, SyncChildControllerArgs e)
{
    if (e.FirstTime)
    {
        e.ChildController.AfterSyncChildController +=
               new EventHandler<SyncChildControllerArgs>(controller_AfterSyncChildController);
        e.ChildController.ActionPerformed +=
               new EventHandler<ActionPerformedArgs>(controller_ActionPerformed);
        e.ChildController.AfterRowInserted +=
               new EventHandler<RowUpdatedArgs>(controller_AfterRowInserted);
        e.ChildController.AfterRowUpdated +=
               new EventHandler<RowUpdatedArgs>(controller_AfterRowUpdated);
    }
}
```

Vervolgens worden bij alle controllers eventlisteners geregistreerd die naar de uitvoering van taken luisteren. Bepaalde taken worden gebruikt om modelobjecten te kopiëren, een requirement link moet worden gelegd met het nieuw aangemaakte object.

De meest belangrijke eventlisteners zijn de eventlisteners die naar de *insert row* en *update row* luisteren. Bij het toevoegen en wijzigen van een rij zal een implementatielink met de betreffende rij worden gelegd.

Hiernaast hebben de modelleerschermen de mogelijkheid om zelf aan te geven wanneer een implementatielink gelegd moet worden. Dit zal bijvoorbeeld worden toegepast wanneer een wijzig- of toevoegactie van een modelobject niet via een controller loopt maar rechtstreeks op de database connectie wordt uitgevoerd.

[5] Controller zijn (kort door de bocht) objecten die een dataset beheren en de visuele componenten die gebruik maken van deze dataset beheren.

[6] Childcontrollers zijn controllers op een gefilterde detail dataset van een andere controller. Bijvoorbeeld, een controller is geïnitialiseerd op *Projecten*, dan zou een childcontroller worden gebruikt voor de bijbehorende *Project Activiteiten*. Een childcontroller kan op zijn beurt ook zijn eigen childcontrollers hebben (*Project Activiteit Documenten)*.

6.4.2 SQL code

Naast de requirement implementatie koppelingen is de rest van de applicatielogica geschreven met behulp van de applicatielogica concepten van Thinkwise. Op het moment van schrijven zijn 21 *Control Procedures* toegevoegd met 508 regels code (dat na het weven resulteert in 1.289 regels code) voor de applicatielogica met betrekking tot requirement- en applicatielogica specificaties. Hiernaast zijn 6 *Control Procedures* toegevoegd voor de invulling van de verschillende views, met 4.116 regels code.

De broncode varieert van simpele *lay-out* procedures om velden alleen-lezen te maken wanneer een requirement onder review staat, tot complexe taken voor het verwerken van goedkeuringen en triggers om de data integer te houden. Hoogtepunt hierbij is het intact houden van de hiërarchische structuur van gebruikersrequirements in rapportages en business intelligence componenten.

Om dit aan te pakken is gebruik gemaakt van *Common Table Expressions* (CTE's). Een CTE heeft de mogelijkheid om op zijn bestaande set een *union all* uit te voeren met een set die is opgebouwd door middel van een *inner join* op zijn bestaande set.

Ter illustratie, onderstaande code laat een CTE zien die is gebruikt voor een *Business Intelligence* view voor het bepalen van gekoppelde modelobjecten aan systeemrequirements. Deze view geeft deze koppelingen weer in de context van een gebruikersrequirement. Hierbij moeten ook de koppelingen van onderliggende gebruikersrequirements worden meegenomen.

```
WITH tab_cte AS
(
   SELECT
      i.project_id,
      i.project_vrs_id,
      s.user_req_id,
      u.user_req_name,
      'tab' AS model_object_type,
      i.tab_id AS model_object_name,
      u.parent_user_req_id
   FROM req_impl i
   JOIN system_req s
      ON  i.project_id     = s.project_id
      AND i.project_vrs_id = s.project_vrs_id
      AND i.system_req_id  = s.system_req_id
   JOIN user_req u
      ON  i.project_id     = u.project_id
      AND i.project_vrs_id = u.project_vrs_id
      AND s.user_req_id    = u.user_req_id
   WHERE i.tab_id IS NOT NULL
   UNION ALL
   SELECT
      rr.project_id,
      rr.project_vrs_id,
      ur.user_req_id,
      ur.user_req_name,
      rr.model_object_type,
      rr.model_object_name,
      ur.parent_user_req_id
   FROM user_req ur
   INNER JOIN tab_cte rr
      ON  rr.project_id         = ur.project_id
      AND rr.project_vrs_id     = ur.project_vrs_id
      AND rr.parent_user_req_id = ur.user_req_id
)

SELECT DISTINCT project_id, project_vrs_id, user_req_id, user_req_name, model_object_type,
model_object_name
FROM tab_cte
```

Een *where clause* in de query waarbij een user_req_id wordt opgegeven zal ook de gekoppelde modelobjecten van onderliggende gebruikersrequirement weergeven. Naast bovenstaande CTE zijn ook andere CTE's voor andere modelobjecten meegenomen. De *union* van de resultaten wordt gebruikt voor het *Business Intelligence* component.

7 Conclusie

De requirements zijn succesvol geïntegreerd in de Thinkwise Software Factory. Hierbij voldoet de Thinkwise Software Factory aan de standaarden van requirements die door verschillende instanties en schrijvers zijn opgesteld.

De Thinkwise Software Factory biedt geautomatiseerde traceerbaarheid tussen de verschillende niveaus van requirements en de uiteindelijk gemodelleerde oplossing. Voor het projectmanagement worden verschillende manier beschikbaar gesteld om het overzicht op de requirements te bewaren, door bijvoorbeeld grafische doorsneden en rapportages.

De analisten worden in het specificeren van de requirements ondersteund door validaties die controleren op de structurele juistheid van de requirements. Hiernaast heeft de analist rapportages tot zijn beschikking zodat belanghebbenden de requirements kunnen beoordelen. Ook biedt de Thinkwise Software Factory verschillende manieren tot zijn beschikking om beoordelingen te verwerken en achterstand in beoordelingen te identificeren.

De designers hebben de mogelijkheid om de verschillende modellen van het systeem te ontwerpen op basis van de gespecificeerde requirements. Hierbij kan de lijst met requirements als een *to do* lijst worden gebruikt. Modelobjecten aangemaakt of gewijzigd in de context van een requirement worden automatisch aan de requirement gekoppeld, zodat er traceerbaarheid ontstaat tussen de requirements en de implementatie. Logica aspecten van het systeem worden door de designers niet langer in externe documenten vastgelegd, maar opgenomen in de Thinkwise Software Factory als applicatielogica specificaties.

Deze applicatielogica specificaties dienen als basis voor de developers om de logica aspecten van het systeem in broncode te realiseren en in het model te weven. Hierbij worden de developers op dezelfde manier ondersteund als de designers die op basis van de requirements werken, de developers kunnen de applicatielogica specificaties beschouwen als een *to do* lijst. De broncode wordt ook rechtstreeks aan de applicatielogica specificaties gekoppeld.

Door deze lagen van traceerbaarheid is uiteindelijk elk aspect van de eindapplicatie terug te brengen naar een wens van de gebruiker. Door deze gestructureerde manier van werken zal de kwaliteit van het eindproduct worden gegarandeerd, aangezien alle aspecten van het systeem op basis van de requirements zijn gerealiseerd.

Dit geeft een praktische invulling aan de ISO definitie van kwaliteit:

‘*The degree to which a set of inherent characteristics fulfills requirements*’.

8 Aanbevelingen

8.1 Requirements

Terminologie met betrekking tot niveaus, typeringen en groeperingen van requirements worden door bedrijven verschillend geïnterpreteerd of gedefinieerd. Een uniform requirements terminologie model wordt geboden door het boek *Succes met de requirements!*, maar dit is niet geaccepteerd als een (internationale) standaard en kan verschillend worden geïnterpreteerd.

Een mogelijkheid is om in de toekomst de niveaus, typeringen en groeperingen dynamisch te ondersteunen in de Thinkwise Software Factory en zo de analist de mogelijkheid geven om zelf het requirements model in te richten, met de verschillende niveaus, afhankelijkheden, traceerbaarheid opties en typespecifieke attributen.

Echter, belangrijke aspecten van requirements kunnen op deze manier worden uitgesloten en de rapportages zullen er niet overzichtelijker op worden. Hiernaast wordt het moeilijker om *outsourcing* toe te passen wanneer er geen voorgedefinieerde standaard wordt gebruikt.

8.2 Applicatielogica

Bij het gebruik van applicatielogica specificaties zou een deel van de control procedure kunnen worden gegenereerd zodat de developer niet bepaalde attributen handmatig moet overnemen uit de specificatie. De Thinkwise Software Factory kan de mogelijkheid bieden om bijvoorbeeld de codegroep en statische toewijzingen vanuit de specificatie over te nemen in de control procedure bij het aanmaken deze.

8.2.1 MDA

De platform modellen (*platform models)* zijn in de Thinkwise Software Factory niet expliciet gemaakt, maar opgenomen in *meta* en *sub* control procedures die tijdens een generatieslag de PSM modelobjecten aanmaken. Deze control procedures bevinden zich in basisprojecten. Deze basisprojecten worden gekoppeld bij het maken van een eindproduct en bepalen dus de beschikbare platform modellen.

Door de definitie van de platform modellen niet in code te specificeren maar expliciet op te nemen als data, is het makkelijker om platform modellen te toe voegen en te wijzigen. Hiernaast wordt het eenvoudiger om een nieuw logicaconcept te introduceren en de platform modellen te voorzien van de benodigde informatie om dit logica concept te ondersteunen.

Vooral met de opkomst van verschillende middle-tier platformen zou een expliciete definitie van de platform modellen de integratie van deze versnellen. Hiernaast kan bij het specificeren van applicatielogica rekening worden gehouden met de architectuur van de applicatie als deze expliciet wordt gedefinieerd door het bepalen van de te gebruik platform modellen. Het is namelijk niet ondenkbaar dat bepaalde platformen compleet verschillende applicatielogica concepten zullen ondersteunen.

8.3 Thinkwise Software Factory

Tijdens het onderzoek en implementatietraject zijn verschillende punten naar voren gekomen waar de Thinkwise Software Factory verbeterd kan worden.

8.3.1 Voorkomen foutmeldingen m.b.t. model integriteit

De Thinkwise Software Factory maakt veel gebruik van semantische *primary keys*, modelobjecten maken vaak niet gebruik van een *identity primary key* (1, 2, 3) maar van een woord ('object_1'). Een semantische primary key kan dan ook handmatig gedefinieerd of gewijzigd worden in de Thinkwise Software Factory.

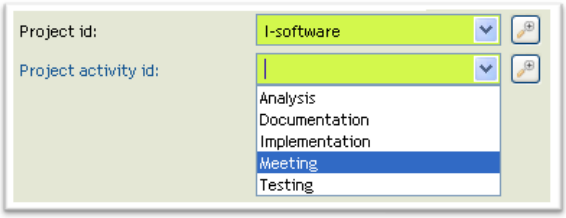
Dit is op zich geen probleem, totdat de ontwerper de primary key van een modelobject probeert te wijzigen wanneer er vanuit een andere tabel naar het modelobject wordt verwezen. Een integriteitsfout zal optreden. Deze integriteitsfouten doen zich ook voor wanneer een ontwerper een modelobject probeert te verwijderen waarnaar verwezen wordt.

De Thinkwise Software Factory biedt taken aan om de *primary key* van een modelobject te wijzigen of een modelobject te verwijderen, inclusief de *foreign key* referenties.

De Thinkwise Software Factory kan zonder al te vele moeite worden aangepast zodat het bewerken van de *primary key(s*) wordt uitgeschakeld wanneer naar een modelobject wordt verwezen. Hetzelfde geld voor het uitschakelen van de mogelijkheid om een record te verwijderen. Wanneer dit het geval is weet de ontwerper dat de taken gebruikt moeten worden voor het wijzigen van de *primary key* of het verwijderen van het modelobject, zonder dat eerst een foutmelding moet optreden.

8.3.2 Gecombineerde foreign keys

In eindproducten van de Thinkwise Software Factory worden vaak gecombineerde *primary key*s gebruikt waarin een *foreign key* is opgenomen. Denk hierbij bijvoorbeeld aan een projectactiviteit record waarin ook het project_id van het bovenliggende project is opgenomen. Wanneer vervolgens een record in een andere tabel naar de projectactiviteit moet verwijzen, wordt op volgorde van de primary key de verschillende sleutelwaarden gekozen. Eerst wordt het project gekozen, vervolgens de bijbehorende project activiteit.



Figuur 32 – Een verwijzing naar een projectactiviteit. Hierbij is gebruik gemaakt van een gecombineerde sleutel.

Deze manier van werken dekt het overgrote deel van de gevallen af waarbij wordt verwezen naar een gecombineerde primary key die deels bestaat uit foreign keys.

Dit is echter niet altijd gewenst. Bij matrixtabellen, tabellen die verwijzen naar meerdere andere tabellen om zo bijvoorbeeld beschikbare combinaties te beschrijven, is het gewenst om in een andere volgorde te kunnen werken, bijvoorbeeld eerst de tweede sleutel, dan de eerste. (Eerst een activiteit selecteren, dan een project dat deze activiteit bevat). Dit wordt niet ondersteund.

8.3.3 Toekomstige Development Process Control uitbreidingen

De kwaliteit van eindproducten van de Thinkwise Software Factory is verbeterd door de requirements en de applicatielogica te integreren in de Thinkwise Software Factory. Echter, naast dit onderwerp zijn er meer onderwerpen geïdentificeerd waar de Thinkwise Software Factory ondersteuning kan bieden om de kwaliteit van eindproducten te verbeteren. Hoewel deze onderwerpen buiten de scope van dit project zijn gevallen, is het aan te raden om in de toekomst projecten op te starten om ook deze *procescontrole uitbreidingen* te ontwerpen en te realiseren

- Broncode versiebeheer en meta-informatie
- Broncode review- en controlemogelijkheden
- Impactanalyses
- Geweven broncode testmogelijkheden

Vooral op het gebied van testen kan veel worden gewonnen. De tests kunnen worden gekoppeld aan de nu beschikbare requirements, om zo de cirkel compleet te maken.

Voor broncode review- en controlemogelijkheden kan gebruik worden gemaakt van de applicatielogica specificaties die nu zijn opgenomen in de Thinkwise Software Factory. Deze specificaties bieden een duidelijke basis om de broncode op te controleren.

Hiernaast heeft de implementatie van de requirements deuren geopend voor het integreren van projectmanagement onderdelen in de Thinkwise Software Factory. Op basis van de requirements kunnen planningen worden gemaakt, kan de voortgang in de gaten worden gehouden en kunnen zelfs urenregistraties van ontwikkelaars worden geïntegreerd. Dit onderdeel is zeker een onderzoek waard.

8.4 PRINCE2 en Thinkwise Software

Bij het project is succesvol gebruik gemaakt van de PRINCE2 projectmanagement methode. Thinkwise Software heeft verschillende PRINCE2 rapportages als templates beschikbaar gesteld. Het administratiesysteem van Thinkwise Software, TBS, is echter niet volledig op PRINCE2 ingericht. Fasen kunnen nog niet gedefinieerd worden, terwijl dit erg belangrijk is voor PRINCE2 en deliverables worden op het moment gedefinieerd als subprojecten.

TBS structuur:	Projecten – Subprojecten – Activiteiten
PRINCE2 structuur:	Projecten – Fasen – Deliverables – Activiteiten

Wanneer er wordt besloten om bij alle projecten PRINCE2 projectmanagement toe te passen is het aan te raden om de structuur van het administratiesysteem aan te passen. Hiernaast kan de optie worden geboden om stuurgroepen te definiëren in het administratiesysteem en zo deze automatisch op te hoogte houden wanneer rapportages van een bepaald type worden gekoppeld aan een project.

9 Bibliografie

1. **ISO 9000.** *Quality management systems -- Fundamentals and vocabulary.* 2000.

2. **IEEE Std 830.** *IEEE Recommended Practice for Software Requirements Specifications.* 1998.

3. **Arendsen, Marten, et al.** *Succes met de requirements!* 2008.

4. **Bredemeyer, Dana en Malan, Ruth.** *Functional Requirements and Use Cases.* 2001.

5. **ISO/IEC 9126-1.** *Software engineering -- Product quality -- Part 1: Quality model.* 2001.

6. **IEEE Std 1233a.** *IEEE Guide for Developing System Requirements Specifications.* 1998.

7. **Ash, Steve.** DSDM Consortium - Knowledgebase: MoSCoW Prioritisation Briefing Paper. [Online] 2007. http://www.dsdm.org/knowledgebase/details/165/moscow-prioritisation-briefing-paper.html.

8. **Object Management Group.** *MDA Guide v1.0.1.* 2003.

9. **The Business Rules Group.** *Defining business rules – what are they really?* [http://www.businessrulesgroup.org/first_paper/BRG-whatisBR_3ed.pdf] juli 2000.

10 Verklarende woordenlijst

Applicatielogica – Bedrijfsspecifieke regels die veelal niet door standaard data beperkingen worden afgedwongen.

AOP – Zie *Aspectgeoriënteerd programmeren.*

Aspectgeoriënteerd programmeren – Een programmeerparadigma waarbij logische delen broncode op voorgedefinieerde locaties van in de applicatie worden toegewezen.

Belanghebbende – Een persoon of groep van personen bij een applicatie betrokken zijn.

Business Intelligence – Een component dat wordt gebruikt door de Thinkwise GUI dat doorsneden kan maken op data en deze doorsneden kan visualiseren door middel van grafieken.

CIM – Zie *Computation Independent Model*.

Computation Independent Model – Een digitaliseringsonafhankelijk model van een systeem.

Control procedure – Een modelobject in de Thinkwise Software Factory. Een control procedure is een drager van (delen) broncode en toewijzingen van broncode die applicatielogica vertegenwoordigen.

DML – Datamanipulatie. Het toevoegen, wijzigen en verwijderen van records uit een tabel of een andere gegevensdrager.

Elicitatie – Het proces dat een analist uitvoert om te begrijpen wat er speelt in een kennisgebied, een werkdomein.

GUI – Grafische gebruikersinterface. Deze aanduiding wordt gebruikt wanneer iets betrekking heeft op de weergave naar eindgebruikers toe.

Intelligent Application Manager –De IAM wordt na het ontwikkeltraject met de Software Factory gebruikt om autorisatie en gebruikersvoorkeuren in te richten van Meta Solution Definition modellen, en verschillende MSD's te combineren in één GUI.

MDA – Zie *Model Driven Architecture*.

Model Driven Architecture – De modelgedreven architectuur van de Object Management Group, een architectuur-paradigma waarbij gebruik wordt gemaakt van modellen en transformaties van modellen.

Middle-tier – De middelste laag van de n-tier architectuur, waar de applicatielogica zich bevind. Ook wel business tier, logic tier of data acces tier genoemd.

MSD – Zie *Meta Solution Definition*.

Meta Solution Definition – Deze elektronische bouwtekening wordt met behulp van de Thinkwise Software Factory gespecificeerd tijdens het ontwikkelproces en beschrijft alle functionele aspecten van een eindapplicatie van de Thinkwise Software Factory.

PIM – Zie *Platform Independent Model*.

Platform Independent Model – Een platformonafhankelijk model van een systeem.

Platform Model – Platform modellen worden ingezet voor de transformatie tussen verschillende viewpoints van een modelgedreven architectuur. Ter illustratie, de Thinkwise Software Factory bevat platform modellen voor SQL-Server, Oracle DB, DB2 om een vertaalslag te maken tussen gegevensentiteiten van het PIM en het PSM.

PSM – Zie *Platform Specific Model*.

Platform Specific Model – Een platform-specifiek model van een systeem.

Requirement – Een eis die aan een systeem wordt gesteld, met mogelijk meerdere condities of beperkingen.

Service bus – Een service bus is een architecturaal patroon waarbij communicatie aanbieders en afnemers van diensten vereenvoudigd wordt.

Stakeholder – Zie *Belanghebbende*.

Processcherm – Een aangepast scherm dat door de Thinkwise GUI wordt ingeladen. Processchermen worden ingezet wanneer GUI functionaliteit vereist is die niet in de MSD specificeerbaar is.

Bijlage 1

Voorbeeld rapportage Requirements

Klantnaam	:	Thinkwise Software	**Requirements overzicht**
Project	:	Workshop	
Project versie	:	1.10	
Datum	:	31-5-2010	
Belanghebbende	:	Management	

Selectiecriteria

Type requirement:		**Status:**	
Business requirements	√	In ontwikkeling	√
Gebruikers requirements	√	Onder review	√
Maximaal niveau	5	Volledig goedgekeurd	√
Systeem requirements	√		

Prioriteit:		**Belanghebbende beoordeling:**	
Must have	√	Geen	√
Should have	√	Afgekeurd	√
Could have	√	Goedgekeurd	√
Will not have	√		

1. Business Requirements

BR-001 - Digitalisatie Medewerkergegevens

De gegevens van medewerkers zullen gedigitaliseerd worden om het papierwerk te verminderen en om de koppeling met projecten makkelijker en overzichtelijker te maken.

Status	:	Onder review	Stabiliteit	:	100	Prioriteit	: Must have
Management beoordeling	:	Goedgekeurd					

BR-002 - Digitalisatie Projectgegevens

Alle aangelegenheden rond het beheren van projecten zullen gedigitaliseerd worden om de projectarchieven te centraliseren

Status	:	Onder review	Stabiliteit	:	100	Prioriteit	: Must have
Management beoordeling	:	Goedgekeurd					

BR-003 - Digitalisatie Sales

Alle aangelegenheden van de Sales afdeling zullen gedigitaliseerd worden om het aanmaken van facturen en factuuroverzichten te vergemakkelijken.

Status	:	Onder review	Stabiliteit	:	100	Prioriteit	: Must have
Management beoordeling	:	Goedgekeurd					

BR-004 - Internet

De medewerkers van de verschillende afdelingen zullen de applicatie via internet kunnen berijken zodat thuiswerk vergemakkelijkt wordt

Status	:	Onder review	Stabiliteit	:	90	Prioriteit	: Should have
Management beoordeling	:	Goedgekeurd					

BR-005 - Digitalisatie Inventaris

De inventaris zal worden bijgehouden, om te formaliseren wie welke objecten in bruikleen heeft.

Status	:	Onder review	Stabiliteit	:	100	Prioriteit	: Will not have
Management beoordeling	:	Goedgekeurd					

BR-006 - Backup Mogelijkheden

Alle data moet dagelijks een backup krijgen, om zo de historie van alle gegevens te behouden

Status : Onder review | Stabiliteit : 100 | Prioriteit : Could have

Management beoordeling : Goedgekeurd

2. Gebruikers- en Systeem Requirements

1.2.3 UR-01-2-3 - Historisch Overzicht

UR-01 Klant Beheer → UR-01-2 Facturatie → UR-01-2-3 Historisch Overzicht

De Sales afdeling heeft toegang tot een historisch overzicht van alle facturen

Status : Onder review | Stabiliteit : 100 | Prioriteit : Must have

Management beoordeling : Goedgekeurd

2.1 UR-02-1 - Medewerkers Registratie

UR-02 Medewerkers Beheer → UR-02-1 Medewerkers Registratie

Systeem Requirements

SR-02-1-8 - Managers

Het systeem zal voor een medewerker een andere medewerker als manager kunnen registreren

Type : Niet-functioneel | Status : In ontwikkeling

Stabiliteit : 100 | Prioriteit : Must have

Management beoordeling : Onbeoordeeld

2.3 UR-02-3 - Uren Beheer

UR-02 Medewerkers Beheer → UR-02-3 Uren Beheer

De Personeelsbeheer afdeling zal voor medewerkers de gewerkte uren beheren

Status : Onder review | Stabiliteit : 100 | Prioriteit : Must have

Management beoordeling : Goedgekeurd

3 UR-03 - Project Beheer

De projectleiders zullen projectgegevens beheren.

Status : Volledig goedgekeurd | Stabiliteit : 100 | Prioriteit : Must have

Management beoordeling : Goedgekeurd

3.1 UR-03-1 - Project Registratie

UR-03 Project Beheer → UR-03-1 Project Registratie

Het management zal nieuwe projecten registreren voor klanten

Status : Onder review | Stabiliteit : 100 | Prioriteit : Must have

Management beoordeling : Goedgekeurd

Systeem Requirements

SR-03-1-1 - Project Registratie

Het systeem zal projecten opslaan. Deze hebben een naam, een klant en een start- en einddatum.

Type : Niet-functioneel | Status : In ontwikkeling

Stabiliteit : 100 | Prioriteit : Must have

Management beoordeling : Onbeoordeeld

SR-03-1-2 - Project Uurtarief

Het systeem kan een uurtarief opslaan voor een project

Type : Niet-functioneel | Status : In ontwikkeling

Stabiliteit : 100 | Prioriteit : Must have

Management beoordeling : Onbeoordeeld

3.2 UR-03-2 - Project Onderhoud

UR-03 Project Beheer → UR-03-2 Project Onderhoud

Systeem Requirements

SR-03-2-1 - Wijzigingen

Het systeem zal wijzigingen aan projecten toestaan.

Type : Niet-functioneel | Status : In ontwikkeling

Stabiliteit : 100 | Prioriteit : Must have

Management beoordeling : Onbeoordeeld

SR-03-2-2 - Project Afronding

In het systeem kunnen projecten kunnen als afgerond worden gemarkeerd. Wanneer dit gebeurt, zal de einddatum afrondingsdatum worden geregistreerd.

Type : Niet-functioneel | Status : In ontwikkeling

Stabiliteit : 90 | Prioriteit : Must have

Management beoordeling : Onbeoordeeld

3.3 UR-03-3 - Project Activiteit Registratie

UR-03 Project Beheer → UR-03-3 Project Activiteit Registratie

Projectleiders zullen nieuwe projectactiviteiten opstellen voor projecten

Status	:	Onder review	Stabiliteit	:	100	Prioriteit	:	Must have

Management beoordeling : Goedgekeurd

Systeem Requirements

SR-03-3-1 - Project Activiteiten

Het systeem zal projectactiviteiten opslaan. Project activiteiten zullen per project geregistreerd kunnen worden. Een project activiteit zal een naam hebben, een startdatum en een geplande einddatum.

Type	:	Niet-functioneel	Status	:	In ontwikkeling
Stabiliteit	:	100	Prioriteit	:	Must have

Management beoordeling : Onbeoordeeld

SR-03-3-2 - Project Activiteit Uurtarief

Het uurtarief van projectactiviteiten kan worden opgeslagen door het systeem.

Type	:	Niet-functioneel	Status	:	In ontwikkeling
Stabiliteit	:	100	Prioriteit	:	Must have

Management beoordeling : Onbeoordeeld

3.4 UR-03-4 - Project Archief

UR-03 Project Beheer → UR-03-4 Project Archief

Systeem Requirements

SR-03-4-1 - Project Document Koppeling

Het systeem zal de mogelijkheid bieden om documenen te registreren. Deze documenten bevatten een link naar een bestand, een naam en een omschrijving. Hiernaast moet een document aan een project gekoppeld worden

Type	:	Niet-functioneel	Status	:	In ontwikkeling
Stabiliteit	:	100	Prioriteit	:	Must have

Management beoordeling : Onbeoordeeld

SR-03-4-2 - Project Activiteit Koppeling

Het systeem zal de mogelijkheid bieden om een document van een project aan een specifieke activiteit te koppelen

Type	:	Niet-functioneel	Status	:	In ontwikkeling
Stabiliteit	:	100	Prioriteit	:	Should have

Management beoordeling : Onbeoordeeld

3.5 UR-03-5 - HR Toewijzing

UR-03 Project Beheer → UR-03-5 HR Toewijzing

Systeem Requirements

SR-03-5-1 - Koppeling Activiteit - Medewerker

Het systeem zal de mogelijkheid bieden om een medewerker aan een project activiteit te koppelen.

Type	:	Niet-functioneel	Status	:	In ontwikkeling
Stabiliteit	:	100	Prioriteit	:	Must have

Management beoordeling : Onbeoordeeld

SR-03-5-2 - Dienst status voor HR Toewijzing

Het systeem zal alleen medewerkers kunnen koppelen die in dienst zijn.

Type	:	Niet-functioneel	Status	:	In ontwikkeling
Stabiliteit	:	100	Prioriteit	:	Must have

Management beoordeling : Onbeoordeeld

Bijlage 2

Exception Report #1 – 30 maart 2010

Exception Report #1 – Process Control in Model Driven Software Development – 30 maart 2010

Dit document beschrijft de oorzaak en gevolgen van een scope verandering die de planning dusdanig beïnvloed dat het niet door middel van een *Change Request* kan worden afgehandeld.

Het Exception Report is opgesteld op 30 maart 2010 door Anne Buit en is onderdeel van het project Software Development Process Control.

Omvang Afwijking

De scope van het deelproject *Applicatielogica Specificaties* is dusdanig veranderd dat de initiële projectplanning niet langer toerijkend is.

Initiële deliverables:

- Applicatielogica specificaties in de SF
- Koppeling Applicatielogica specificaties – Control procedures/templates

Gewenste deliverables:

- Requirements in de SF
- Koppeling Requirements – Modelobjecten
- Koppeling Requirements – Applicatielogica specificaties
- Applicatielogica specificaties in de SF
- Koppeling Applicatielogica specificaties – Control procedures/templates

Oorzaak afwijking

Uit onderzoek is gebleken dat het opnemen van de requirements in de Thinkwise Software Factory een krachtige uitbreiding zal zijn. Requirements vormen een solide basis voor het modelleren en specificeren van de applicatielogica specificaties. Op het moment biedt de Thinkwise Software Factory hier geen mogelijkheden voor, naast enkele MS Word templates.

Deze scope wijziging is vanuit de eigenaar van de business case, Victor Klaren, naar voren gekomen. Veel onderzoek op het gebied van requirements is binnen het project reeds verricht.

De deelvraag *'Hoe kan de brug tussen functionele specificaties en de Thinkwise Software Factory worden geslagen?'* zal beter worden beantwoord door het implementeren van de requirements. Het project wijkt in eerste instantie niet af van het opgestelde project initiatie document met betrekking tot de business case.

Consequenties afwijking

Het implementeren van de requirements in de Thinkwise Software Factory vergt veel onderzoek, aangezien er veel literatuur is op het gebied van requirements en de attributen en werkwijzen die hierbij komen kijken. Een apart ontwerp moet worden opgesteld voor de Requirements, naast het ontwerp voor de Applicatielogica Specificaties.

Door de scope verandering zal de naam van het deelproject moeten worden omgedoopt in *Requirements en Applicatielogica Specificaties*. De Applicatielogica Specificaties uitbreiding is in zekere zin gebaseerd op de nieuwe Requirements uitbreiding en kan niet volledig worden ontworpen tot het Requirements ontwerp is afgerond. De extra benodigde werkuren is geschat rond de 200 uur.

Alternatieven en effecten

Door de tijdsgebonden aard van het afstudeertraject waarin dit project plaatsvindt, zullen twee andere deelprojecten komen te vervallen. Hierbij zal de stuurgroep de definitieve keuze moeten maken.

Twee van de volgende deelprojecten zullen komen te vervallen binnen de scope van het afstudeertraject:

Ontwikkelfase 2 – Broncode Uitbreidingen

De deliverables van de ontwikkelfase voor broncode uitbreidingen dienen antwoord te geven op de deelvraag *'Op welke manier kan versiebeheer en meta-informatie behoud worden toegepast op broncode binnen de Thinkwise Software Factory?'*.

Ontwikkelfase 3 – Geweven Applicatielogica Uitbreidingen

De deliverables van deze fase dienen antwoord te geven op de deelvraag *'Welke mogelijkheden kan de Thinkwise Software Factory bieden met betrekking tot controle en review van geweven applicatielogica?'*.

Ontwikkelfase 4 – Impactanalyse en Ondersteunende Integriteit

De deliverables van de vierde ontwikkelfase dienen antwoord te geven op de deelvraag *'Hoe kan de Thinkwise Software Factory de ontwikkelaar op de hoogte worden gesteld van de impact van modelwijzigingen?'*.

Ontwikkelfase 5 – Gestructureerd Testen van Applicatielogica

De deliverables van de vijfde ontwikkelfase moeten antwoord te geven op de deelvraag *'Hoe kan het gestructureerd testen van applicatielogica worden gerealiseerd?'*. Deze fase kan naast programmatuur ook documentatie in de vorm van een handleiding opleveren om de ontwikkelaar instructies te geven met betrekking tot het testen.

Anticiperend op de business case is een concept planning bijgevoegd waarbij ontwikkelfase 4 en 5 komen te vervallen. Hierbij is de vrijgekomen tijd ter beschikking gesteld aan de eerste ontwikkelfase.

Huidige planning

Start fase
Start ..... 1 februari
Eind ..... 5 februari
Geplande tijd ..... 36 uren

Initiatie fase
Start ..... 8 februari
Eind ..... 26 februari
Geplande tijd ..... 144 uren

Ontwikkelfase 1– Applicatielogica Specificaties
Start ..... 1 maart
Eind ..... 19 maart
Geplande tijd ..... 108 uren

Ontwikkelfase 2 – Template Uitbreidingen
Start ..... 22 maart
Eind ..... 9 april
Geplande tijd ..... 104 uren

Ontwikkelfase 3 – Geweven Applicatielogica Uitbreidingen
Start ..... 12 april
Eind ..... 7 mei
Geplande tijd ..... 100 uren

Ontwikkelfase 4 – Impactanalyse en ondersteunende integriteit
Start ..... 10 mei
Eind ..... 4 juni
Geplande tijd ..... 100 uren

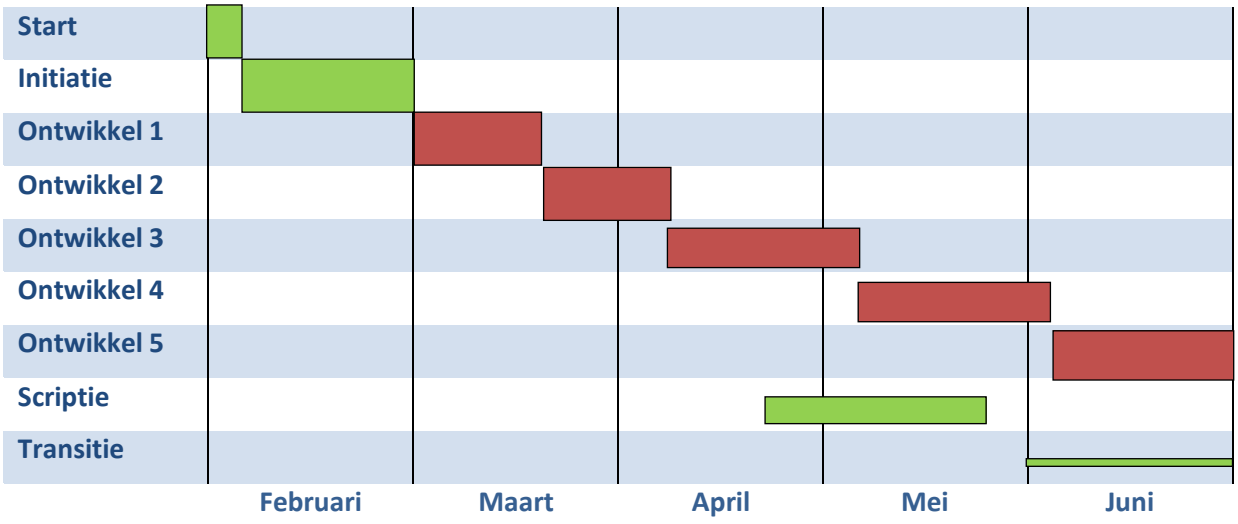
Ontwikkelfase 5 – Applicatielogica Testmogelijkheden
Start ..... 7 juni
Eind ..... 30 juni
Geplande tijd ..... 108 uren

Scriptie fase[7]
Start: ..... 20 april
Eind: ..... 25 mei
Geplande tijd: ..... 88 uren

Transitie fase[8]
Start: ..... 1 juni
Eind: ..... 30 juni

Geplande tijd: ..... 24 uren

Grafische weergave huidige planning



[7] Naast de geplande scriptie uren zal door het hele project heen aan de scriptie worden gewerkt. Dit zal vallen onder de activiteit documentatie van de desbetreffende deliverable.
[8] Afrondingsfase, presentatie en overdracht. Geen deliverables of managementproducten.

Planning bij het vervallen van ontwikkelfase 4 en 5

Start fase
Start ..... 1 februari
Eind ..... 5 februari
Geplande tijd ..... 36 uren

Initiatie fase
Start ..... 8 februari
Eind ..... 26 februari
Geplande tijd ..... 144 uren

Ontwikkelfase 1– Requirements & Applicatielogica Specificaties
Start ..... 1 maart
Eind ..... 7 mei
Geplande tijd ..... 108 uren

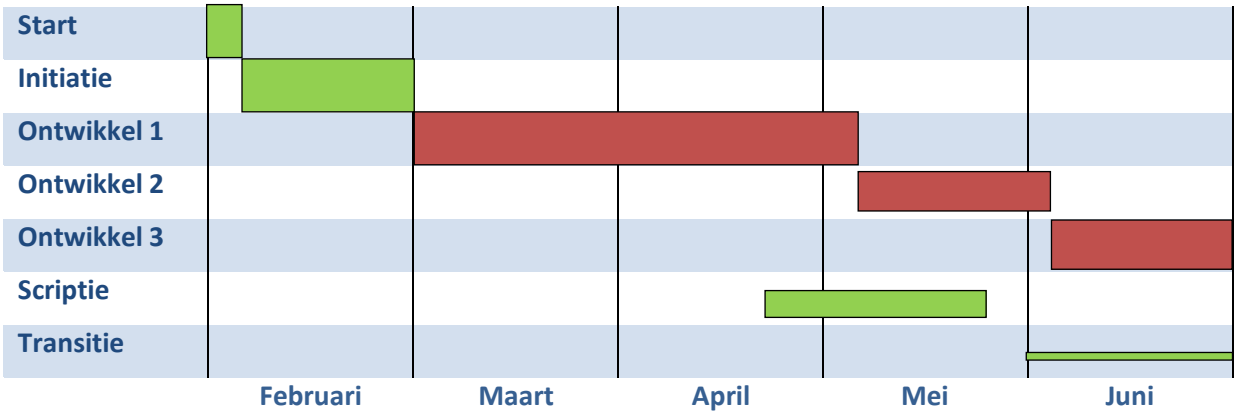
Ontwikkelfase 2 – Template Uitbreidingen
Start ..... 10 mei
Eind ..... 4 juni
Geplande tijd ..... 100 uren

Ontwikkelfase 3 – Geweven Applicatielogica Uitbreidingen
Start ..... 7 juni
Eind ..... 30 juni
Geplande tijd ..... 108 uren

Scriptie fase
Start: ..... 20 april
Eind: ..... 25 mei
Geplande tijd: ..... 88 uren

Transitie fase
Start: ..... 1 juni
Eind: ..... 30 juni
Geplande tijd: ..... 24 uren

Grafische planning bij het vervallen van ontwikkelfase 4 en 5



Bijlage 3

Generiek Boomstructuur Component – Ontwerpschets

Generiek Boomstructuur Component – Ontwerpschets

Initialisatie op één tabel. Een scherm biedt detailtabellen van geselecteerde record node in tabpanel en een formulier voor DML op geselecteerde record node. Presentatieveld van tabel gebruiken voor record node tekst (presentatie icoon introduceren? Conditioneel presentatie icoon zou mooi zijn!):

```
Tabel N
        + Record
        + Record
        + Record
        + Record
```

Groeperen op record attribuut (Scherm moet detailtabellen en formulier uitschakelen wanneer een groepeer attribuut waarde wordt geselecteerd. Mogelijk sommaties weergeven bij groepeer attribuut selectie?). Mogelijkheid om groepen te sorteren (asc/desc).

```
Tabel N
        - Groepeer Attribuut X Waarde A
                + Record
                + Record
                + Record
        - Groepeer Attribuut X Waarde B
                + Record
                + Record
        + Groepeer Attribuut X Waarde C
```

Groeperen op meerdere record attributen:

```
Tabel N
        + Record (X attribuut is leeg en Y attribuut is leeg bij dit record)
        + Record
        + Record
        + Record
        - (Groep) Attribuut X Waarde A
                + Record (Y attribuut is leeg bij dit record)
                + Record
                - (Groep) Attribuut Y Waarde A
                        + Record
                        + Record
                - (Groep) Attribuut Y Waarde B
                        + Record
        - (Groep) Attribuut X Waarde B
                - (Groep) Attribuut Y Waarde A
                        + Record
                        + Record
        - (Groep) Attribuut X Waarde C
                + Record
                - (Groep) Attribuut Y Waarde B
                        + Record
        - (Groep) Attribuut Y Waarde A
                + Record (X attribuut is leeg bij dit record)
                + Record
        - (Groep) Attribuut Y Waarde B
                + Record
                + Record
```

Groeperen op attribuut dat een foreign key (FK) is, waarbij de FK tabel een recursieve tabel is (dat mag ook dezelfde tabel zijn als degene waarop het boomstructuur component is geïnitialiseerd!). De groepering op het attribuut fk_tabel_id is dan intern hiërarchisch gegroepeerd op het fk_tabel_parent_id van de fk_tabel records. Hoe dit in een groepeerscherm wordt aangegeven moet worden onderzocht. Een hiërarchische FK groepering mag ook geen eindeloze *loop* opleveren.

```
Tabel N
        + Record
        + Record
        - (Groep) FK-Attribuut F Waarde 1 (Groepen op dit niveau hebben fk_tabel_parent_id = null)
                + Record
                + Record
                - (Groep) FK-Attribuut F Waarde 1.1 (1.1 is een child van 1, fk_tabel_parent_id = 1)
                        + Record
                        + Record
                - (Groep) FK-Attribuut F Waarde 1.2
                        + Record
        - (Groep) FK-Attribuut F Waarde 2
                - (Groep) FK-Attribuut F Waarde 2.1 (2.1 is een child van 2, fk_tabel_parent_id = 2)
                        + Record
                        + Record
        - (Groep) FK-Attribuut F Waarde 3
                (Groep) Record
```