Master Thesis Informatiekunde

Onderhoud van logica door business stakeholders

Radboud Universiteit Nijmegen

Everest B.V.

Auteur: Paul Verhoeven

Studentnummer: 0641685

Afstudeernummer: 172 IK

Datum: Augustus 2012

Supervisors:

Dr. S.J.B.A (Stijn) Hoppenbrouwers, Faculteit NWI, Radboud Universiteit Nijmegen

Ing. W. van . Stokkum MSc, Sr. Business Architect, Everest B.V.

Abstract

Business Proces Management suites zijn in hun opmars en worden de laatste jaren steeds meer ingezet om op een snelle, flexibele en meer platformonafhankelijke manier business applicaties te realiseren. Met name bij bedrijven waar de bedrijfsvoering bepaald wordt door veel wet- en regelgeving, zijn dit soort systemen populair.

In de marketinguitingen van de leveranciers van dit soort pakketten wordt veelal geschermd met de belofte dat ‘de business’ zelf deze wetten en regels zou kunnen onderhouden. Deze mogelijkheid zou allerlei voordelen met zich meebrengen waaronder een efficiëntere en effectievere aansluiting van de IT systemen op de bedrijfsvoering.

Hoewel we overtuigd zijn van deze voordelen lijken mensen uit ‘de business’ vaak helemaal niet in staat om dit soort regels middels deze pakketten te kunnen onderhouden. Met name in grote onderhoudsprojecten worden deze zogenaamde business stakeholders nog altijd ver weggehouden van de logica waarmee deze regels worden geïmplementeerd. De onderhoudstaken die betrekking hebben op deze regels worden na specificatie, in zijn geheel uitgevoerd en getest door technische experts, wat een langere doorlooptijd van dit soort wijzigingen als gevolg heeft. De focus op steeds hogere productiviteit en een als maar kortere time-to-market van producten zorgt ervoor dat de wens vanuit de business stakeholders om deze onderhoudstaken over te nemen alsmaar groeit.

Voor dit master thesis onderzoek hebben we een uitgebreide analyse uitgevoerd op een onderhoudsproject voor de House applicatie bij Westland Utrecht Bank. Met deze analyse hebben we getracht inzicht te krijgen in de redenen die er voor zorgen dat business stakeholders niet bij dit soort onderhoudstaken worden betrokken en de problemen die moeten worden opgelost om deze onderhoudstaken wel voor business stakeholders beschikbaar te maken.

De resultaten van deze analyse laten echter zien dat het idee dat logica door business stakeholders kan worden onderhouden mogelijk nog verder van de realiteit staat als gedacht.

Voorwoord

Deze master thesis is het resultaat van een 8 maanden durend onderzoek bij Everest B.V., in opdracht van de Radboud Universiteit Nijmegen. Dit onderzoek was voor mij bijzondere, soms frustrerende, maar vooral erg leerzame ervaring.

Ik zou graag de volgende personen willen bedanken voor het mede mogelijk maken van dit verslag en het onderzoek:

Allereerst wil ik mijn begeleiders, Stijn Hoppenbrouwers en Wim van Stokkum , bedanken voor hun uitstekende begeleiding en inhoudelijke ondersteuning. Zonder hun feedback en sturing was het mij nooit gelukt om mijn onderzoek en thesis op een goede en manier en binnen de gestelde periode te realiseren.

Daarnaast wil ik graag de medewerkers van Everest bedanken voor hun medewerking en tijd gedurende deze periode. De interviews, mailwisselingen en aangereikte documentatie hebben als waardevolle bronnen gediend voor mijn onderzoek.

Ten slotte wil ik mijn familie, vrienden en in het bijzonder mijn vriendin Roxanne bedanken voor hun interesse, steun en vaak welkome afleiding gedurende deze afsluitende periode van mijn studententijd.

Paul Verhoeven,
27-8-2012, ‘s-Hertogenbosch

Inhoudsopgave

1. Inleiding

1.1 Achtergrond

Een bedrijfsmodel of enterprisemodel is vaak een semi-formele of formele beschrijving van één of meer aspecten van een bedrijf, bijvoorbeeld de organisatie structuur, processen, regels, informatie of infrastructuur. Deze modellen beschrijven een huidig beeld (as-is) of een toekomstig beeld (to-be) van een bepaald aspect van een dergelijke organisatie.

Modellen dienen meerdere doelen. Modellen kunnen bijvoorbeeld sturing bieden aan een organisatie door als blauwdruk te dienen voor de gewenste situatie. Zo helpt het maken van procesmodellen bij het duidelijk definiëren van taken binnen een organisatie, wat uiteindelijk kan leiden tot een efficiëntere bedrijfsvoering. Ook kunnen modellen worden ingezet bij het verkrijgen van inzicht in de huidige situatie. Zo kan het modelleren van een bepaald aspect van de organisatie helpen bij het identificeren van knelpunten. Daarnaast kunnen modellen ook gebruikt worden als ontwerp voor IT-systemen, zoals bijvoorbeeld voor de realisatie van een informatie systeem dat moet aansluiten bij de huidige manier van werken binnen een organisatie.

Het maken van bedrijfsmodellen van verschillende aspecten van je organisatie wordt in het algemeen dan ook gezien als een best practice. Bedrijven zijn echter constant onderhevig aan interne en externe veranderingen, zoals reorganisaties, veranderingen in wet- en regelgeving, concurrentie, veranderende klantwensen, etc. Het is daarom belangrijk om deze bedrijfsmodellen niet enkel eenmalig te creëren, maar constant te onderhouden en aan te sluiten bij de huidige situatie.

De behoefte om deze modellen te creëren en te onderhouden heeft geleid tot de ontwikkeling van allerlei modelleernotaties en ondersteunende software tools om deze modellen in te definiëren, te presenteren en te beheren. Doordat deze digitale modellen veelal middels formele talen zijn beschreven, zijn deze software tools ook het startschot geweest voor veel wetenschappelijk onderzoek naar het interpreten en analyseren van de formele data uit deze modellen. Dit heeft onder andere geleid tot nieuwe executeerbare modelleertalen zoals BPEL en BPMN 2.0 en technieken om deze modellen automatisch te kunnen analyseren.

Met name het moeten kunnen executeren van modellen heeft geleidt tot nieuwe ontwikkelmethodes en ontwikkeltools gericht op het scheiden en apart beheren van executeerbare bedrijfsprocessen en –regels. Zo zijn er Business Process Modeling tools ontstaan om processen in een soort centrale database onder te brengen en middels interfaces (API's) aan te bieden aan externe systemen. Hierna volgde de Business Rules Approach voor het scheiden en centraliseren van business rules (bedrijfsregels) uit deze processen. Deze executeerbare business rules werden volgens hetzelfde idee gecentraliseerd middels Business Rules Management tools.

Het scheiden en centraal onderbrengen van executeerbare processen en business rules uit IT-systemen biedt een aantal voordelen.

Allereerst maakt het centraliseren van business rules en processen het mogelijk om deze modellen te delen over verschillende applicaties, dit is een groot voordeel in huidige organisaties waarin vaak een grote verscheidenheid aan IT-systemen actief is.

Daarnaast kan het scheiden van de business rules en processen van elkaar en van de rest van het systeem, beheersmatige voordelen bieden. Door een goede scheiding te maken tussen deze onderdelen zijn er minder afhankelijkheden waardoor wijzigingen sneller en met minder risico kunnen worden geïmplementeerd.

Een andere grote belofte die dit soort systemen maken is dat ze de business zelf in staat zouden stellen om deze executeerbare modellen te beheren. Hierdoor zouden de IT-systemen welke gebruikmaken van deze modellen beter en sneller kunnen worden aangesloten op de bedrijfsvoering.

Bovengenoemde voordelen maken het voor organisaties makkelijk om zich te kunnen aanpassen, een eigenschap die gezien de huidige dynamiek in de markten als een belangrijk concurrentievoordeel wordt gezien.

De trend van de laatste jaren is dat deze tools en modellen steeds meer geïntegreerd worden in grote ontwikkel platformen genaamd Business Process Management suites (BPM suites) waarin naast modellen voor processen en business rules ook domeinmodellen en interactiemodellen kunnen worden beheerd. Deze systemen kunnen vervolgens op basis van deze modellen complete business services en business applicaties genereren.

Ondanks de beloofde voordelen van deze systemen en het feit dat deze tools al geruime op de markt zijn, lijkt de praktijk erop te wijzen dat de deze voordelen niet altijd worden behaald. Zo blijkt met name het idee dat de business zelf deze systemen zouden kunnen onderhouden nog vaak geen realiteit te zijn.

Everest B.V., een Nederlandse leverancier en ontwikkelaar van BPM suite Aquima , deelt deze ervaring. De implementaties die Everest voor haar klanten realiseert maken een groot deel van de bovengenoemde voordelen waar voor hun klanten. Echter het zelf kunnen onderhouden van deze implementatie blijkt vaak nog problematisch. De interesse van Everest naar de onderliggende oorzaken van dit probleem heeft geleid tot het onderzoeksvoorstel zoals beschreven staat in paragraaf 1.2

1.2 Onderzoeksvoorstel

Everest is een Nederlandse ontwikkelaar en leverancier van de BPM suite Aquima en realiseert implementaties op basis van dit product voor een groot aantal prominente bedrijven.

Everest richt zich met hun Aquima product op klanten waarin een hoge dynamiek van IT-systemen wordt verwacht, zoals bedrijven in de publieke en financiële sector. Deze Aquima implementaties zijn vaak complexe business applicaties die ingezet worden in de middle office en front office van deze bedrijven.

Hoewel Aquima implementaties voor een groot deel uit modellen bestaan, worden deze implementaties vanwege hun complexiteit niet enkel door de klanten zelf gerealiseerd, maar in samenwerking met specialisten van Everest. Everest heeft hiervoor getrainde business engineers en analisten in dienst, die naast modelleer ervaring ook beschikken over kennis van het business domein van de klant. Nadat een Aquima implementatie is gerealiseerd wordt er doorgaans een onderhoudsproject gestart waarin een klantenteam van Everest ondersteuning biedt bij het implementeren van wijzigingsverzoeken van de klant.

In een dergelijk onderhoudsproject kunnen we twee type stakeholders onderscheiden, de business stakeholders en de IT stakeholders. De IT stakeholders beschouwen we als de stakeholders met een technische achtergrond, dit zijn de medewerkers uit het Everest klantteam en de technische medewerkers van de klant. De business stakeholders zijn de stakeholders die werkzaam zijn bij de klant welke niet beschikken over een technische achtergrond en welke direct betrokken zijn bij het aanleveren en implementeren van de wijzigingsverzoeken.

De mate van ondersteuning die het Everest klantteam in deze onderhoudsprojecten levert, verschilt in de praktijk per klant. Sommige klanten kunnen de wijzigingen grotendeels zelf doorvoeren, al dan niet met de hulp van technische medewerkers van de klant. Hierbij hoeft het klantteam enkel bij QA en grote wijzigingen ondersteuning te bieden. Aan de andere kant zijn er klanten welke voor het onderhoud vrijwel geheel afhankelijk van het Everest klantteam. Het laatste blijkt vaak voor te komen bij de meer complexe en grotere Aquima implementaties. Deze situatie wordt als niet ideaal gezien aangezien het leidt tot een langere doorlooptijd dan nodig voor klanten van Everest. Everest heeft de behoefte om deze knelpunten met betrekking tot het beheer voor hun klanten op te lossen.

De huidige wetenschappelijke gemeenschap die zich richt op de problemen rondom het modelleren van IT systemen lijkt zich echter met name te focussen op de problemen gedurende de realisatie van dergelijke systemen. Zo worden er veel aandacht besteed aan de ontwikkeling van nieuwe modelleertalen en ontwikkelmethodes voor het beter en sneller kunnen realiseren van IT systemen.

Echter tonen sommige onderzoeken aan dat van de kosten die gedurende de leeftijd van IT systemen worden gemaakt, zo'n 60% tot 80% in de onderhoudsfase plaatsvindt (Lientz, 1978). In het licht van deze cijfers wordt er volgens ons te weinig gekeken naar de problemen die specifiek gedurende deze onderhoudsfase plaatsvinden en hoe de modelleertalen en tools op dit vlak kunnen worden verbeterd.

Meer onderzoek naar de hoe de huidige generatie modelleer omgevingen tijdens de onderhoudsfase worden gebruikt is volgens ons dan ook noodzakelijk. Om het onderzoek in deze richting te stimuleren biedt Everest de onderhoudsprojecten bij haar klanten aan als onderzoeksonderwerp.

1.3 Onderzoeksvraag

In deze thesis gaan we in op het onderzoeksvoorstel van Everest uit paragraaf 1.2.

De Aquima BPM suite is ontwikkeld vanuit het oogpunt dat gevorderde business stakeholders voor een groot deel zelf onderhoud kunnen doen aan de implementatie. Zoals uit het onderzoeksvoorstel van Everest blijkt is dit echter voor een deel van haar klanten op dit moment niet mogelijk.

Het doel van dit onderzoek is om inzicht te geven in de problemen die hieraan ten grondslag liggen en mogelijke oplossingsrichtingen aan te wijzen die onderhoud door business stakeholders mogelijk kunnen maken.

De onderhoudswerkzaamheden in dergelijke onderhoudsproject zijn verschillend van aard en kunnen betrekking hebben op verschillende delen van de applicatie. Zo kan men de volgende type onderhoudswerkzaamheden onderscheiden (Swanson, 1976):

- **Correctief onderhoud**: Aanpassing van software na oplevering voor het corrigeren van ontdekte fouten.
- **Adaptief onderhoud**: Aanpassing van software met als doel het systeem bruikbaar te houden in een veranderende en veranderde omgeving.
- **Perfectief onderhoud**: Aanpassing van software na oplevering voor het verbeteren van de performance of andere kwaliteitseigenschappen.
- **Preventief onderhoud**: Aanpassing van software voor het detecteren en oplossen van potentiële fouten.

In de praktijk bij Everest blijkt dat het grootste deel van de onderhoudswerkzaamheden uit adaptief onderhoud te bestaan. Daarnaast is dit ook het type onderhoud waarbij de business stakeholders het meest zijn betrokken (zij zetten veranderingen in de bedrijfsvoering in gang of merken ze op). In de context van dit onderzoek richten we ons daarom enkel op het adaptief onderhoud aan Aquima applicaties.

Voor puur visuele aanpassingen en aanpassingen op het gebied van autorisaties en productconfiguraties biedt Everest op dit moment al speciale editors aan welke door de IT of business stakeholders kunnen worden gebruikt. Hiermee wordt echter maar een klein deel van de onderhoudswerkzaamheden ondersteund. Gezien de markten waarin Everest haar Aquima product verkoopt, vindt het grootste deel van de wijzigingen in de praktijk plaats in de logica van de Aquima applicaties. Voor dit onderzoek richten we ons daarom specifiek op het onderhouden van logica als onderdeel van adaptief onderhoud.

Idealiter zou een vergelijkend onderzoek tussen verschillende onderhoudsprojecten van de Aquima klanten van Everest veel inzicht kunnen geven in dit probleem. Helaas zou een dergelijk onderzoek qua tijd buiten de scope van een master thesis onderzoek vallen. Vanwege de beperkte tijd van dit

onderzoek, hebben we ons daarom gelimiteerd tot het onderzoeken van één specifieke Aquima klant waarbij business stakeholders zelf geen logica onderhouden. We hebben hiervoor de Aquima implementatie van Westland Utrecht Bank gekozen, genaamd House. Westland Utrecht Bank is een grote hypotheekverstrekker in Nederland en de House applicatie is een business applicatie die ingezet wordt als middle office systeem voor het verwerken van hypotheek aanvragen.

Hoewel we ons met dit onderzoek richten op één specifiek onderhoudsproject, hopen we problemen en oplossingsrichtingen te kunnen identificeren welke niet enkel voor House gelden maar ook in bredere zin beschrijvend en toepasbaar zijn. We zullen in de conclusies van deze thesis de toepasbaarheid van onze bevindingen verder toelichten.

Met bovengenoemde grenzen in acht genomen proberen we de volgende onderzoeksvraag te beantwoorden middels dit onderzoek:

Hoofdvraag: Hoe kunnen we het onderhoud van logica door business stakeholders mogelijk maken?

Om deze vraag te beantwoorden kunnen de volgende deelvragen worden gesteld:

Deelvraag 1: Waarom is het een probleem dat business stakeholders geen onderhoudsactiviteiten op het gebied van logica kunnen uitvoeren?

Deelvraag 2: Welke problemen verhinderen op dit moment het onderhoud van logica door business stakeholders?

Deelvraag 3: In welke mate zijn deze problemen op te lossen?

1.4 Onderzoeksmethoden

Omdat er op dit moment in de literatuur geen duidelijk beeld is van de problemen die het onderhoud van logica door business stakeholders in de weg staat, is dit onderzoek exploratief van aard. Voor het verkrijgen van inzicht in de problemen die er spelen zullen we daarom putten uit bronnen die aanwezig zijn in de praktijk bij Everest en uit reeds aanwezige literatuur over dit onderwerp.

Hieronder zullen we de bronnen waar we via Everest over kon beschikken kort toelichten:

1.4.1 Interviews

Tijdens het onderzoek stonden we vrij om personen binnen Everest te interviewen welke betrokken waren bij de House implementatie. Om een goed beeld te krijgen van de onderhoudsactiviteiten en de problemen binnen het House project hebben we getracht personen te interviewen welke tussen de business en IT stakeholders van het project in staan. In het House onderhoudsproject hebben we daarom verschillende interviews afgenomen bij zowel de productmanager van Everest Mortage Solutions en een (technische) business analist uit klantteam voor House.

Het doel van deze interviews was om zoveel mogelijk inzicht te krijgen in het onderhoudsproces en de problemen die in de praktijk spelen. Vanwege de onbekendheid van de problemen en de beperkte beschikbaarheid van de geïnterviewden is er daarom gekozen voor een semi-gestructureerde interview methode.

1.4.2 Documentatie

We hebben voor dit onderzoek toegang gekregen tot een set van 500 change management documenten welke tussen begin 2007 en eind 2011 zijn opgeleverd. Deze change management documenten beschrijven verschillende wijzigingsverzoeken voor de House applicatie. Er zijn verschillende type documenten te onderscheiden zoals Request For Change documenten en requirements documenten al dan niet gerelateerd aan elkaar. In de toelichting van de House casus in Hoofdstuk 3 gaan we verder in op de verschillen tussen deze documenten en de relaties ertussen.

De documenten die we hebben gekregen zijn niet alles omvattend en compleet en zijn slechts een deel van de documentatie welke is opgeleverd gedurende de bovengenoemde periode. Daarnaast zijn deze documenten slechts een stroom van communicatie tussen de stakeholders over de onderhoudswerkzaamheden. Het kan zijn dat bepaalde details van de specificaties via andere kanalen zijn gecommuniceerd zoals via persoonlijk contact of email, en om die reden ontbreken in de documenten. Derhalve hebben we deze documenten dan ook niet kunnen gebruiken voor kwantitatief onderzoek. Daarnaast zou een dergelijke analyse niet binnen de scope van dit onderzoek hebben gepast. We hebben deze documenten echter wel gebruikt om een beeld te vormen van de verschillende type wijzigingsverzoeken die voorkomen. Daarnaast geeft het een indicatie voor de kwaliteit en bruikbaarheid van de specificaties.

1.4.3 Implementatie en ontwikkelomgeving

Gedurende het onderzoek heeft Everest ons tevens toegang gegeven tot de testomgeving van House. Deze testomgeving bood zowel toegang tot de applicatie als de implementatie in de Aquima ontwikkelomgeving.

Dit heeft ons de mogelijkheid gegeven om de applicatie, de implementatie maar ook de bruikbaarheid van de Aquima omgeving in combinatie met deze implementatie uitgebreid te onderzoeken.

1.5 Aanpak onderzoek en structuur van de thesis

In dit hoofdstuk ga ik verder in op de aanpak van het onderzoek en de structuur van de thesis.

Om de hoofdvraag van dit onderzoek te kunnen beantwoorden zullen we allereerst de deelvragen beantwoorden in de hoofdstukken 2 tot en met 7. Uiteindelijk zullen we de antwoorden op deze deelvragen gebruiken voor de conclusie in hoofdstuk 8 waarin we antwoord proberen te geven op de hoofdvraag.

De structuur van deze thesis is daarmee als volgt:

Hoofdstuk 2: Onderhoud door business stakeholders
In dit hoofdstuk zullen we allereerst een algemene omschrijving geven van het business-IT alignment probleem zoals dat voorkomt in veel organisaties. Vervolgens zullen we toelichten hoe het business-IT alignment probleem zich afspeelt in de context van de House casus. Ten slotte zullen we het belang van onderhoud door business stakeholders toelichten.

Hoofdstuk 3: Omschrijving casus House
In dit hoofdstuk wordt een algemeen beeld gegeven van de House casus. We zullen de volgende onderdelen van de casus behandelen: het huidige changeproces, de stakeholders, de functionaliteit van de House applicatie, het ontwerp van de implementatie en het Aquima platform.

Hoofdstuk 4: Analyse huidige onderhoud situatie House casus
In dit hoofdstuk gaan we dieper in op de huidige onderhoudssituatie in het House onderhoudsproject. Op basis van de interviews zullen we beschrijven welke redenen er op dit moment zijn om onderhoud door business stakeholders niet toe te staan.

Hoofdstuk 5: Analyse onderhoudsmogelijkheden House casus
In hoofdstuk 5 zal worden gekeken naar de daadwerkelijke mogelijkheden voor business stakeholders op het gebied van onderhoud. Hiervoor kijken we naar verschillende aspecten. Allereerst wordt er gekeken over welke kennis en vaardigheden de business stakeholders en beschikken. Daarna gaan we in op de verschillende onderhoudstaken in het House changeproces. Vervolgens zullen we een analyse uitvoeren op de House implementatie en de ontwikkelomgeving om een inschatting te kunnen maken van de kennis en capaciteiten die nodig zijn bij uitvoeren van deze onderhoudstaken.

Hoofdstuk 6: Algemene problemen onderhoud door business stakeholders
In dit hoofdstuk zullen we met behulp van de bevindingen uit de casus een aantal algemene problemen benoemen die gedurende het onderzoek naar voren zijn gekomen. Tevens zullen we bij elk probleem mogelijke oplossingsrichtingen benoemen.

Hoofdstuk 7: Conclusie
Het antwoord op de hoofdvraag zal in dit hoofdstuk worden gegeven. Tevens zullen we de grenzen en toepasbaarheid van onze bevindingen beoordelen en zullen we een aantal richtingen voor verder onderzoek aanwijzen.

De bovengenoemde hoofdstukken staan als volgt in relatie met de onderzoeksvragen:

Vraag	**Beantwoord in hoofdstuk(ken)**
Deelvraag 1: *Waarom is het een probleem dat business stakeholders zelf geen onderhoudsactiviteiten kunnen uitvoeren?*	Hoofdstuk 2: Onderhoud door business stakeholders
Deelvraag 2: *Welke problemen verhinderen op dit moment het onderhoud van logica door business stakeholders?*	Hoofdstuk 4: Analyse huidige onderhoud situatie House casus Hoofdstuk 5: Analyse onderhoudsmogelijkheden House casus Hoofdstuk 7: Problemen en oplossingen bij onderhoud door business stakeholders
Deelvraag 3: *In welke mate zijn deze problemen op te lossen?*	Hoofdstuk 7: Problemen en oplossingen bij onderhoud door business stakeholders
Hoofdvraag: *Hoe kunnen we het onderhoud van logica door business stakeholders mogelijk maken?*	Hoofdstuk 8: Conclusie

2. Onderhoud door business stakeholders

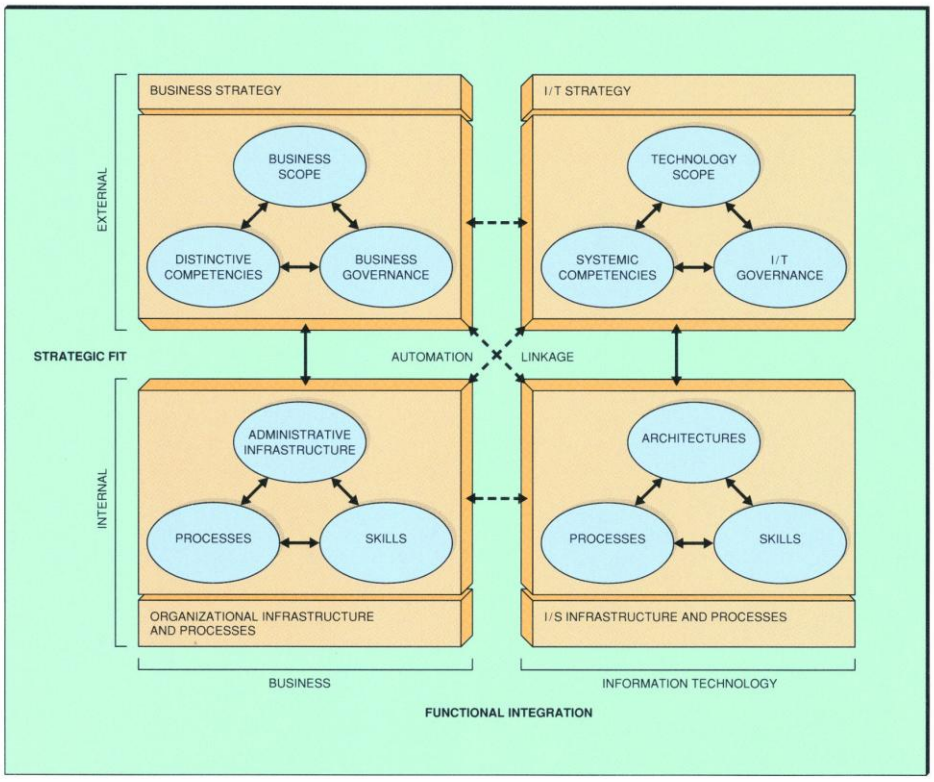
In hoofdstuk 1 hebben we de voordelen van BPM suites binnen organisaties kort behandeld. In dit hoofdstuk zullen we dieper ingaan op de onderliggende problemen die dit soort ontwikkelplatformen proberen op te lossen en de noodzaak om business stakeholder in het onderhoud van IT-systemen te betrekken.

Op dit moment worden BPM suites onder andere in de markt gezet met als belofte de business meer te betrekken in het creëren en onderhouden van deze systemen. Deze belofte wordt gedaan omdat ze inspelen op een algemeen probleem wat zich al sinds de intrede van IT in organisaties manifesteert: het business-IT alignment probleem.

Hoewel er veel definities zijn voor business-IT alignment, gebruiken we in deze thesis de volgende definitie uit het Strategic Alignment Model (Henderson, 1992):

"Een effectieve en efficiënte afstemming tussen de business en IT strategieën binnen een organisatie"

Met effectief en efficiënt wordt in dit geval bedoeld dat er door de afstemming van de strategieën waarde wordt gecreëerd voor de organisatie. Met strategie wordt zowel de formulering als de uitvoering van deze strategie bedoeld.



Figuur 1: Strategic Alignment Model

Een veelgebruikt model wat ingaat op business-IT alignment is het Strategic Alignment Model (Henderson, 1992) (zie Figuur 1). In dit model worden vier domeinen onderkent die met elkaar in overeenstemming moeten zijn om deze ideale afstemming te realiseren: de business strategie, de IT strategie, de business en de IT. Het SAM model identificeert daarnaast twee type integraties tussen business en IT domeinen, namelijk strategische integratie en functionele integratie. Strategische integratie wordt gezien als de afstemming tussen de externe componenten van de organisatie, de business strategie en de IT strategie. Functionele integratie wordt gezien als de afstemming tussen de interne componenten, de organisatorische infrastructuur en processen en de IT infrastructuur en processen.

Het business-IT alignment probleem wordt gezien als het probleem in de afstemming tussen alle vier de domeinen. Een veelgenoemde oorzaak van dit probleem zijn de grote verschillen tussen de mensen die in het IT domein werken en de mensen die in het business domein werken. Zo verschillen IT stakeholders vaak sterk in kennis, vaardigheden en cultuur van business stakeholders. Deze verschillen zorgen ervoor dat afstemming tussen business en IT stakeholders vaak problematisch verloopt.

In de context van dit onderzoek kijken we met name naar hoe dit probleem zich manifesteert tijdens de functionele integratie in het SAM model, waar IT infrastructuur en processen moeten worden afgestemd op de business infrastructuur en de business. Door de verschillen in kennis en vaardigheden zijn beide stakeholders vaak gebonden tot hun eigen expertise, IT stakeholders hebben vaak weinig kennis van hoe de business werkt terwijl business stakeholders vaak niet de kennis en vaardigheden bezitten om de systemen zelf aan te passen of te realiseren. Om tot een succesvolle implementatie te komen moet er daarom een gedeeld begrip van het probleem en oplossing ontstaan. Dit gedeeld begrip wordt echter vaak om verschillende redenen niet bereikt. Zo neemt men er uit kosten overwegingen vaak geen tijd voor, of zegt of denkt men elkaar te begrijpen terwijl dit daadwerkelijk niet het geval is.

De gevolgen van het business-IT alignment probleem zijn vaak IT systemen welke niet goed aansluiten op de business of tegen te hoge kosten zijn gerealiseerd, ze bieden in die zin vaak geen goede return on investment voor de organisaties waarin ze zijn ingevoerd.

De gedachte dat de business in staat zou zijn om de implementatie modellen van deze systemen te kunnen lezen en zelfs zou kunnen aanpassen, wordt al jaren gezien als de heilige graal voor het oplossen van dit probleem. Aan de ene kant zouden de IT systemen mogelijk beter aansluiten en sneller gerealiseerd kunnen worden om dat de problematische communicatie tussen deze twee partijen voor een deel zou kunnen worden vermeden. Aan de andere kant zou het ook de business meer gevoel van controle kunnen geven over haar eigen processen.

De komst van de executeerbare bedrijfsmodellen en bijhorende tools kunnen in dit licht gezien worden als oplossingen voor het hierboven beschreven communicatie probleem. Door gebruikt te maken van business-vriendelijke talen en representaties zou een dergelijk model zowel als implementatie model en als lees- en schrijfbaar bedrijfsmodel voor business stakeholders kunnen dienen.

In de praktijk blijkt deze gedachte te rooskleurig. Zo tonen de onderhoudsprojecten bij Everest aan dat business stakeholders nog vaak geen rol spelen bij de directe onderhoud van deze modellen.

Hoewel dit als een probleem wordt gezien door Everest, bieden de Aquima implementaties volgens hun wel voldoende toegevoegde waarde voor hun klanten. Zo biedt het Aquima platform voordelen op gebied van ontwikkeltijd en een hogere aanpasbaarheid doordat er vaak veel minder code hoeft te worden geschreven. Het bovengenoemde communicatie probleem wordt in Everest onderhoudsprojecten vaak vermeden met de inzet van ervaren cross-domein specialisten die ingezet worden als mediators tussen de business stakeholders van de klant en de business engineers en technical engineers van het Everest klantteam. De specialisten hebben uitgebreide kennis van het business domein van de klant en het Aquima platform en de implementatie, daarnaast beschikken ze over goede communicatieve vaardigheden. Hoewel met deze aanpak de kans op een ineffectieve implementaties veroorzaakt door communicatieproblemen aanzienlijk verkleind, lost het de problemen met betrekking tot efficiëntie en het gevoel van gebrek aan controle vanuit de business niet op.

Hoewel we ons realiseren dat business stakeholders waarschijnlijk nooit in het geheel de taken van engineers zouden kunnen overnemen, lijkt elke mogelijkheid om business stakeholders meer actief te laten deelnemen in het ontwikkel- en onderhoudstraject van IT systemen duidelijke voordelen te bieden. Een praktijk onderzoek zoals in deze thesis gepresenteerd, gericht op het identificeren van de problemen die de deelname van business stakeholders verhinderen, lijkt ons dan ook noodzakelijk.

Met de analyse van de huidige onderhoudssituatie bij House hopen we meer duidelijkheid te kunnen krijgen over deze problemen en mogelijke oplossingen voor deze problemen. In hoofdstuk 3 zullen we allereerst de House casus in meer detail toelichten, alvorens we in hoofdstuk 4 en 5 verdere analyse op dit gebied uitvoeren.

3. Omschrijving House casus

3.1 Everest Mortgage Solution

De House business applicatie bestaat in de basis uit het Everest Mortgage Solution (EMS) systeem in combinatie met een klant specifieke module voor Westland Utrecht Bank (WUB).

EMS is een door Everest ontwikkelde standaardoplossing voor de hypotheekmarkt. Het systeem ondersteund het hele hypotheekproces van aanvraag tot notaris. EMS wordt gebruikt door twee grote hypotheekverstrekkers in Nederland, namelijk SNS Hypothekengroep en Westland Utrecht Bank (voorheen Nationale Nederlanden Hypotheken).

Een EMS applicatie bestaat uit twee onderdelen, een generiek deel en een klant specifiek deel. Het generieke deel verzorgt de standaard functionaliteit waar hypotheekbanken zich onderling niet op onderscheiden in de markt. Dit deel wordt up-to-date gehouden door Everest op het gebied van wet- en regelgeving, technische standaarden en marktnormen.

Met het klant specifieke deel kunnen klanten van EMS zich onderscheiden op commercieel gebied, zoals werkwijzen en bank specifieke acceptatieregels voor hypotheekaanvragen.

Het EMS systeem kan dus op twee manieren door de klantorganisatie worden aangepast:

- Het instellen van de EMS parameters die de standaard functionaliteit regelen
- Ontwikkeling van op maat gemaakte klant modules

De EMS parameters kunnen door de klanten zelf worden aangepast in een speciaal daarvoor bestemde beheerapplicatie genaamd het Product model. De ontwikkeling van de klant specifieke module wordt grotendeels door Everest gedaan. Beide klanten van EMS hebben op deze manier hun eigen parameter instellingen en klant specifieke module.

Figuur 2: Verdeling standaard functionaliteit en klant specifiek deel EMS

3.2 Implementatie

Het EMS systeem is modulair opgezet en bestaat uit de volgende 7 modules:

- **Midoffice**
  De Midoffice module is de hoofd module en is verantwoordelijk voor de invoer en validatie van de hypotheek aanvraag.

- **Beoordeling**
  De Beoordeling module bevat alle beoordelingsregels welke controleren of een aanvraag voldoet aan klant specifiek of algemene acceptatieregels voor hypotheekaanvragen.

- **Product model**
  Het Product model is de configuratie interface voor het instellen van parameters van het EMS system. Hierin kunnen klant specifieke en algemene parameters worden ingesteld voor o.a. product configuratie en prijsregels.

- **Autorisatie model**
  Het Autorisatie model is de configuratie interface voor het instellen van verschillende autorisaties voor gebruikers binnen het systeem.

- **Werklijst**
  Deze module is verantwoordelijk voor de werklijst interface van de applicatie. De werklijst is een lijst met taken die behandelaars op basis van nieuwe hypotheek aanvragen moeten uitvoeren. De werklijst wordt samengesteld op basis van geconfigureerde prioriteringsregels uit de Procescontrole module.

- **Procescontrole**
  De Procescontrole module is verantwoordelijk voor het verloop van het acceptatie proces van hypotheek aanvragen. In deze module worden de prioriteringsregels gedefinieerd welke bepalen in welke volgorde en door wie bepaalde taken moet worden uitgevoerd.

- **Interfaces**
  In de Interfaces module worden alle koppelingen gerealiseerd met externe systemen zoals de beoordelingssystemen van de Nationale Hypotheek Garantie, Bureau Krediet Registratie en interne backoffice systemen zoals SAP.

Een groot deel van het EMS systeem is gemaakt met het Aquima platform. De modules Midoffice, Beoordeling, Werklijst en Procescontrole zijn volledige gerealiseerd in Aquima. De andere modules zijn ontwikkeld middels de Java programmeertaal.

Voor het maken van klant specifieke modules wordt er gebruik gemaakt van overervingscapaciteiten van het Aquima platform. Een klant specifieke module is een extensie op een standaard Aquima module van het EMS systeem.

3.3 House changeproces

De huidige hypotheekmarkt is constant in beweging, niet alleen door de concurrentie tussen de verschillende hypotheekverstrekkers, maar met name door het veranderlijke karakter van de wetgeving en marktnormen omtrent hypotheken.

De veranderingen in de interne en externe omgeving van deze hypotheekverstrekkers leidt uiteindelijk tot verschillende wijzigingen in de EMS applicatie en haar klant specifieke modules.

Veranderingen in de EMS applicatie welke voor de gehele hypotheekmarkt gelden, zoals wet- en regelgeving en marktnormen, worden geïnitieerd, gepland en geïmplementeerd door het EMS onderhoudsteam van Everest. De veranderingen welke betrekking hebben op de klant specifieke modules worden geïnitieerd vanuit de klant en met behulp van de klantteams ingepland en geïmplementeerd.

Omdat we voor dit onderzoek enkel het House onderhoudsproject onderzoeken, zullen we alleen het changeproces beschrijven van het adaptief onderhoud aan de klant specifieke modules.

Het changeproces van het House onderhoudsproject bestaat uit de volgende processtappen:

3.3.1 Indienen Request For Change (RFC)

Het changeproces begint met het indienen van een Request For Change (RFC). Een RFC is een document welke een omschrijving bevat van het wijzigingsverzoek van de stakeholder.

Een wijzigingsverzoek ontstaat vanuit de behoefte voor verandering. Dit kan zijn door beleidswijzigingen maar ook door ervaringen in het gebruik van de applicatie. De stakeholders welke de wijzigingsverzoeken indienen in het House changeproces zijn over het algemeen functionele applicatie managers, productmanagers of business analisten. Functionele applicatie managers dienen RFC's in namens de eindgebruikers, deze RFC's zijn vaak ontstaan door ervaringen in het gebruik van de applicatie. Productmanagers en business analisten dienen doorgaans RFC's in op basis van beleidswijzigingen.

3.3.2 RFC bespreking Review Board

De volgende stap in het changeproces is de bespreking van de RFC's door het Review Board. Het Review Board is een multidisciplinair team van verschillende business en IT stakeholders waarbij zowel de stakeholders van de klant als die van Everest zijn vertegenwoordigd. In een periodieke meeting wordt er meer informatie verzameld over de nieuwe RFC's en worden ze beoordeeld op impact, kosten en haalbaarheid.

3.3.3 Review Board beslist over implementatie RFC

De eerste beslissing die het Review Board neemt met betrekking tot de RFC's is of ze daadwerkelijk als aanpassingen op de applicatie moeten worden geïmplementeerd. Zo zijn sommige RFC's onhaalbaar of kunnen beter als werkinstructie worden geïmplementeerd.

3.3.4 Review Board beslist over specificatie niveau RFC

In deze stap wordt er gekeken of het wijzigingsverzoek dat beschreven staat in de RFC voldoende is om over te gaan tot ontwerpen en implementeren van de wijziging. Bij grote veranderingen of complexe regelgeving is het vaak wenselijk om het wijzigingsverzoek op te splitsen in meer gedetailleerde beschrijvingen genaamd Customer Requirements. Wanneer dat niet noodzakelijk is dan wordt er gelijk over gegaan naar de ontwerp fase, met het opstellen van System Requirements.

3.3.5 Opstellen Customer Requirements

Customer Requirements documenten worden opgesteld indien een wijzigingsverzoek met meer detail dient te worden gespecificeerd. De business analisten van WUB zijn verantwoordelijk voor het opstellen van deze documenten. Elke Customer Requirement bevat een uniek nummer waardoor de requirements kunnen worden gerelateerd aan de RFC's.

3.3.6 Opstellen System Requirements

System Requirements documenten bevatten alle handelingen die moeten worden uitgevoerd op de implementatie modellen van de applicatie. In dit document wordt de oplossing voor het probleem uit het wijzigingsverzoek ontworpen. De business analisten en business engineers uit het Everest klantteam zijn verantwoordelijk voor het opstellen van deze documenten. Elke System Requirement bevat wederom een uniek nummer waardoor ze kunnen worden gerelateerd aan RFC's of Customer Requirements.

3.3.7 Implementatie

De implementatie processtap wordt grotendeels uitgevoerd door de business engineers en technical engineers uit het Everets klantteam. Business engineers zijn verantwoordelijk voor de wijzigingen in de Aquima implementatie modules, terwijl technical engineers wat complexere technische handelingen verrichten zoals het aanpassen van maatwerk code en koppelingen met andere systemen. Zeer simpele wijzigingen welke configureerbaar zijn zoals parameterwaarden en autorisaties worden uitgevoerd door de functioneel applicatie managers van WUB zelf middels de configuratie editors van EMS.

3.3.8 Test

De test processtap bestaat uit een aantal kleinere stappen. Het schrijven van de test scenario's en unittesting gebeurd tijdens en voor het implementeren al door de business engineers en technical engineers op de ontwikkelomgeving. Deze test scenario's worden vervolgens toegevoegd aan de regressietest welke op de testomgeving wordt uitgevoerd. Vervolgens wordt de testomgeving gekopieerd naar de acceptatie omgeving, waar een acceptatietest wordt uitgevoerd door een testteam van WUB. In de acceptatietest wordt er gekeken of de geïmplementeerde wijzigingen wel voldoen aan de eisen die gesteld zijn in het RFC document.

3.3.9 Migratie naar productie

De laatste stap in het changeproces is het migreren van de geteste aanpassingen vanuit de acceptatieomgeving naar de productieomgeving.

3.4 Stakeholders

In de paragraaf 3.3 zijn verschillende stakeholders al benoemd. In deze paragraaf zullen we hun rol en hun werkzaamheden in het onderhoudsproject nog even kort toelichten. Onderstaande lijst is waarschijnlijk niet alles omvattend, we hebben ervoor gekozen om enkel de stakeholders te benoemen welke veelvuldig naar voren kwamen in de verschillende interviews en change management documenten.

3.4.1 Westland Utrecht Bank

Gebruikers

Een gebruiker is iemand die de House applicatie gebruikt voor het opvoeren van een hypotheek aanvraag van een (potentiële) klant van WUB. Deze gebruikers worden doorgaans behandelaars genoemd. Als een gebruiker niet tevreden is over het deel van de applicatie dat hij gebruikt dan rapporteert hij dit aan de functioneel beheer afdeling via een intern issue tracking system (een gebruiker heeft geen bevoegdheid om direct RFC's aan te maken).

Functioneel applicatie managers

Een functioneel applicatie manager is verantwoordelijk voor het functionele beheer van de verschillende bedrijfsapplicaties binnen WUB met als doel dat de gebruikers ten alle tijden hun werk kunnen uitvoeren. Een onderdeel hiervan is het afhandelen van bugreports en functionele wijzigingsverzoeken van House gebruikers. De functionele applicatie beheerder bepaalt welke wijzigingsverzoeken een aanpassing vereisen van de applicatie. Hij of zij heeft hiervoor de verschillende configuratie editors van EMS tot zijn beschikking zoals het Product model en het Autorisatie model. Hiermee kan hij kleine wijzigingen zelf door voeren. Voor wijzigingsverzoeken welke niet geconfigureerd kunnen worden wordt een RFC opgesteld.

Productmanagers

Een product manager beheerd één of meerdere hypotheekproducten. RFC's voor nieuwe hypotheekproducten of veranderingen in bestaande hypotheekproducten zijn typisch afkomstig van een product manager.

Business analisten

Een business analist bij WUB is iemand die gespecialiseerd is in het ontwerpen en analyseren van de business. Binnen het House project zijn business analisten met name actief bij het opstellen van verschillende Customer Requirements en de acceptatietests.

3.4.2 Everest

Business analisten

Een business analist in het klantteam van Everest fungeert als tussenpersoon tussen de business engineers van Everest en de business analisten van WUB. Ze controleren de Customer Requirements en RFC's en helpen mee met het opstellen van de System Requirements.

Business engineers
Een business engineer heeft veel kennis van de EMS implementatie en het Aquima platform. Business engineers helpen mee met het opstellen van de System Requiremenst en implementeren deze ook.

Technical engineers
Technical engineers bezitten naast kennis van de implementatie en het Aquima platform ook over diepgaande technische kennis. Technical engineers worden onder andere ingezet voor aanpassingen in de maatwerk Java code en het realiseren van koppelingen.

3.4.3 Business stakeholders vs IT stakeholders

In paragraaf 1.2 zijn we al kort ingaan op de verdeling van de IT en business stakeholders binnen de onderhoudsprojecten van Everest. Binnen de House casus is deze verdeling als volgt te maken:

Business stakeholders
De meeste stakeholders bij WUB zijn business stakeholders, hieronder vallen dus de gebruikers, productmanagers, en business analisten.

IT stakeholders
De stakeholders bij Everest zijn over het algemeen IT stakeholders, hieronder vallen de business engineers en technical engineers

Zoals we in hoofdstuk 2 hebben toegelicht, zijn de business analisten uit het Everest klantteam eigenlijk meer hybride stakeholders, aangezien ze de kwaliteiten en kennis van beide stakeholders bezitten. Je zou je kunnen afvragen waarom organisaties niet enkel dit type hybride medewerkers aannemen. Helaas blijkt ondanks de opkomst van studies die deze twee disciplines combineren, het huidige aanbod op de arbeidsmarkt van deze zgn. *dual thinkers* nog altijd laag (Dualthinkers, 2010). Dit maakt het voor bedrijven moeilijker om dit type medewerkers aan te nemen.

3.5 Aquima platform

In deze paragraaf zullen we een korte omschrijving geven van het Aquima platform en zijn mogelijkheden.

Het Aquima platform is een modelgedreven Business Proces Management suite voor het ontwikkelen van business applicaties. Het Aquima platform bestaat uit twee onderdelen, de Aquima Runtime en de Aquima Studio. De Aquima Studio is de ontwikkelomgeving waarin applicaties kunnen worden gemodelleerd. Met de Aquima Runtime kunnen deze modellen geëxecuteerd en getest worden. De Aquima Runtime wordt aangeboden voor verschillende platformen, zodat dezelfde applicatie modellen op meerdere platformen kunnen worden uitgevoerd.

In de context van dit onderzoek zullen we de Aquima Studio op bruikbaarheid door business stakeholders gaan analyseren. Om deze reden zullen we enkel de Aquima Studio verder toelichten in deze paragraaf.

3.5.1 Aquima Studio

Op dit moment is het ontwikkelen op het Aquima platform enkel mogelijk via de Aquima Studio. De type implementatie elementen welke de Aquima implementatietaal vormen en de ontwikkelomgeving kun je in dit geval niet los van elkaar zien, ze zijn samen verantwoordelijk voor de bruikbaarheid van het platform voor business stakeholders.

3.5.2 Aquima implementatietaal

In het de Aquima Studio is een applicatie opgebouwd uit verschillende modellen zoals een domein model, interactie model, een document model en een procesmodel. Door relaties tussen de elementen van deze verschillende modellen te definiëren en door het toepassen van logica kan er uiteindelijk een executeerbare applicatie worden gerealiseerd.

Onderverdeeld naar het type model heb je binnen de Aquima Studio de beschikking over de volgende type implementatie elementen:

- **Logica:** expressions, business rules, decision tables, decision tree’s
- **Services:** services voor het ophalen en versturen van informatie.
- **Proces:** taken, events, condities
- **Domein:** entiteiten, attributen en relaties ertussen
- **Interactie:** pagina’s, tekstvelden, invulvelden, knoppen

- **Documenten:** documenten, tekstblokken

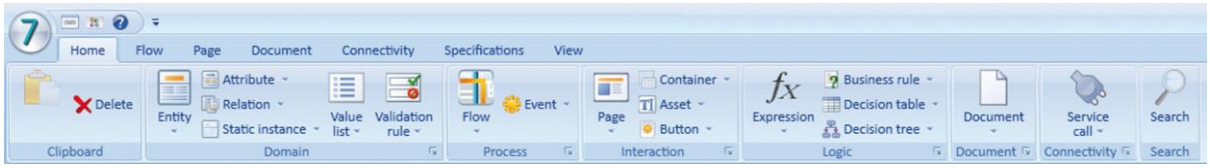
Deze bovengenoemde types implementatie elementen zijn hiermee vergelijkbaar met andere BPM suites op de markt zoals Tibco Business Studio en Cordys Business Proces Management.

3.5.3 Inzetbaarheid

Met aanbieden van de verschillende type modellen en implementatie modellen combineert Aquima concepten uit verschillende tools zoals Business Process Modelling tools en Business Rules Management Systemen. Dit maakt de Aquima Studio multi-inzetbaar, het kan zowel als complete oplossing voor business applicaties worden ingezet als voor het realiseren van een business rule repository, workflow engine en reporting tool.

3.5.4 Interface

Voor de interface van de Aquima Studio is een zogenaamde Microsoft Office look and feel gekozen. Zo wordt er gebruikt gemaakt van stijl elementen uit de Microsoft Office programma's en is ook de zgn. Microsoft Office Ribbon gebruikt voor navigatie binnen de studio. De keuze voor deze look and feel moet de herkenbaarheid van het pakket door business stakeholders vergroten, aangezien er vaak in hun zakelijke omgeving ook met Microsoft Office programma's wordt gewerkt.



Figuur 4: Gebruik van Microsoft Office Ribbon in Aquima studio

3.5.5 Applicatie structuur

Aquima projecten kunnen worden opgedeeld in modules. De inhoud van deze modules is vrij in te delen, ze kunnen alle beschikbare type implementatie elementen bevatten. Zo kunnen modules gebruikt worden om meerdere gerelateerde applicaties binnen één project te beheren, maar om bijvoorbeeld verschillende type logica van elkaar te scheiden.

Daarnaast is het mogelijk om een extensie op een reeds bestaande module te maken door gebruik te maken van overerving functionaliteit van het platform. De klant specifieke aanpassingen aan het EMS systeem worden op deze manier gerealiseerd. Wanneer de klant wil afwijken van de standaard beoordelingsregels, wordt er een klant specifieke module gemaakt die overerft van de standaard EMS Beoordelingsmodule. In deze klant specifieke module vervolgens standaard beoordelingsregels worden geherdefinieerd volgens het acceptatiebeleid van de klant.

3.5.6 Versiebeheer

De Aquima Studio ondersteunt uitgebreide versiebeheer mogelijkheden. Elke verandering wordt geregistreerd en kan worden teruggedraaid. Tevens kunnen meerdere versies van een Aquima applicatie naar elkaar worden gedraaid.

4. Analyse huidige onderhoudssituatie House casus

In dit hoofdstuk zullen we een analyse uitvoeren op de huidige onderhoudssituatie van de House applicatie met betrekking tot het onderhouden van logica. Met onderhoudssituatie bedoelen we de huidige manier waarop de onderhoudsactiviteiten plaatsvinden zoals in het procesmodel in paragraaf 3.3 staat beschreven.

4.1 Positie business stakeholders

Uit het procesmodel blijkt dat de business stakeholders op dit moment niet de mogelijkheid krijgen om zelf aanpassingen te doen aan de logica. Uit de interviews met de productmanager van Everest Mortage Solutions en de business analist uit het House klantteam kwam naar voren dat de keuze voor het uitsluiten van de business stakeholders vooral gebaseerd wordt op twee factoren:

- **Risico's**
  Het huidige changeproces is ingericht om implementatie risico's zoveel mogelijk te vermijden. Er wordt gewerkt met vaste teams van experts welke veel ervaring hebben met de implementatie, het Aquima platform en het business domein. De EMS applicatie is een belangrijke applicatie binnen WUB. Het is een kritisch schakel in het hypotheek aanvraag proces en het binnenhalen van nieuwe klanten. Een verstoring in deze applicatie of een fout in de beoordeling van aanvragen heeft daarom grote gevolgen en kan directe financiële schade aanrichten. Hoewel het test proces grondig is vreest men dat het vrijgeven van de implementatie modellen aan ondeskundige stakeholders een verhoogd risico zal opleveren gezien de complexiteit van de applicatie.

- **Effectiviteit van changeproces**
  Om fouten bij aanpassingen door ondeskundige stakeholders te voorkomen verwacht men dat ze in de huidige situatie veel correctieve maatregelen achteraf zouden moeten nemen om de risico's te kunnen beheersen. Door deze correctieve maatregelen zou de kosten en tijd van het onderhoud toenemen en de effectiviteit van het gehele changeproces afnemen.

4.2 Huidige problemen

De factoren uit paragraaf 4.1 zijn gebaseerd op verwachtingen welke worden gevormd door een aantal problemen die op dit moment al spelen in het huidige changeproces voor de House applicatie. Deze problemen zullen we nu kort toelichten:

4.2.1 Vaardigheden business stakeholders

De business stakeholders welke op dit moment de specificaties voor de implementaties aanleveren hebben moeite met het opstellen van complexe logica. Er is geen moeite met het

opstellen van simpele logische statements zoals ALS x DAN y ANDERS z. Bij complexere geneste logica worden echter fouten gemaakt en/of is de logica op een dergelijke manier gestructureerd dat er bij de ontwerper onzekerheid ontstaat over de betekenis van de statements. Onze bevindingen tijdens het onderzoek van de Customer Requirement en RFC documenten bevestigen dit beeld. De fouten en onduidelijkheden lijken met name te worden veroorzaak doordat business stakeholders de neiging hebben om nesting in natuurlijke taal te vermijden door geneste logica op te schrijven als losse statements waarin een deel van de logica overlapt met de andere statements.

We nemen als voorbeeld een logisch statement voor het besturen van een koffie en thee automaat om dit probleem te illustreren. Een automaat moet koffie of thee maken indien er een product is gekozen, er genoeg geld is ingeworpen en de startknop is ingedrukt. In andere gevallen moet de automaat een informerend statusbericht tonen aan de gebruiker.

Wanneer dit statement op een goede manier is gestructureerd is het prettig te lezen en is het snel duidelijk wat er in welke situatie moet gebeuren:

ALS koffie geselecteerd OF thee geselecteerd DAN
ALS start knop ingedrukt DAN
ALS genoeg geld ingeworpen DAN
maak koffie of thee
ANDERS
toon bericht “niet genoeg geld ingeworpen”
ANDERS
ALS genoeg geld ingeworpen DAN
toon bericht “druk op start”
ANDERS
toon bericht “wacht op geld”

Business stakeholders hebben echter de neiging om het bovenstaande logische statement op te stellen in vier deels redundante statements:

ALS koffie geselecteerd OF thee geselecteerd EN genoeg geld ingeworpen EN start knop ingedrukt DAN maak koffie of thee

ALS koffie geselecteerd OF thee geselecteerd EN niet genoeg geld ingeworpen EN startknop niet ingedrukt DAN toon bericht “wacht op geld”

ALS koffie geselecteerd OF thee geselecteerd EN genoeg geld ingeworpen EN startknop niet ingedrukt DAN toon bericht “druk op start”

ALS koffie geselecteerd OF thee geselecteerd EN niet genoeg geld ingeworpen EN startknop ingedrukt DAN toon bericht “niet genoeg geld ingeworpen”

Dit soort statements veroorzaken vaak verwarring bij de ontwerper, hij of zij moet goed kijken wat de verschillen zijn en zich een beeld hiervan vormen. Daarnaast raakt de business stakeholder zelf ook sneller het overzicht kwijt waardoor hij of zij mogelijk bepaalde randgevallen vergeet. Onder ander het bovenstaande probleem zorgt er op dit moment voor dat complexe logica regelmatig gecorrigeerd en opnieuw gevalideerd moet worden met de indienende stakeholder.

4.2.2 Specificatie van oplossingen door stakeholders

In de verschillende wijzigingsverzoeken uit de change management documenten wordt vaak niet alleen het probleem beschreven maar ook een mogelijke oplossing. Bij wijzigingsverzoeken die betrekking hebben op een specifieke business rule of business concept is dit vaak geen probleem omdat de stakeholders kennis hebben van dit domein. Echter bij wijzigingsverzoeken welke meerdere delen van de applicatie raken, zoals functionele wijzigingen, blijken de beschreven oplossingen regelmatig onjuist of onvolledig.

Zo kan bijvoorbeeld een wijzigingsverzoek bestaan uit het volgende beschrijving van het probleem en de oplossing:

Probleem: "Een klant mag vanaf nu niet jonger zijn dan 21 jaar"
Oplossing: "Het geboortedatum veld van de aanvrager moet daarom vanaf nu geen geboortedatum meer toelaten als het verschil met de huidige datum kleiner is dan 21 jaar"

Hoewel de oplossing technisch mogelijk is kan het zijn dat de ontwerper van het Everest klantteam dit niet wil implementeren als invoer restrictie op dit betreffende veld, maar graag als validatieregel na het invullen van het hele scherm wil implementeren omdat dit bij alle schermen op deze manier gebeurd. Daarnaast zou de ontwerper zich kunnen afvragen of de oplossing wel compleet is, waarschijnlijk moet dezelfde controle plaatsvinden in het wijzigingsformulier voor de aanvraag of bij het importeren van klantgegevens uit een extern systeem.

De reden voor deze foutieve oplossingen wordt gezocht in het feit dat de business stakeholders vaak te weinig overzicht hebben over de gehele functionaliteit van de applicatie en geen kennis hebben van de gevolgen van hun oplossingen op de implementatie. Het beschrijven van oplossingen in deze gevallen wordt dan ook als ongewenst beschouwd. Om deze reden is het huidige changeproces van House ook zo ingericht dat de oplossingen met name worden bedacht en uitgeschreven door de business analisten van het Everest klantteam in de vorm van System Requirements.

4.2.3 Aanbieden van logica

Hoewel het onderhoud van complexe logica door business stakeholders op dit moment als onmogelijk wordt geacht, zijn er volgens de geïnterviewden wel stukken logica in het implementatie model welke wel zouden kunnen worden onderhouden. Het zou dan met name gaan om eenvoudige logica welke weinig impact heeft op de rest van de applicatie, zoals bepaalde simpele beoordelingsregels. Het selecteren van geschikte logica voor onderhoud zou dan echter niet per type regel moeten gaan, maar per individueel stukje logica. Het op deze manier 'cherry picken' van logica zou echter al snel teveel werk zijn gezien het deel van de implementatie dat hiervoor in aanmerking zou komen. De huidige Aquima Studio zou daarnaast niet geschikt zijn om de risico's het aanbieden van deze logica, omdat implementatie risico's niet goed kunnen worden beheerst op dit moment. Zo zouden de logische elementen uit Aquima Studio bijvoorbeeld niet kunnen worden gelimiteerd in expressieve vrijheid. In een decision table kunnen bijvoorbeeld alternatieven makkelijk worden toegevoegd of aangepast zodat ze kritische attributen uit het domeinmodel kunnen beïnvloeden.

4.2.4 Ervaring IT stakeholders WUB

Op dit moment zijn er al configuratie schermen beschikbaar voor de functionele applicatie managers. Deze configuratie schermen bieden toegang tot het wijzigen van allerlei parameterwaarden en ook kleine stukken logica die te maken hebben met de configuratie van producten zoals prijsregels en regels voor het opvragen van bewijsstukken bij nieuwe hypotheek aanvragen.

De huidige ervaringen met dit systeem zijn echter niet geheel positief. Zo blijken de IT stakeholders soms foutieve parameterwaarden in te voeren ondanks een twee ogen beleid waarbij functioneel applicatie managers elkaar controleren. Daarnaast wordt er regelmatig om assistentie gevraagd bij de medewerkers uit het Everest klantteam. Hoewel dit grotendeels te danken is aan het feit dat de huidige interfaces niet meer zijn dan een lijst van invulvelden, zonder veel uitleg en ondersteuning voor invoer, beseft men hierdoor dat het ontwikkelen van een interface voor business stakeholders een grote uitdaging zal zijn.

4.3 Conclusie

Het besluit om deze business stakeholders op dit moment te weren in het uitvoeren van de onderhoudstaken lijkt ons begrijpelijk. Het is goed voor te stellen dat door de genoemde problemen uit paragraaf 4.2 de huidige projectleiding binnen het House onderhoudsproject zorgen baart over de mogelijke risico's en de extra kosten die onderhoud door business stakeholders met zich mee zou kunnen brengen. Deze bezorgdheid is echter voornamelijk gebaseerd op verwachtingen en aannames aangezien er op dit moment maar weinig inzicht is in de problemen die zich daadwerkelijk voor zouden kunnen gaan en de oplossingen die hiervoor zouden moeten worden ontwikkeld.

Om meer inzicht te krijgen in de problemen die moeten worden opgelost zal er grondiger moeten worden gekeken naar hoe de huidige implementatie en ontwikkelomgeving bij de onderhoudsactiviteiten en business stakeholders passen.
In hoofdstuk 5 zullen we daarom een uitgebreide analyse uitvoeren op de verschillende factoren die een rol spelen in de onderhoudsmogelijkheden voor business stakeholders waarna we in hoofdstuk 6 zullen we de gevonden problemen in een bredere zin bespreken en mogelijke oplossingsrichtingen proberen aan te wijzen voor deze problemen.

5. Analyse onderhoudsmogelijkheden House casus

In dit hoofdstuk beschrijven we de analyse die we hebben uitgevoerd op de House casus. We hebben hiervoor verschillende factoren onderzocht die van invloed kunnen zijn op de onderhoudsmogelijkheden voor business stakeholders. Het doel van deze analyse is om inzicht te krijgen in welke onderhoudsmogelijkheden er zijn voor business stakeholders in het House changeproces maar ook te identificeren welke eigenschappen deze onderhoudsmogelijkheden beperken.

De onderzochte factoren zijn:

- **Business stakeholders:** welke kennis en capaciteiten bezitten de business stakeholders en in hoeverre sluiten ze aan bij de kennis en capaciteiten die vereist zijn om de wijzigingen uit te voeren.
- **Onderhoudstaken:** welke type wijzigingen komen er in het onderhoudsproject voor en in welke verhouding.
- **Implementatie:** hoe is de logica waar de geïdentificeerde onderhoudstaken betrekking op hebben geïmplementeerd en in hoeverre verschilt deze implementatie van de specificaties van de onderhoudstaken
- **Ontwikkelomgeving:** in hoeverre ondersteunt de ontwikkelomgeving de onderhoudsactiviteiten die nodig zijn voor het uitvoeren de onderhoudstaken.

Uiteindelijk zullen we op basis van de analyse van deze factoren een concluderend oordeel geven over de onderhoudsmogelijkheden voor business stakeholders in de House casus op dit moment.

In hoofdstuk 6 zullen de geïdentificeerde problemen uit zowel hoofdstuk 5 als uit eerdere hoofdstukken bespreken en zullen we eventuele oplossingsrichtingen aanwijzen voor deze problemen.

Voor dat we beginnen met de analyse van de verschillende onderhoudsfactoren zullen we een aantal theoretische begrippen introduceren welke als raamwerk dienen voor onze analyse.

5.1 Termen en definities

In deze paragraaf zullen we verschillende termen en definities introduceren die betrekking hebben op het onderhoudsproces en de cognitieve activiteiten die tijdens dit proces plaatsvinden.

5.1.1 IEEE Standard for Software Maintenance

De IEEE Standard for Software Maintenance (IEEE, 1998) geeft een goed beeld van de verschillende activiteiten die te vinden zijn in het onderhoudsproces bij de meeste bedrijven. De IEEE standaard gaat uit van dezelfde hoofdactiviteiten als het standaard software life cycle model. De basisgedachte hierachter is dat er zowel bij het ontwikkelen van nieuwe software als het onderhouden van software een vertaling moet plaatsvinden vanuit het specificatiedomein naar het implementatiedomein. Het verschil in deze domeinen moet worden overbrugt door analyse- en ontwerpactiviteiten.

Het onderhoudsproces bestaat volgens deze standaard uit een zestal hoofdactiviteiten met onderliggende deelactiviteiten:

Hoofdactiviteit	**Deelactiviteiten**	**Gerelateerde activiteiten uit House casus**
Identificatie van probleem- en modificatie onderdelen, classificatie en prioritering	• Registreren change request • Classificeren • Accepteren of afwijzen wijziging • Globale impactanalyse • -Prioriteren	• Indienen Request For Change • RFC bespreking Review Board • Review Board beslist over implementatie RFC
Analyse	• Haalbaarheidsanalyse (alternatieven evalueren, inschatting maken kosten) • Gedetailleerde analyse (voorlopige plan opstellen voor ontwerp, implementatie, test en oplevering)	• RFC bespreking Review Board • Review Board beslist over implementatie RFC • Review Board beslist over detailniveau specificatie • Opstellen Customer Requirements

Ontwerp	• Identificeren van modificatie-elementen • Updaten van requirements- en systeemdocumentatie • Creëren van testcases voor regressietesten	• Opstellen System Requirements
Implementatie	• Implementeren • Unittesting • Risicoanalyse	• Implementatie
Regressie/ systeem testen	• System functional test • Regressie test • Interface test	• Testen
Acceptatie testen	• Acceptatie test op functioneel niveau • Interopabilitytest Regressietest	• Testen
Opleveren	• Verplaatsen van product naar productieomgeving	• Migratie naar productie

Tabel 1: IEEE Standard for Software Maintenance

De IEEE standaard moet worden gezien als een alles omvattende template die kan worden gebruikt voor het inrichten van een onderhoudsproces dat is afgestemd op de organisatie en het type wijzigingen, het is dus niet bedoeld als blauwdruk voor het ideale changeproces.

In de derde kolom van Tabel 1 hebben we de overeenkomende processtappen uit het House changeproces vermeld. Hoewel onze analyse van het changeproces van House niet heel grondig is geweest, komen de door ons waargenomen activiteiten duidelijk overeen met de activiteiten genoemd in de IEEE standaard.

Het IEEE standaard definieert verder voor elke hoofdactiviteit input en output documenten. Ze gaan er dus vanuit dat het onderhoudsproces een overdracht is van informatie tussen verschillende disciplines.

5.1.2 Onderhoudsactiviteiten

Vessey gaat dieper in op de cognitieve activiteiten tijdens de ontwerp en implementatie activiteiten uit het onderhoudsproces (Vessey, 1986). Het uitvoeren van een onderhoudstaak zoals het ontwerpen en implementeren van een wijziging wordt door haar gezien als een probleemoplossende taak die uitgevoerd wordt door een enkel persoon.

Door gebruik te maken van think-aloud protocols bij ontwikkelaars tijdens het oplossen van bugs in een applicatie heeft Vessey de volgende cognitieve activiteiten geïdentificeerd:

- **Planning**
  Het zetten en evalueren van doelen en het bepalen en uitvoeren van een strategie om het probleem aan te pakken.

- **Knowledge building**
  Betekenis vormen van de applicatie en de onderhoudstaak. Deze betekenis wordt enerzijds gevormd door kennis welke de ontwikkelaar al heeft van de applicatie en de taak en anderzijds gevormd door gebruik te maken van beschikbare bronnen zoals de implementatie zelf en de documentatie.

- **Diagnosis**
  Het zoeken naar implementatie statements welke moeten worden aangepast. Dit is een proces waarbij een bepaalde aanname wordt gedaan over de betekenis van stukken implementatie. Deze aanname wordt getest en afgewezen of geaccepteerd.

- **Modification**
  Het aanbrengen van modificaties in de implementatie elementen en het testen van het resultaat.

Volgens Vessey lopen bovengenoemde activiteiten vaak door elkaar en is de volgorde en het aantal keer dat deze activiteiten worden uitgevoerd zeer afhankelijk van de capaciteiten en kennis van de ontwikkelaar en de onderhoudstaak. Wel wordt er verondersteld dat de activiteiten die betrekking hebben op het begrijpen, zoals planning, knowledge building en diagnosis, vóór het daadwerkelijk modificeren van de implementatie plaatsvinden.

In de theorie van Vessey wordt er gesproken over ontwikkelaars als de personen die de onderhoudstaak uitvoeren. Op de plekken waar ontwikkelaar staat hadden we echter net zo goed een ander persoon die onderhoudstaak uitvoert kunnen vermelden, aangezien de theorie van Vessey niet uitgaat van specifieke capaciteiten of kennis die enkele voor ontwikkelaars gelden. We zullen daarom vanaf nu de term onderhoudsmedewerker gebruiken in het geval we ieder die een onderhoudstaak uitvoert bedoelen.

5.1.3 Begrijpen van de applicatie

Veel literatuur stipt het begrijpen van de applicatie aan als belangrijke activiteit in het onderhoudsproces (Mayrhauser, 1995) (Vessey, 1986) (Letovsky, 1986b). Er zijn zelfs schattingen dat

zo'n 50% tot 90% van de tijd die wordt besteed aan onderhoud wordt besteed aan het begrijpen van de applicatie (Standish,1984).

Het begrijpen van de applicatie wordt gezien als het toekennen van betekenis aan de implementatie van een applicatie. Dit omhelst niet alleen begrijpen wat losse statements of groepen van statements doen, maar ook begrijpen van wanneer en in en welke volgorde deze statements worden aangeroepen en hoe en wanneer bepaalde transformaties op de data plaatsvinden (Pennington, 1987b).

Bij het toekennen van een betekenis aan de implementatie creëert de onderhoudsmedewerker een netwerk van mentale representaties, ook wel mentaal model genoemd, van de applicatie. Het vormen van een mentaal model van het probleem domein, in dit geval de applicatie, wordt vaak gezien als essentieel onderdeel voor correct oplossen van problemen. Foutieve oplossingen worden vaak in verband gebracht met een incompleet of incorrect gevormd mentaal model van het probleem domein.

Er bestaan veel theorieën over hoe dit mentale model wordt gevormd en uit welke representaties dit is opgebouwd (Robson, 1991). De meeste methodes gaan er vanuit dat een mentaal model van een applicatie bestaat uit een hiërarchie van verschillende representaties op verschillende abstractieniveaus. Representaties op het hoogste abstractieniveau zijn hierbij representaties van de functionaliteit van de applicatie of van concepten of regels in het business domein. Representatie op het laagste abstractieniveau zijn representaties van de elementen en de structuur van de implementatie (Mayrhauser,1995).

Bij het uitvoeren van een onderhoudstaak wordt de hiërarchie van representaties gebruikt om de te modificeren implementatie elementen te herkennen. Delen van deze modellen en de relaties daartussen zijn mogelijk al intern aanwezig bij de onderhoudsmedewerker. Zo heeft de oorspronkelijke programmeur van de applicatie vaak een goed beeld van de functionaliteit en de manier waarop deze is geïmplementeerd. Zijn mentale model van de applicatie zal dus representaties op alle abstractie niveaus bevatten. Een eindgebruiker welke alleen met de applicatie werkt, zal enkel een goed beeld hebben van de functionaliteit. Zijn mentale model van de applicatie zal dus veelal bestaan uit representaties op functioneel niveau en zijn representaties op lagere abstractie zijn vaak niet aanwezig.

Het vormen van een mentaal model van de applicatie die voldoende kennis bevat om de onderhoudstaak uit te voeren gaat op verschillende manieren en is afhankelijk van de kennis die een persoon al bezit en de kennis die voorhanden is. De kennis die een persoon al bezit worden interne representaties genoemd, de kennis die voorhanden is zoals code, requirements documenten of systeemdocumentatie worden externe representaties genoemd. Wanneer er onvoldoende interne representaties aanwezig zijn om de onderhoudstaak uit te voeren, moet die kennis worden vergaard uit de externe representaties, er moet dus code, modellen of documentaties worden gelezen en begrepen.

Voor het vergaren van deze kennis uit deze bronnen worden in het algemeen twee aanpakken onderscheiden (Mayrhauser,1995):

- **Top-down aanpak**
  Op basis van de bepaalde aannames over de werking van de applicatie (representaties op het hoogste niveau) word er gezocht naar abstracties op steeds lager niveau die deze werking verklaren.

- **Bottom-up aanpak**
  Bij een bottom-up aanpak worden er vanuit low level abstracties, zoals statements in de code, abstracties gemaakt op een steeds hoger niveau om zo uiteindelijk een compleet begrip te vormen van de applicatie of een deel daarvan.

Beide aanpakken lijken gecombineerd te worden toegepast bij het vormen van een mentaal model van de applicatie (Letovsky, 1986a).

Een belangrijk aspect in het begrijpen van de applicaties wordt het herkennen van *plans* genoemd (Letovsky, 1986a). Plans zijn patronen in de implementatie die een bepaalde functionaliteit of intentie van de code vertegenwoordigen. Voorbeeld van het herkennen van een plan is het herkennen van een sorting-algoritme op basis van bepaalde code statements in een for loop. Het sorting-algoritme wordt hierbij gezien als het plan van de onderliggende statements. Gevorderde ontwikkelaars hebben door hun ervaring een grote set van plans als interne representaties voorhanden, waardoor ze voor hun onbekende implementaties sneller kunnen doorgronden. Hiermee kan een gelijk worden getrokken met geoefende schakers welke intern over een grote hoeveelheid aan spelsituaties beschikken.

Verder is aangetoond dat zogenaamde *delocalized plans*, een negatief effect hebben op het vormen van een mentaal model van de applicatie (Letovsky, 1986b). Met een *delocalized plans* wordt in dit geval bedoeld dat implementatie elementen welke een plan vormen niet bij elkaar staan, maar verspreid over de implementatie. Het herkennen van een delocalized plan vergt meer tijd en ook capaciteiten van de onderhoudsmedewerker en bemoeilijk daarmee het vormen van een mentaal model van de applicatie.

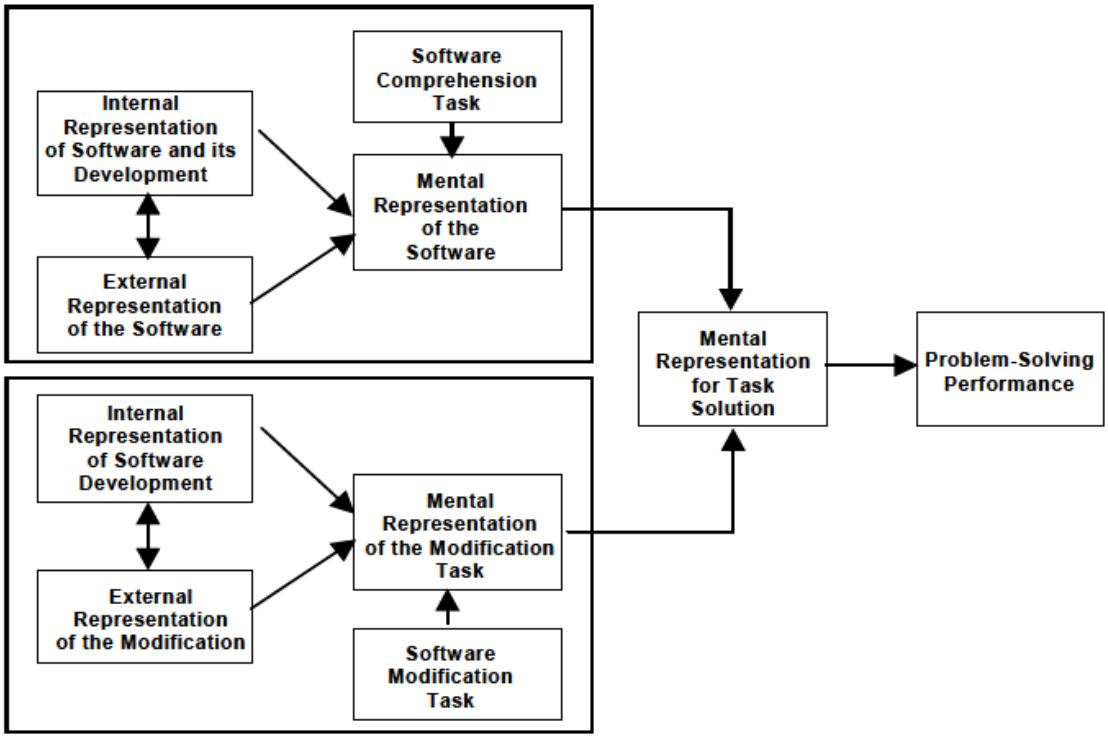
Hoewel de onderzochte literatuur gebaseerd is op onderzoek onder ontwikkelaars op low level programmeertalen, is er weinig reden om te geloven dat dezelfde cognitieve processen niet aanwezig zijn bij het onderhouden van hedendaagse BPM suite implementaties. Enkel de low level implementatie representaties van de applicatie zijn van een hoger abstractie niveau dan bijvoorbeeld applicaties welke geheel geschreven zijn in Java code. Doordat er minder verschil is tussen het hoogste abstractieniveau (representaties van de functionaliteit van de applicatie en het business domein)en laagste abstractieniveau (implementatie representaties), zou het vormen van een mentaal model van de applicatie waarschijnlijk wel minder kennis en capaciteiten vereisen bij BPM suite implementaties dan bij Java implementaties.

5.1.4 Cognitive fit

Begrijpen van applicatie en modifcatie: Dual-task problem solving model

Met het dual-task problem solving model (Figuur 5) hebben Shaft & Vessey de relatie onderzocht

tussen de onderhoudsperformance en de twee onderdelen in het onderhoudsproces, het begrijpen van het applicatie en de modificatie (Shaft&Vessey, 2006). Het model bestaat uit drie onderdelen, namelijk het vormen van een mentaal model van de applicatie, het vormen van een mentaal model van de onderhoudstaak en het vormen van een mentaal model van de onderhoudstaakoplossing. Hierbij wordt er wederom uitgegaan van het idee dat elk mentaal model gevormd wordt door aanwezige interne en een extern representaties op dat gebied. Bij het vormen van het mentale model van de taakoplossing wordt er uitgaan van het feit dat er geen externe representaties van de oplossing voorhanden zijn, dit model wordt dus puur gevormd door de andere twee modellen.
. Dit is eigenlijk logisch, als er een uitgeschreven oplossing van de onderhoudstaak was geweest, dan zou er geen probleem zijn dat moet worden opgelost.



Figuur 5: Dual-task problem solving model van Shaft & Vessey

Het mentale model van de applicatie wordt gevormd door de al aanwezig kennis van de applicatie als interne representatie en de aanwezige implementatie en documentatie wordt gezien als de externe representatie.

Het mentale model van de onderhoudstaak wordt gevormd door de al aanwezige kennis over software development (waaronder plans, kennis van implementatie talen) als interne representatie en de omschrijving van de onderhoudstaak wordt gezien als externe representatie.

De twee mentale modellen samen worden gebruikt door de onderhoudsmedewerker om een mentaal model te vormen van de taakoplossing. De onderhoudsmedewerker gebruikt dit mentale model om de uiteindelijke modificatie uit te voeren.

Shaft & Vessey gaan in hun experimenten uit van een versimpeld mentaal model van de applicatie. Het mentale model van de applicatie bestaat uit representaties van de functionaliteit, of representaties van de implementatie, of beide. Met behulp van het dual-task problem solving model heeft Vessey aangetoond dat wanneer de mentale representatie van de applicatie overeenkomt met de mentale representatie van de onderhoudstaak (cognitieve fit) dat er een positief effect is op het uitvoeren van de onderhoudstaak. Andersom is dit verband ook aangetoond, zo is er een negatief effect gemeten wanneer deze mentale representaties niet met elkaar overeenkomen (géén cognitieve fit). Deze verbanden zijn getest door de onderhoudstaak af te stemmen op de kennis van de onderhoudsmedewerker van de applicatie. Zo is er een positief effect gemeten wanneer er de onderhoudstaak een wijziging op control flow niveau beschreef en de onderhoudsmedewerker kennis had van het implementatie model en is er een negatief effect gemeten bij dezelfde onderhoudstaak wanneer deze onderhoudsmedewerker geen kennis had van het implementatie model van de applicatie. Eenzelfde experiment is er gedaan met een functionele wijziging als onderhoudstaak en het wel of niet aanwezig zijn van functionele kennis van de applicatie. Hierbij werden dezelfde effecten gemeten.

Het negatieve effect wordt door Shaft & Vessey toegeschreven aan het feit dat er tijdens het uitvoeren van de onderhoudstaak geen match is tussen het mentale model van de onderhoudstaak en de representaties in het mentale model van de applicatie. De onderhoudsmedewerker moet in dat geval tijdens het uitvoeren van de taak extra tijd steken in het vormen van representaties die wel aansluiten bij de onderhoudstaak of de applicatie wat een negatief effect heeft op de onderhoudsperformance.

Probleemoplossing en persoonlijke capaciteiten

Vessey heeft daarnaast een onderzoek gedaan naar de invloed van de capaciteiten van een persoon en het gebruik van ondersteunende technologie op zijn performance tijdens het oplossen van een probleem(Vessey, 1991). Met ondersteunende technologie werd in dit onderzoek het toepassen van ondersteunende weergaven van gegevens bedoeld, meer specifiek het gebruik van tabellen en grafieken. In een studie onder 128 MBA studenten werd iedere student twee verschillende opdrachten gegeven (de problemen). Deze opdrachten moesten worden opgelost met behulp van gegevens die weergegeven werden middels een tabel of een grafiek. Van te voren is bepaald welke opdracht het best past bij welke weergave (tabel of grafiek) en welke kwaliteiten de student hadden met betrekking tot het oplossen van de opdrachten.

Hieruit kwamen een aantal bevindingen naar voren:

- Wanneer een weergave wordt afgestemd op het type opdracht volgt er een groot positief effect op de performance van de uitvoerende student tijdens het uitvoeren van de opdracht.
- Wanneer de probleemoplossende kwaliteiten van de student overeenkwamen met het type opdracht en de weergave die werd uitgevoerd volgde er een gematigd positief effect op de performance van de uitvoerende student tijdens het uitvoeren van de opdracht.

- Wanneer de probleemoplossende kwaliteiten van de student overeenkwam met enkel het type opdracht of enkel de weergave werd er een zeer gering positief effect gemeten.

Dit leidde tot de conclusie dat met name het afstemmen van weergave van het probleem domein met de een bepaald taak die moet worden uitgevoerd zeer belangrijk is voor het snel en correct uitvoeren van deze taak (oplossen van het probleem).

5.1.5 Definities onderhoud op Aquima implementaties

In deze paragraaf zullen we een aantal definities opstellen die we zullen gebruiken tijdens onze analyse van de implementatie en de ontwikkelomgeving in paragraaf 5.3 en 5.4. Deze definities baseren we op onze bevindingen met betrekking tot het uitvoeren van onderhoud op de Aquima implementatie en de literatuur uit paragrafen 5.1.2 t/m 5.1.4.

Onderhoudstaak

Een onderhoudstaak is een taak die moet worden uitgevoerd om een bepaalde wijzigingsverzoek te implementeren. De RFC's en Customer Requirements die de business stakeholders op dit moment aanleveren aan de ontwerpers uit het Everest klantteam kunnen we zien als een omschrijving van de onderhoudstaak.

Deze documenten beschouwen we voor het gemak als uitgeschreven versie van het mentale model dat de business stakeholder heeft van de onderhoudstaak. Hoewel dit natuurlijk afhangt van schriftelijke uitdrukkingsvaardigheid van diegene die het document opstelt, is het aannemelijk dat de geschreven onderhoudstaak voor een groot deel overeenkomt met het mentale model dat de schrijver heeft van de onderhoudstaak. We gaan er in onze analyse dus vanuit dat het mentale model van de onderhoudstaak ongeveer gelijk is aan de beschrijvingen die we in de RFC's en Customer Requirements hebben aangetroffen.

In de documenten die we hebben onderzocht kunnen we meestal twee een bepaald *doelconcept* onderscheiden en een of meerdere *acties* die op dat doelconcept moeten worden uitgevoerd.

- **Doelconcept**
  Een doelconcept is het concept dat moet worden gewijzigd, zoals bepaalde functionaliteit van de applicatie, beoordelingsregel, business rule of een business concept.

- **Acties**
  Onder acties verstaan we een modificatie op het doelconcept of het toevoegen en verwijderen van een dergelijke concept.

We gaan er vanuit dat dit doelconcept en de acties onderdeel zijn van het mentale model van de onderhoudstaak.

Onderhoudsoplossing

Het mentale model van de oplossing beschouwen in deze context als mentale representaties van de modificaties op een of meerdere elementen uit de Aquima implementatie. Dit model wordt gedurende het uitvoeren van de onderhoudstaak gevormd door de onderhoudsmedewerker en steeds verder verfijnd.

Onderhoudsactiviteiten

Het vertalen van de onderhoudstaak naar de onderhoudsoplossing is in dit geval het probleem dat moet worden opgelost door de onderhoudsmedewerker. De stappen begrijpen en modificeren kunnen we in de context van de House casus als volgt definiëren:

- **Begrijpen**
  Men kiest op basis van een bepaalde zoekstrategie een element uit de implementatie en probeert deze te begrijpen (zoals een decision table). Deze zoekstrategie kan inhouden dat er wordt afgegaan op bekende namen van modules en elementen die verwant zijn aan het doelconcept, of men kan op basis van een bepaalde context zoals een pagina of een processtap proberen te navigeren naar onderliggende logica.
  Om het gekozen element te begrijpen wordt er gekeken naar elementen waar het element gebruik van maakt en de elementen die gebruik maken van dit element. De relaties van een element kun je op deze manier zien als een boomstructuur, waar men zowel omhoog als omlaag in gaat om de betekenis van een node (het element) te achterhalen. Ook van elementen die op deze manier worden bezocht wordt een betekenis gevormd. De zoektocht naar het juiste element wordt gestuurd op basis van de betekenissen die ter beschikking komen bij de onderhoudsmedewerker. Deze zoektocht duurt net zo lang totdat er een betekenis is gevormd uit een of meerdere elementen die overeenkomen met het doelconcept uit de onderhoudstaak.

- **Modificeren**
  Tijdens het uitvoeren van de onderhoudstaak vormt men een model van de onderhoudsoplossing. Hierbij worden acties op het doelconcept omgezet naar modificaties op implementatie niveau en de impact van deze modificaties wordt vervolgens geëvalueerd. De impact wordt bepaald door de elementen te bekijken welke direct en indirect afhankelijk zijn van de modificatie-elementen (dit zijn de elementen die gebruik maken van de modificatie-elementen). Ook deze elementen moeten worden begrepen om een representaties van de juiste oplossing te kunnen visualiseren. Uiteindelijk zullen de modificaties plaatsvinden als er genoeg vertrouwen is dat het mentale model van de oplossing de juiste is. Nadat de modificaties zijn uitgevoerd, zal de applicatie worden getest om te kijken of de oplossing daadwerkelijk het gewenste effect heeft gehad. Deze test wordt wederom opgebouwd op basis van de impact van de gemodificeerde elementen. Wanneer de test slaagt wordt de wijziging geaccepteerd, wanneer dit niet het geval is worden de *Begrijpen* en *Modificeren* stappen herhaald.

5.2 Business stakeholders

De kennis en capaciteiten van business stakeholders bepaalt voor een deel welke onderhoudstaken ze wel of niet uit kunnen voeren. In deze paragraaf proberen we daarom een beeld te schetsen van de kennis en capaciteiten die ze bezitten.

We hebben tijdens dit onderzoek helaas niet de mogelijkheid gehad om de geïdentificeerde business stakeholders te spreken. We baseren onze bevindingen en analyse daarom op de informatie uit de interviews en de change management documentatie.

Als we kijken naar het changeproces van House dan zijn de huidige business stakeholders enkel betrokken bij het specificeren van de onderhoudstaken (productmanagers, gebruikers) en het opstellen van functionele specificaties (business analisten). De stappen ontwerp tot met de oplevering van een wijziging worden afgehandeld door gespecialiseerde analisten, engineers en testers.

Wanneer we onderhoud van logica voor business stakeholders toegankelijk zouden willen maken betekent dit dat we de stappen van ontwerp tot oplevering op een dermate manier moeten ondersteunen dat business stakeholders deze stappen wel zelf kunnen uitvoeren.

Uit het dual task problem solving model in paragraaf 5.1.3 blijkt dat de performance van de ontwerp-, implementatie- en testactiviteiten grotendeels afhankelijk zijn van de al aanwezig kennis (interne representaties) van de applicatie op zowel het functionele als het implementatie vlak, en ervaring op het gebied van software ontwikkeling (plans, kennis van de implementatietaal etc).

In andere belangrijke factor is uiteraard kennis van het business domein. In de cognitieve modellen genoemd in de paragraaf 5.1 wordt deze kennis vaak niet als los kennisgebied beschouwd maar als onderdeel van het mentale model van de applicatie. Hoewel er een zeker overlap lijkt te zijn tussen kennis over de applicatie en het business domein waarin deze applicatie wordt gebruikt, is het volgens ons nuttig deze kennisgebieden apart van elkaar te zien. Een product manager welke niet met de applicatie werkt kan weinig kennis hebben van de applicatie maar toch veel kennis hebben van het business domein.

Samengevat zijn de volgende kennisgebieden volgens ons belangrijk bij het kunnen uitvoeren van onderhoud:

- Kennis van business domein
- Kennis van functioneel domein
- Kennis van implementatie domein
- Kennis op het gebied van software ontwikkeling

We hebben tijdens ons onderzoek drie verschillende business stakeholders geïdentificeerd welke betrokken zijn bij het changeproces van House: gebruikers, business analisten en productmanagers.

Tijdens de interviews hebben we gevraagd naar de kennis van de business stakeholders op elk van de deze kennisgebieden. De resultaten zullen we nu bespreken:

- **Kennis van het business domein**
  Aangezien het business stakeholders zijn hebben de gebruikers, business analisten, en productmanagers in het House project over het algemeen de meeste kennis van het business domein. We kunnen echter wel een onderscheid maken tussen specifieke kennis en brede kennis. Door hun rol als architect van de business beschikken de business analisten vaak over een brede kennis van het business domein op zowel procesvlak als op het gebied van beleid en producten. Gebruikers en productmanagers beschikken over meer specifieke kennis beschikken van dat deel van het business domein waar ze vanuit hun rol in organisatie mee te maken hebben. Zo heeft een gebruiker veel kennis van het proces waarvan hij deel uit maakt en hebben productmanagers veel kennis van de beoordelingsregels en configuraties van de verschillende hypotheekproducten.

- **Kennis van het functionele domein**
  Business analisten hebben over het algemeen een goed beeld van het functionele domein van de applicatie aangezien ze betrokken zijn bij het opstellen van functionele specificaties en acceptatietests. Gebruikers hebben op dit vlak weer meer specifieke kennis van het deel van de applicatie waarmee ze dagelijks werken. Productmanagers hebben weinig tot geen kennis van de functionaliteit van de applicatie aangezien ze vanuit hun positie niet vaak in aanraking komen met de House applicatie.

- **Kennis van implementatie domeint**
  Uit de onderzochte change management documenten en de interviews is gebleken dat geen van de huidige business stakeholders kennis heeft van het implementatie domein. In de onderzochte RFC's en requirements documenten werd in geen enkel geval een verwijzing naar de interne structuur van de applicatie gemaakt. Bij wijzigingen die betrekking hadden op het gedrag van de applicatie werd enkel gerefereerd naar waarneembare delen van de applicatie, bij wijzigingen in de business rules en business concepten werden er ook geen specifieke implementatie elementen genoemd, hoogstens de namen van parameters welke via de Product model module bekend waren geworden.

- **Kennis op het gebied van software ontwikkeling**
  Gezien het feit dat de business stakeholders op geen enkele manier betrokken zijn bij de implementatie van wijzigingen en het ontbreken van enige kennis over de implementatie ontbreekt ook kennis op dit vlak bij alle genoemde business stakeholders in het House onderhoudsproject.

Het is niet opmerkelijk te noemen dat kennis van het implementatie domein van de applicatie en kennis van software ontwikkeling ontbreken bij de huidige business stakeholders. Hun positie in het huidige changeproces en hun rol in de organisatie biedt ze daar ook geen aanleiding toe.

Het gebrek aan kennis op deze domeinen zal volgens het dual task problem solving model een negatieve uitwerking hebben op het uitvoeren van de onderhoudstaken, aangezien de business

stakeholders deze benodigde representaties tijdens het uitvoeren van het onderhoudstaak zullen moeten vormen. Het opbouwen van kennis op deze gebieden voordat ze onderhoudstaken gaan uitvoeren is dus waarschijnlijk noodzakelijk. Het is echter zeer de vraag of business stakeholders wel bereid zijn tijd te investeren in het opdoen van deze kennis. Het vergaren van deze kennis is niet enkel een kwestie van tijd, maar is ook een intensieve cognitieve taak. Het vormen van een mentaal model van de applicatie berust grotendeels op het vermogen om abstracties te kunnen vormen op verschillende niveaus. Deze capaciteiten zijn helaas vaak enkel weggelegd voor mensen met een hoog analytisch vermogen zoals technische stakeholders.

Hoe groot de bovengenoemde probleem echt zijn, hangt af van de specifieke onderhoudstaken die op dit moment plaatsvinden, de implementatie van de applicatie en de ondersteuning die de Aquima Studio biedt op dit gebied. In de volgende paragrafen zullen we verder in gaan op deze factoren.

5.3 Onderhoudstaken

Middels de House applicatie worden dagelijks grote aantallen hypotheekaanvragen ingediend en afgehandeld. Het is dan ook belangrijk dat de processen en schermen zoveel mogelijk stabiel blijven, grotere veranderingen zouden immers kunnen leiden tot verlies in productiviteit. De applicatie wordt wat dat betreft ook gezien als een volwassen applicatie, de processen en schermen zijn over de jaren geoptimaliseerd en grote functionele veranderingen zijn er vrijwel niet meer.

Voor een exacter beeld van de verschillende onderhoudstaken en de verhoudingen daartussen hebben we gekeken naar de change management documentatie en geput uit de informatie die naar boven kwam tijdens de interviews.

In de documentatie kwamen we uiteenlopende wijzigingsverzoeken tegen. Naast enkele grote wijzigingen met betrekking tot koppelingen en nieuwe functionaliteit, kan het grootste deel van de wijzigingsverzoeken als volgt worden ingedeeld:

- **Wijzigingen in business concepten of business rules**
  Dit zijn wijzigingsverzoeken welke veranderingen in beleid beschrijven die moeten worden doorgevoerd in het systeem. Bijvoorbeeld:

  “Vanaf nu is de executiewaarde van de bestaande woning gelijk aan 69% van de WOZ waarde als de bron van de waardebepaling gelijk is aan WOZ”

- **Wijzigingen in het waarneembare gedrag van de applicatie**
  Dit zijn wijzigingsverzoeken welke een functionele verandering beschrijven in de schermen en documenten. Vaak hebben deze wijzigingsverzoeken betrekking op een verandering in de huidige manier van werken. Bijvoorbeeld:

  “Als behandelaar wil ik meer keuze hebben uit standaard afwijsredenen als ik een voorbehoud afwijs.“

  Of

  “Bij elke volgende beoordeling dienen de overrule velden te worden getoond met de laatst ingevulde data. Uitsluitend als de beoordelingsregel risico verhogend is gewijzigd, dient het veld met de overrule reden blanco te worden weergegeven”

- **Wijzigingen in zowel het waarneembare gedrag van de applicatie als in de business rules en business concepten.**
  Dit zijn wijzigingsverzoeken welke een verandering beschrijven in de business concepten of business rules gekoppeld aan een actie in de front-end. Bijvoorbeeld:

  “De aanvraag valt in de beoordeling uit met als reden: (“Gevraagde lening is meer dan

volgens het beleid is toegestaan, er is geen taxatierapport aanwezig") als de gevraagde netto bevoorschotting voor een te bouwen woning> 115% executiewaarde en er is geen taxatierapport aanwezig"

De type wijzigingen kunnen vaak gerelateerd worden aan de indienende business stakeholders. Zo zijn wijzigingen afkomstig van gebruikers zoals behandelaren vaak wijzigingen in het waarneembare gedrag van de applicatie, dus meer op functioneel gebied. Terwijl wijzigingen die betrekking hebben op business rules en business concepten vaak afkomstig zijn van business stakeholders welke niet direct gebruik maken van de applicatie, zoals productmanagers.

Een opvallend soort wijzigingen uit de laatst genoemde categorie zijn de beoordelingsregels. Beoordelingsregels zijn een set van regels voor het beoordelen van een aanvraag. Deze regels zijn ondergebracht in de module Beoordeling (zie paragraaf 3.2). Beoordelingsregels zijn een essentieel onderdeel in het hypotheekaanvraagproces, ze bepalen uiteindelijk de status van een hypotheekaanvraag. De regels zijn opvallend omdat ze een soort tussenvorm zijn van regels op functioneel niveau en business rules. Het combineert een regel uit het beleid met een actie in het systeem.

Deze regels worden volgens een vast patroon gedefinieerd door de business stakeholders:

*Een aanvraag moet **<status>** krijgen als **<voorwaarden>** met de melding **<melding>***

Een simpel voorbeeld van een instantie van zo'n regel is:

Een aanvraag moet geweigerd worden als BKR registratie negatief is met de melding "Aanvrager heeft een negatieve BKR registratie, aanvraag geweigerd".

Het grootste deel van de wijzigingen in het changeproces van House en in de door ons onderzochte documentatie bestond uit wijzigingen in beoordelingsregels. Er is op dit moment dan ook een grote behoefte onder EMS klanten om deze regels zelf te kunnen onderhouden. Naast beoordelingsregels komen ook regelmatig wijzigingen voor die betrekking hebben op de business rules en business concepten. Functionele wijzigingen die gericht zijn op veranderingen in de waarneembare delen van de applicatie blijken van de drie categorieën het minst voor te komen.

In paragraaf 5.4 zullen we, door te kijken naar de huidige implementatie van de House applicatie, dieper ingaan op kennis en capaciteiten die nodig zijn om de bovengenoemde onderhoudstaken uit te kunnen voeren.

5.4 Implementatie

In deze paragraaf zullen we de implementatie van de House applicatie analyseren.
Eerst gaan we in op de algemene eigenschappen van de implementatie en vervolgens zullen we de logische elementen bekijken waar de onderhoudstaken uit paragraaf 5.3 betrekking op hebben. Per onderhoudstaak zullen we teven de gevolgen op de onderhoudsmogelijkheden voor business stakeholders beschrijven.

5.4.1 Algemene eigenschappen

De House implementatie is in onze ogen een grote implementatie. Zo bestaat het interactiemodel uit meer dan 150 pagina's en bevat het datamodel meer dan 200 entiteiten. Het grootste deel van de implementatie bestaat uit logica, er zijn meer dan 1000 logische elementen waarvan het grootste deel geïmplementeerd middels decision tables. De House implementatie wordt door medewerkers van Everest gezien als de meeste complexe Aquima implementatie die is gerealiseerd. Uit de interviews kwam naar voren dat het minimaal een jaar duurt voordat nieuwe engineers in het klantteam van Everest voldoende kennis hebben opgebouwd van de implementatie en de functionaliteit van de applicatie om de meeste onderhoudstaken te kunnen uitvoeren.

De complexiteit en omvang van implementatie heeft onder andere te maken met de hoeveelheid informatie waaruit een hypotheekaanvraag bestaat en de grote hoeveelheid regels waaraan hypotheekproducten zijn gebonden. Daarnaast beschikt de House applicatie een geavanceerde front-end om de aanvragen zo efficiënt mogelijk te kunnen registreren. Deze efficiënte wordt onder andere bereikt door deze schermen van de applicatie zo dynamisch mogelijk te maken. Er wordt dus in verhouding veel logica gebruikt voor het realiseren van zowel de interactie als voor het implementeren van de business rules en business concepten.

In paragraaf 5.3 hebben we gezien dat de onderhoudstaken in de House casus met name betrekking hebben op de front-end en de business concepten en business rules. Een speciale rol spelen verder de beoordelingsregels welke gespecificeerd werden als een combinatie van deze twee onderdelen. We zullen nu de kennis en capaciteiten die nodig zijn voor het uitvoeren van deze onderhoudstaken analyseren.

5.4.2 Toelichting analyse onderhoudstaken

Intuïtief gezien heeft het uitvoeren van onderhoudstaken door business stakeholders het meeste kans van slagen als er zo min mogelijk capaciteiten en nieuwe kennis van de business stakeholders worden gevraagd.

In paragraaf 5.2 hebben we geconstateerd dat het mentale model van de applicatie bij business stakeholders over het algemeen geen representaties bevat op implementatie niveau, maar dat hun mentale model van de applicatie hoogstens bestaat uit representaties op functioneel niveau.

Wanneer we business stakeholders onderhoudstaken willen laten uitvoeren moeten ze de representaties op implementatie niveau opbouwen door betekenissen te vormen van de implementatie elementen. Daarnaast hebben we vastgesteld dat de business stakeholders veel

kennis hebben van het business domein, ook deze beschikbare kennis beschouwen we als een mentaal model. Het mentale model van de functionaliteit van de applicatie en het mentale model van het business domein kunnen beide helpen bij het vormen van betekenissen en herkennen van implementatie elementen. Zo zal de herkenning van elementen sneller gaan wanneer ze overeenkomsten vertonen met representaties uit de mentale modellen van de business stakeholders. Volgens het dual task problem solving model van Shaft & Vessey gaat het uitvoeren van de onderhoudstaak het best wanneer het uiteindelijk gevormde mentale model van de applicatie representaties bevat die overeenkomen met representaties uit het mentale model van de onderhoudstaak (in ons geval baseren we ons op de omschrijving van deze taak in de RFC's en requirements documenten).

In paragraaf 5.1.5 hebben we beschreven hoe deze onderhoudsactiviteiten zouden plaatsvinden bij het uitvoeren van een onderhoudstaak op een Aquima implementatie. Als we kijken naar de stappen die worden uitgevoerd dan kunnen we concluderen dat de manier waarop bepaalde logica is geïmplementeerd van grote invloed is op de kennis en capaciteiten die voor het uitvoeren van een bepaalde onderhoudstaak nodig zijn. In het bijzonder lijken de volgende twee factoren een belangrijke rol te spelen:

- **Impact van de wijziging op de implementatie**
  Het aantal te begrijpen implementatie elementen bepaald de hoeveelheid kennis die de business stakeholder moet vergaren of bezitten. Wanneer er voor de implementatie van een wijziging veel elementen moeten worden gemodificeerd of wanneer deze modificaties op zichzelf veel invloed hebben op andere elementen dan moet de onderhoudsmedewerker veel elementen begrijpen voor het correcte uitvoeren van een onderhoudstaak. Hoe kleiner de impact van een wijziging hoe minder elementen er begrepen moet worden, en dus hoe minder capaciteiten dit vereist van de onderhoudsmedewerker.

- **Overeenkomst met mentale modellen business stakeholders**
  Implementatie elementen kunnen sneller worden begrepen als ze qua naamgeving en compositie overeenkomen met concepten uit voor de onderhoudsmedewerker bekende domeinen. De elementen die moeten worden begrepen moeten dus aansluiten bij de representaties in de bij de onderhoudsmedewerker aanwezige mentale modellen van de functionaliteit van de applicatie en het business domein en de onderhoudstaak. Doordat de implementatie elementen kunnen worden gerelateerd aan representaties uit deze mentale modellen kunnen betekenissen van deze elementen worden gevormd zonder de onderliggende elementen te bekijken (business concepten kunnen worden herkent zonder bijvoorbeeld de onderliggende afleiding te begrijpen of te bekijken). Dit zorgt ervoor dat de elementen welke gemodificeerd dienen te worden sneller kunnen worden herkent en dat de impact van een wijziging sneller kan worden bepaald.

Hoewel we ons realiseren dat bovenstaande factoren niet allesomvattend zijn, zijn ze volgens ons voldoende bruikbaar voor het maken van een algemene inschatting over de kennis en capaciteiten die nodig zijn bij het uitvoeren van een onderhoudstaak. We zullen nu de verschillende onderhoudstaken welke we hebben geïdentificeerd in paragraaf 5.3 beoordelen aan de hand van deze twee factoren.

5.4.3 Wijzigingen in business concepten of business rules

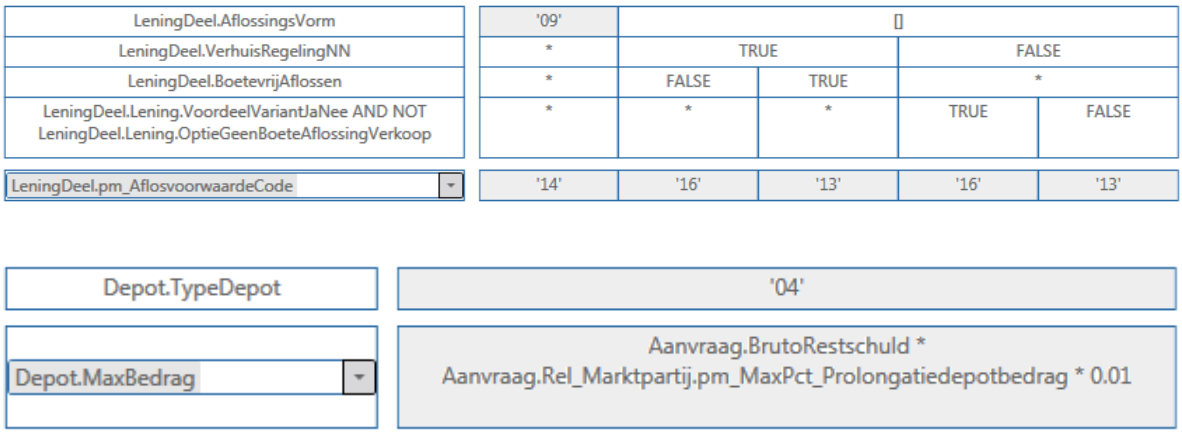
Overeenkomst met mentale modellen business stakeholders

Door business logica te scheiden van applicatie logica kan deze logica op zo'n manier worden geïsoleerd dat implementatie elementen enkel het business domein beschrijven. Daarnaast geeft het isoleren van business logica je de ruimte om de logica op zo'n manier te ontwerpen dat het ook qua compositie en naamgeving aansluit op de manier waarop business concepten en business rules in de natuurlijke taal worden gespecificeerd. Deze aspecten zijn beide belangrijk wanneer je de business logica wilt aansluiten bij de mentale modellen van de business stakeholders.

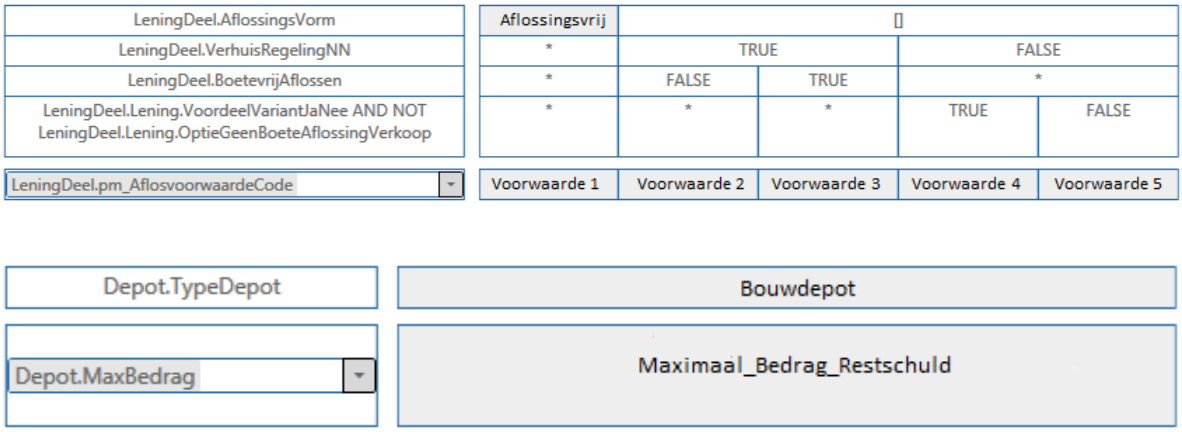
In de implementatie van House zijn de business concepten en business rules veelal middels modules gescheiden van de rest van de logica in de applicatie. Zo worden er voor resultaten van logica begrijpbare attributen aangemaakt in het datamodel. Een berekening voor het brute inkomen van een persoon, resulteert bijvoorbeeld in een waarde in het attribuut 'BrutoInkomen' van entiteit 'Persoon'. Hierdoor kan er in andere logische elementen gebruik gemaakt worden van dit attribuut, zonder dat de berekening van het bruto inkomen steeds opnieuw moet worden gedefinieerd. Dit voortkomt dat dit soort business logica op verschillende plekken wordt gemengd met applicatie logica. In die zin beschrijven de business logica elementen puur de concepten uit het business domein. Voor het begrijpen van deze elementen is dus vrijwel geen kennis van het ontwerp van de rest van de applicatie nodig. Ondanks deze scheiding lijkt de implementatie van de business logica op het gebied van notatie en compositie echter nog vaak niet goed aan te sluiten bij de manier waarop ze worden beschreven in de RFC's en Customer Requirements. De implementatie van de business logica lijkt dus niet helemaal aan te sluiten bij het mentale model van het business domein van de business stakeholders.

Zo worden er veel nummers of codes gebruikt in plaats van namen van de corresponderende business concepten en worden er ook regelmatig berekeningen en afleidingen gevonden waar in de natuurlijke taal een concept voor gebruikt wordt (zie figuur 6). Dit maakt het lezen en aanpassen van dit soort business rules en business concepten nog steeds lastig aangezien de business stakeholder een nieuwe compositie en notatie moet aanleren.

Dit is eigenlijk onnodig aangezien de implementatie elementen van het Aquima platform voldoende ondersteuning bieden om de implementatie van de business concepten en business rules af te stemmen met de beschrijvingen in de natuurlijke taal. Zo kunnen op bestaande logische elementen op een eenvoudige manier 'business-vriendelijker' gemaakt worden door de genoemde codes en afleidingen onder te brengen in herbruikbare expressies welke qua naamgeving aansluiten op de corresponderende concepten in de natuurlijke taal (zie figuur 7).



Figuur 6: Gebruik van implementatie codes en berekeningen in business logica



Figuur 7: Aangepaste implementatie waarbij codes en berekeningen zijn vervangen voor herbruikbare expressies.

De transformatie in figuur 7 zorgt voor een beter aansluiting bij het business domein. Business stakeholders zijn beter bekend met dit concepten uit dit domein en zullen dus de logica uit figuur 7 makkelijker kunnen begrijpen.

De onderste decision table uit figuur 7 geeft aan hoe belangrijk het is om de compositie van business logica af te stemmen op de zogenaamde betekenisstructuur uit de natuurlijke taal. In communicatie middels natuurlijke taal maken we veelvuldig gebruik van concepten zonder daarbij de onderliggende betekenis steeds uit te hoeven leggen. Dit is niet alleen efficiënt in het geval de ontvangende partij de betekenis toch al weet, maar ook pragmatisch. Vaak weet men niet zo goed de onderliggende betekenis of doet de exacte betekenis er niet toe. Een business stakeholder zou bijvoorbeeld niet zo snel de berekening voor het maximale bedrag van de restschuld van een aanvraag voorhanden hebben (zie voorbeeld rechts onderin in figuur 6), maar zal zonder veel moeite een beleidsregel kunnen opstellen die op een goede manier gebruik maakt van dit business concept.

In een implementatie van deze concepten is het uiteraard noodzakelijk dat deze betekenis op een exacte manier wordt vastgelegd, wanneer hier echter niet dezelfde betekenisstructuur wordt aangehouden als in de natuurlijke taal dan zou een business stakeholder bij een wijziging gedwongen worden om een bepaalde berekening iedere keer weer op een exacte manier te specificeren. Omgekeerd moet de business stakeholder uit alle berekeningen een betekenis afleiden en deze relateren aan een voor hem bekend business concept, wat het begrijpen van de implementatie vertraagd en bemoeilijkt.

De reden om deze betekenisstructuur niet aan te houden bij de compositie van de business logica, heeft met name te maken met tijd en kosten. Voor een complex business domein zoals het hypotheek domein is het relatief kostbaar om op deze manier een business logica implementatie te realiseren. Daarnaast moeten deze extra inspanningen ook voor een deel gedurende de onderhoudsfase van de applicatie worden geleverd, bijvoorbeeld wanneer er nieuwe business concepten en business rules worden geïntroduceerd.

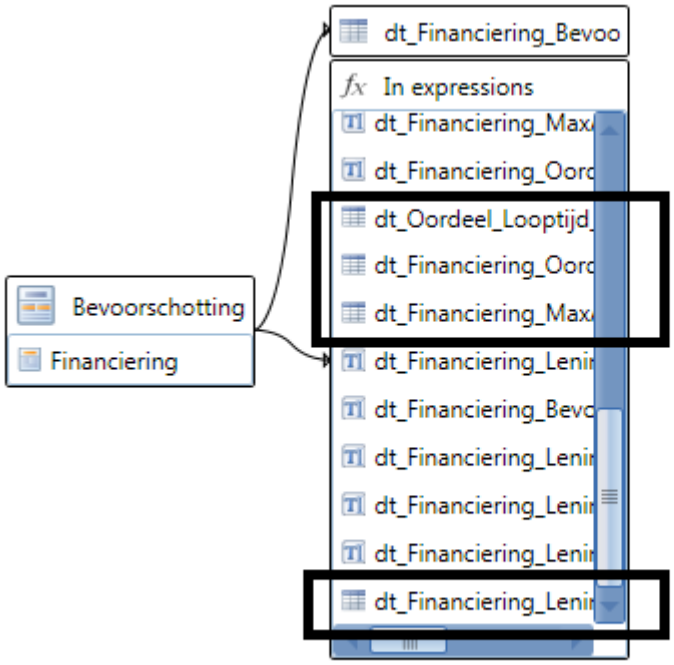
Software ontwikkeling en onderhoud op software hebben echter vaak te maken met enorme tijdsdruk, een korte time-to-market is steeds belangrijker gezien de techniek steeds meer de toegevoegde waarde van een onderneming bepaald. Deze tijdsdruk zorgt ervoor dat bovengenoemde verbeteringen aan het ontwerp van de business logica, helaas vaak maar moeilijk te verkopen zijn aan het projectmanagement.

Impact van de wijziging op de implementatie

Hoewel in de vorige paragraaf is gebleken dat de structuur van de logica niet altijd is afgestemd op de structuur van deze business concepten en business regels in de natuurlijke taal, geldt dit met name voor de implementatie van elementaire business concepten en berekeningen welke minder vaak worden gewijzigd. Deze business concepten en business regels die het vaakst wijzigen zijn vaak geïmplementeerd middels een enkel decision table. Voor wijzigingen in business concepten en business rules hebben de modificaties meestal maar betrekking op een enkel implementatie element.

Het aantal elementen wat begrepen moeten worden voor het bepalen van de impact van de modificaties is echter potentieel veel groter. Onder andere de dynamisch front-end van de House applicatie zorgt ervoor dat er vanuit de interactie elementen veel business logica elementen worden aangeroepen. Hierdoor ontstaat er een grote afhankelijkheid tussen de business logica en de rest van de implementatie.

Voor de logica elementen die business rules implementeren valt deze impact echter nog mee, ze worden vaak maar op een enkele plaatsen in de applicatie aangeroepen. Daarnaast zijn er vaak geen hogere samengestelde business concepten die gebruik maken van deze business rules. Wanneer het echter om business logica elementen gaat die afleidingen en concepten representeren die elementairder zijn, zoals berekeningen, is de impact van een wijziging een stuk groter.



Figuur 8: Afhankelijkheidsschema van een beslistabel welke een berekening implementeert

In figuur 8 is middels de dependancy viewer van Aquima Studio te zien dat het resultaat van de decision table voor het berekenen van de bevoorschotting van een hypotheek, door vier andere decision tables wordt gebruikt. Deze vier gerelateerde beslistabellen worden daarnaast mogelijk weer in andere elementen zoals interactie elementen gebruikt. Het analyseren en testen van een modificatie in deze berekening is problematisch voor business stakeholders. Niet alleen moeten er veel gerelateerde elementen worden begrepen, er is al heel snel kennis nodig van de interne werking van de applicatie. Een business logica element kan worden aangeroepen door een logisch element wat niet meer een concept uit het business domein beschrijft, maar bijvoorbeeld bepaalt welke webservice er moeten worden aangeroepen of welk tekstblok er moet worden getoond in een document.

Het gebrek aan kennis van de implementatie en ervaring op het gebied van software ontwikkeling, maakt het evalueren en testen van een onderhoudsoplossing bij dit soort onderhoudstaken zeer lastig voor business stakeholders.

5.4.4 Wijzigingen in het waarneembare gedrag van de applicatie

Overeenkomst met mentale modellen business stakeholders

In onderhoudstaken die betrekking hebben op een wijziging in het waarneembare gedrag van de applicatie wordt er meestal gerefereerd naar de 'output' van de applicatie zoals schermen, documenten, knoppen, invulvelden, tekstblokken, etc.

Het doelconcept is in dit geval het gedrag waarnaar verwezen wordt. De logica die dit gedrag realiseert is vaak gekoppeld aan de implementatie elementen die deze output realiseren. Dus in

tegenstelling tot wijzigingen in business logica waarbij het doelconcept van een onderhoudstaak vaak overeenkomt met een enkel implementatie element, hebben wijzigingen in het waarneembare gedrag van de applicatie vaak betrekking op meerdere implementatie elementen van verschillende types. De wijziging genoemd in paragraaf 5.3 is hier een goed voorbeeld van:

"Bij elke volgende beoordeling dienen de overrule velden te worden getoond met de laatst ingevulde data. Uitsluitend als de beoordelingsregel risico verhogend is gewijzigd, dient het veld met de overrule reden blanco te worden weergegeven"

De logica welke bovenstaand gedrag moet realiseren bevindt zich niet in een enkel logisch element maar is verspreid over verschillende logische elementen gekoppeld aan de interactie elementen die de overrule velden representeren. Daarnaast zal dit gedrag niet gerealiseerd kunnen worden zonder extra functionaliteit toe te voegen voor het opslaan en ophalen van deze overrule gegevens uit de laatst ingevulde beoordeling. Bovenstaande acties zijn niet zonder veel kennis van het implementatie domein te bedenken. Het mentale model van de onderhoudstaak ligt in die zin vaak ver af van het mentale model van de onderhoudsoplossing.

Het bovenstaande voorbeeld is daarnaast geen uitzondering in de House casus. Er zijn echter wel simpelere voorbeelden te bedenken zoals de onderstaande onderhoudstaak:

"Als veld A is ingevuld moet er bij veld B niet optie 3 maar voortaan optie 4 worden voorgeselecteerd."

Bovenstaande voorbeeld zal in veel applicaties als wijzigingsverzoek kunnen voorkomen. Wanneer men echter op zoek zal gaan naar het logische element in de applicatie welke verantwoordelijk is voor deze functionaliteit, zal men moet begrijpen hoe dit soort logica is geïmplementeerd:

- Hangt er een actie aan veld A welke zorgt voor het voorselecteren van een waarde uit veld B?
- Is er een logische conditie in veld B welke bepaalt welke waarde er voorgeselecteerd moet worden?
- Is er een algemeen stukje logica welke automatisch aangeroepen wordt als de waarde van veld A in een bepaald attribuut van het entiteiten model wordt geplaatst?

Bovenstaande drie vragen illustreren dat zelfs bij een simpele onderhoudstaak iemand vele ideeën kan hebben bij hoe iets geïmplementeerd is en hoe de implementatietaal werkt. Er moet bij het snel en correct uitvoeren van dit type onderhoudstaken daarom dus kennis aanwezig zijn van het ontwerp van de implementatie en de eigenschappen van de implementatietaal. Bij het onderhouden van business logica is dit veel minder het geval omdat deze logica in het ontwerp vaak veel meer op zichzelf staat en een domein beschrijft waar de business stakeholders al kennis van hebben.

Impact van de wijziging op de implementatie

Bij wijzigingen in het waarneembare gedrag van de applicatie is het aantal elementen wat gemodificeerd moeten worden al snel groter aangezien het gedrag als doelconcept door meerdere

implementatie elementen wordt bewerkstelligd.

De impact van een wijziging in het waarneembare gedrag van de applicatie is daarnaast potentieel nog hoger dan bij wijzigingen in de business logica aangezien er vaak meerdere elementen moeten worden gemodificeerd, welke allemaal weer hun eigen impact hebben op zowel de business logica elementen als andere elementen van de applicatie.

Bij business logica kan het herkennen van de te modificeren elementen en het bepalen van de impact makkelijker worden gemaakt door de compositie en naamgeving van de logica aan te sluiten bij de betekenisstructuur en notatie van business rules en business concepten uit de natuurlijke taal. Dit is bij onderhoudstaken die betrekken hebben op het waarneembare gedrag lastiger. De manier waarop logica kan worden aangeroepen vanuit interactie elementen en de manier waarop deze logica de weergave van interactie elementen kan beïnvloeden wordt bepaald door het type modelleer elementen van het Aquima platform. Dit maakt het moeilijk om de implementatie van bepaald gedrag van de applicatie aan te sluiten bij het mentale model wat de business stakeholders heeft van de onderhoudstaak.

Door minder logica in bijvoorbeeld de interactie te gebruiken zouden aanpassingen op het waarneembare gedrag van de applicatie wel makkelijker kunnen worden gemaakt, aangezien dit de mogelijke impact van wijzigingen zou kunnen beperken. Dit zou echter weer gevolgen hebben op de efficiëntie van de interactie en dus vaak niet wenselijk zijn.

5.4.5 Wijzigingen in zowel het waarneembare gedrag van de applicatie als in de business rules en business concepten

Wanneer de business logica goed gescheiden is van de applicatie logica die de wijziging in het waarneembare gedrag voor zijn rekening neemt, kunnen de wijzigingen in deze categorie beschouwd worden als twee losse onderhoudstaken, één wijziging in de business rules en business concepten en één wijziging met betrekking tot het waarneembare gedrag van de applicatie. Deze twee onderhoudstaken kunnen vervolgens op dezelfde manier geanalyseerd worden als in paragrafen 5.4.3 en 5.4.4.

Bij de implementatie van de logica voor beoordelingsregels is echter geen duidelijke scheiding gemaakt tussen de onderliggende business logica en de applicatie logica welke de actie realiseren. We zullen daarom enkel het onderhouden van deze beoordelingsregels als onderhoudstaak analyseren in de volgende paragraaf.

5.4.6 Wijzigingen in beoordelingsregels

Overeenkomst met mentale modellen onderhoudsmedewerker

In de House implementatie is er voor gekozen om elke beoordelingsregel te implementeren middels één implementatie element voor de logica en één of meerdere parameters voor de melding. Dit implementatie element komt qua naamgeving ook overeen met het deel van de aanvraag die het beoordeeld. Dus op dit niveau sluit het aan bij het doelconcept in het mentale model van de

onderhoudstaak. Kijken we echter naar de implementatie van de regels zelf, dan zijn er wel degelijk verschillen op dit niveau.

Aanvraag.Rel_Marktpartij.pm_MaximumAantalRekeningen !=?	TRUE		FALSE
SIZE (COLLECT LeningDeel FROM ALL LeningDeel WHERE (LeningDeel.pm_NieuweRekeningInSAS !=? AND LeningDeel.pm_NieuweRekeningInSAS)) > Aanvraag.Rel_Marktpartij.pm_MaximumAantalRekeningen	TRUE	FALSE	*

Aanvraag.Oordeel_AantalRekeningen	'Afwijzen'	'Accepteren'	'Accepteren'
Aanvraag.Oordeel_AantalRekeningen_Code	'AANVRAAG_AFW_AANTALREKENINGEN'	'OK'	'OK'
Aanvraag.Oordeel_AantalRekeningen_Score	SIZE (COLLECT LeningDeel FROM ALL LeningDeel WHERE (LeningDeel.pm_NieuweRekeningInSAS !=? AND LeningDeel.pm_NieuweRekeningInSAS))	0	0
Aanvraag.Oordeel_AantalRekeningen_Cat	'Overig'	'Overig'	'Overig'

Figuur 9: Decision table van een beoordelingsregel

In de implementatie van de beoordelingsregel in figuur 9 is te zien dat de logica in de decision table zowel de beoordeling van de hypotheek aanvraag als de actie afleidt. De af te leiden attributen voor de actie zijn hierbij standaard voor alle beoordelingsregels (zie tabel 2).

Attribuut	**Omschrijving**
code (.._Code),	De code van de melding. Deze code is een parameter in de Product model module. Een aanpassing in de tekst van de melding moet dus in het Product model gedaan worden.
score (…_Score),	Het score attribuut wordt gebruikt als numerieke waarde voor weergave in de melding (de betekenis van deze score kan van alles zijn, het wordt gebruikt als een generieke variabele)
categorie (…_Cat),	De categorie waaronder de melding in de front-end moet worden weergegeven.

Tabel 2: Afspraken over standaard af te leiden attributen bij beoordelingsregels

De attributen in tabel 2 vormen de interface voor beoordelingsregels en zijn onderdeel van het interne ontwerp van een dergelijke regel. Dit is problematisch voor business stakeholders aangezien ze dus over kennis van deze interface en het interne ontwerp van de regels moeten beschikken om dit soort onderhoudstaken uit te voeren. Dit is een gevolg van het niet goed scheiden van de business logica van de applicatie logica.

De implementatie van beoordelingsregels zou op dit vlak verbeterd kunnen worden door voor elke

beoordelingsregel twee desicion tables te gebruiken, waarvan er één enkel de business logica bevat en de ander de business logica aanroept en op basis hiervan de attributen voor de actie afleidt. Dit heeft echter wel het gevolg dat er voor sommige wijzigingen op het gebied van beoordelingsregels twee implementatie elementen moeten worden aangepast.

Als we verder kijken naar enkel implementatie van de business logica van beoordelingsregels dan gelden ook hier de opmerkingen met betrekking tot de compositie en notatie welke in paragraaf 5.4.2 zijn benoemd.

Impact van de wijziging op de implementatie

Doordat elke beoordelingsregel middels één decision table element is geïmplementeerd, hebben de modificaties die onderdeel zijn van een onderhoudstaak vaak maar betrekking op een enkel element. De beoordelingsregels worden daarnaast maar op één plaats aangeroepen in de House applicatie, namelijk bij het beoordelen van de een aanvraag.

De impact van een wijziging in een beoordelingsregels op de implementatie is dus zeer klein. De beoordelingsregels zijn echter wel onderdeel van een kritische beslissing in het hypotheekaanvraagproces. Een foutief aangepaste of geïmplementeerde beoordelingsregel kan voor zorgen dat er voor een hypotheekaanvraag een verkeerde status wordt afgeleid. Wanneer een beoordelingsregel een bepaald financieel risico moet afdekken, kunnen dit soort fouten leiden tot grote financiële gevolgen voor de bank.

5.5 Ontwikkelomgeving

Niet alleen de implementatie maar ook de ontwikkelomgeving en de implementatietaal kan bijdragen aan het onderhoudsgemak van een applicatie.

Zo kan een implementatietaal voorzien in functiebibliotheken voor bepaalde veel gebruikte expressies. Door gebruik te maken van deze functies hoeft er minder code te worden geschreven wat uiteindelijk de onderhoudbaarheid van deze code ten goede komt. Ook kan de syntax van een taal bijdragen aan het snel kunnen schrijven maar ook begrijpen van expressies.

Een ontwikkelomgeving kan het onderhouden van software ondersteunen door de code van de applicatie op een bepaalde manier te representeren. Daarnaast kan het ondersteuning bieden voor veel voorkomende handelingen die op de code moeten worden uitgevoerd, zoals bijvoorbeeld zoek- en refactoringsmogelijkheden.

In deze paragraaf zullen we onderzoeken in hoeverre Aquima Studio (de ontwikkelomgeving en de Aquima implementatietaal) bijdraagt aan de onderhoudbaarheid van de logica.

We zullen de bijdrage aan de onderhoudbaarheid onderzoeken door Aquima Studio functionaliteit te toetsen op deze volgende drie gebieden: representatie, analyse en modificatie.

5.5.1 Representatie

Met representatie functionaliteit bedoelen we hoe een ontwikkelomgeving de implementatie presenteert aan de onderhoudsmedewerker.

Voor onderhoudstaken is een passende representatie nuttig om de snel betekenissen van elementen te kunnen afleiden.

Als we kijken naar de representaties van logica van Aquima Studio, dan kan logica worden geïmplementeerd middels vier verschillende type implementatie elementen welke ieder hun eigen representatie hebben op element niveau. Binnen deze representaties wordt er gebruikt gemaakt van de Aquima expressietaal. We zullen eerst de representaties van deze elementtypes bekijken en vervolgens de syntax van de expressietaal behandelen.

- **Expressions**
  Simpele berekeningen welke geen condities bevatten kunnen in expressies worden uitgedrukt. Omdat er is weinig visuele ondersteuning nodig is om deze logica te doorgronden wordt enkel een vrij invulveld getoond waarbij de invoer moet voldoen aan de syntax van de expressietaal van Aquima.

- **Business rules**
  Hoewel de naam doet vermoeden dat hier alle soorten business rules in kunnen worden gedefinieerd, zijn business rules als element type van Aquima Studio bedoelt voor het implementeren van simpele afleidingen waarbij er een waarde voor een enkel attribuut

wordt afgeleid. In de representatie wordt een template gepresenteerd van een 'ALS..DAN' statement waarbij voor de conditie en de af te leidde waarde een expressie kan worden ingevuld. Het attribuut dat wordt afgeleid kan worden gekozen middels keuzelijsten.

- **Decision tables**
  Complexere logica met meerdere afleidingen en meerdere condities kunnen middels desicion tables worden geïmplementeerd, hierbij wordt een tabel representatie gebruikt waarin de cellen expressies bevatten en er de mogelijkheid is om kolommen (alternatieven) of rijen toe te voegen (afleidingen of waarden).

- **Decision tree's**
  Complexe logica waarbij beslissingen gecombineerd worden met een bepaalde interactie met een gebruiker, kunnen worden gerealiseerd met decision tree's. Met dit type implementatie element kunnen bijvoorbeeld enquêtes of intelligente vragenlijsten voor het vaststellen van diagnoses worden gerealiseerd. De representatie is een boom waarbij elk blad een vraag, een beslissing, of een te tonen resultaat kan bevatten.

Bovenstaande representaties zijn volgens ons als opzichzelfstaande representaties goed leesbaar voor business stakeholders. Zo zijn er ook verschillende representaties in de change management documenten gevonden die lijken op representaties van de decision tree's en decision tables. In de House casus worden over het algemeen de business rule en decision tree element types niet gebruikt en worden in plaats daarvan vrijwel enkel decision tables en expressions gebruikt voor het realiseren van logica. De reden hiervoor is dat business rules ook in decision tables kunnen worden gerealiseerd en de engineers van Everest deze representatie prefereren boven die van de business rule. De decision tree's blijken maar in zeer specifieke gevallen de voorkeur te hebben, in de House applicatie komen er geen situaties voor waarin dit element op een goede manier toepasbaar is.

Syntax

De syntax van de expressietaal is ook een onderdeel van de representatie mogelijkheden van het platform. De syntax van de expressietaal is vrij simpel en lijkt op de Visual Basic taal die gebruikt wordt in Excel spreadsheets. Ook de functiebibliotheken welke verschillende veelgebruikte functies bevatten lijken hiervan te zijn afgeleid.

De expressietaal is volgens ons echter niet geschikt voor business stakeholders. Allereerst bevat de expressietaal enkel operatoren en functies in de Engelse taal. Aangezien de business stakeholders in het House project in de Nederlandse taal spreken en schrijven kan dit voor sommige stakeholders problemen opleveren. Ook worden er voor operatoren vaak weinigzeggende symbolen gebruikt in plaats van namen. De '[]' in decision tables betekend bijvoorbeeld 'overig'. Hierdoor zijn de expressies vaak moeilijk te begrijpen door iemand zonder programmeer achtergrond. Daarnaast zijn de functies in de functiebibliotheken vrij elementair, waardoor vaak meerdere functies moeten worden gecombineerd om een simpele bewering in de natuurlijke taal te kunnen bewerkstelligen. Tabel 3 illustreert deze problemen:

Natuurlijke taal	Expressietaal van Aquima
Het aantal nieuwe rekeningen in SAS van alle leningdelen bij elkaar mag niet groter of gelijk zijn aan parameterwaarde: Aanvraag.Rel.Marktpartij.pm_MaximumAantalRekeningen	SIZE(COLLECT LeningDeel FROM ALL LeningDeel WHERE (LeningDeel.pm_NieuweRekeningInSAS != ? AND LeningDeel.pm_NieuweRekeningInSAS)) > Aanvraag.Rel.Marktpartij.pm_MaximumAantalRekeningen

Tabel 3: Natuurlijke taal versus de expressietaal Aquima

Hoewel het voorbeeld uit Tabel 3 een vrij extreem voorbeeld is dat via een expressie ook kan worden opgelost, zorgen deze eigenschappen ervoor dat de expressie taal van Aquima qua representatie niet goed aansluit bij de mentale representatie van de business stakeholders.

5.5.2 Analyse

Onder analyse functionaliteit verstaan we de ondersteuning die het platform biedt met betrekking tot het vinden van elementen en het vormen van betekenissen over meerdere elementen heen (herkennen van plans) en het bepalen van de impact van een mogelijke wijziging.

Voor het analyseren van de implementatie is het met name belangrijk om elementen te kunnen verkennen die gerelateerd zijn aan een bepaald element. Met gerelateerd bedoelen we zowel elementen die gebruiken maken van een bepaald element (omhoog) als elementen die gebruikt worden door dat element (omlaag).

Er zijn in Aquima Studio drie mogelijkheden om logica te kunnen benaderen, via elementen uit de modellen, via de lijst weergave en via de dependancy viewer. We zullen alle drie mogelijkheden beoordelen op bruikbaarheid voor het uitvoeren van analyse activiteiten.

- **Lijstweergave**
  Aquima Studio biedt allereerst de mogelijkheid om alle elementen van een bepaald implementatie type in een lijst te zien. Binnen deze lijst worden de naam en het cluster van elementen getoond. Een cluster is een label waarmee handmatig groepen van elementen kunnen worden gemaakt. Deze weergave is enkel bruikbaar als je de exacte naam en het type van het element weet of op zoek bent naar alle elementen binnen een bepaald cluster. De onderhoudsmedewerker is op dit vlak dus afhankelijk van hoe goed de elementen zijn gelabeld. Deze lijst toont daarnaast geen relaties en is dus niet heel bruikbaar voor analyse. Daarnaast is bladeren door de lijst met veel elementen niet handig (zoals bij de implementatie van House). Deze weergave vinden we dus niet geschikt voor het kunnen analyseren van de implementatie.

- **Dependancy viewer**
  De dependancy viewer is bedoeld voor het uitvoeren van analyse activiteiten en kan vanuit elk element worden opgeroepen. Het toont middels een graaf zowel elementen die gebruik

maken van dat element als elementen die gebruikt worden door dat element. Helaas toont het enkel de naam en het type icoontje van de direct gerelateerde elementen (één niveau diep). Deze eigenschappen maken het inschatten van impact van een mogelijke wijziging helaas een intensieve taak. Onze ervaring was dat één niveau in de House implementatie vaak te weinig is en dat de naam en het type element vaak te weinig zeggen over het gebruik van dat element. Het gevolg was dat er meerdere keren moest worden doorgeklikt naar diepere niveaus en detailweergaven van elementen om de impact te kunnen bepalen. Hierdoor ben je al snel het overzicht kwijt en weet je niet meer welke elementen nu precies gerelateerd waren aan elkaar en wat deze relatie betekende. Voor iemand die de implementatie niet goed kent en afhankelijk is van deze viewer is het inschatten van de impact op deze manier erg lastig.

- **Elementen benaderen via andere modellen**
  Vanuit de verschillende proces-, interactie- en documentmodellen kan naar de achterliggende logische elementen worden gegaan. Zo kan er vanuit de pagina editor naar een logische expressie worden genavigeerd en vervolgens kunnen de elementen die zijn aangeroepen in deze expressie worden bezocht. Deze modelrepresentaties zijn echter gericht op het bewerken van de modellen en niet op het efficiënt verkennen van de gerelateerde logische elementen. Daarnaast heeft ook niet iedereen kennis van de applicatie of van het proces waarin een bepaalde logica wordt aangeroepen. Zodra naar een logische element is genavigeerd, is tevens het gebruik van de dependance viewer noodzakelijk om verdere gerelateerde elementen te ontdekken.

De huidige mogelijkheden om logica vanuit verschillende contexten te kunnen benaderen en hun relaties te verkennen is volgens ons voor business stakeholders te omslachtig en ook moeilijk te begrijpen. Er moeten vaak alsnog vele elementen in detail worden bekeken om relaties te verklaren en betekenissen te kunnen vormen. We geloven daarom niet dat de analyse functionaliteit van Aquima Studio de business stakeholders het op enige manier makkelijker maakt om bepaalde onderhoudstaken uit te voeren.

5.5.3 Modificatie

Onder modificatie functionaliteit verstaan we in hoeverre het platform de modificatie activiteit ondersteund. Hiermee bedoelen we zowel het maken van de wijziging als het testen van de wijziging.

Door de ondersteuning van het platform tijdens een wijziging kunnen syntactische, semantische fouten worden voorkomen. Voor business stakeholders is ondersteuning tijdens het uitvoeren van de modificaties belangrijk omdat ze weinig ervaring hebben op het gebied van formele talen. Vaak zijn ze gewend om wijzigingen enkele te beschrijven in natuurlijke taal. Deze beschrijvingen zijn niet heel precies en er is sprake van ambiguïteit. Daarnaast blijken er de praktijk bij Everest ook problemen te zijn in het opstellen van complexe logica. Zo hebben ze moeite met het formuleren van correcte en complete logische statements (zie paragraaf 4.2.1).

Wanneer we business stakeholders willen ondersteunen moeten we ze enerzijds helpen om hun

wijzigingen op de goede syntactische manier te formuleren, maar anderzijds ook helpen bij het opstellen van semantische correcte logica die ook nog eens aansluit op het gedrag wat ze uiteindelijk willen bewerkstelligen. De modificaties op de logische elementen in Aquima Studio behelzen aanpassingen op de logische operatoren en aanpassingen op de condities.

De logische operatoren zijn in logische elementen vaak onderdeel van de representaties, waardoor syntactische fouten op dit vlak veelal worden voorkomen. Een voorbeeld hiervan is het toevoegen van kolom aan een decision table om een alternatieve afleiding te definiëren.

De condities in deze logische elementen zijn expressies die een bewering doen over attributen uit het entiteiten model van de Aquima implementatie. Op dit vlak is er actieve ondersteuning tijdens het formuleren van expressies. Zo zijn er automatische aanvullingssuggesties met betrekking tot de syntaxis of verwijzingen naar waarden uit het entiteiten model en andere implementatie elementen. Daarnaast is er ondersteuning voor syntax-highlighting, zodat correct of incorrecte expressies worden gemarkeerd tijdens het typen.

Bovenstaande ondersteuning is belangrijk voor business stakeholders welke weinig ervaringen hebben de syntax van formele talen. Het stelt ze voor een deel instaat syntactische correcte logica te definiëren.

Een groot gemis qua ondersteuning is echter of de gemaakte modificaties wel overeenkomen met het mentale model van de onderhoudstaak. Met andere woorden, zorgt de modificatie wel voor het gewenste gedrag. Op dit vlak er is nog te veel vrijheid om fouten te maken en te weinig mogelijkheden om modificaties snel te kunnen evalueren.

De expressie editor houdt bijvoorbeeld geen rekening met de context waarin een bepaald implementatie element gebruikt wordt. Wanneer we het hebben over de context van een beoordelingsregel, dan weten we bijvoorbeeld dat het wordt aangeroepen nadat alle informatie over een aanvraag bekend is in het systeem en dat de regel enkel een aanvraag kan afkeuren op basis van deze informatie. Met deze wetenschap kun je de keuze mogelijkheden in de expressie editor beperken zodat er bijvoorbeeld in de condities enkel gebruik gemaakt kan worden van aanvraag informatie die op het moment van aanroepen in het systeem bekend is. Op dit moment kan dat niet, waardoor er informatie uit het entiteiten model kan worden gebruikt die op het moment van aanroepen van de beoordelingsregel nog niet bekend is of voor het beoordelen van een aanvraag helemaal niet relevant is.

Daarnaast is het op dit moment niet mogelijk om bepaalde delen van een logische element te beschermen tegen wijzigingen, door bijvoorbeeld een template voor een bepaalde type business rule te definiëren. Het gevolg is dat een minder kundige gebruiker bijvoorbeeld de afleiding van een beoordelingsregel kan aanpassen zodat het bijvoorbeeld geen status van een aanvraag meer afleid, maar het rente percentage van een aanvraag.

De ondersteuning voor het evalueren van een modificatie is op dit moment ook niet gericht op minder ervaren gebruikers. Er kunnen per implementatie element testscenario's worden opgesteld

en ook kunnen er regressietesten over meer dan een enkel element worden uitgevoerd. Maar bij een aanpassing op een decision table, zoals bijvoorbeeld het toevoegen van een conditie aan een regel, is vaak ook een aanpassingen op een testscenario nodig om deze nieuwe conditie te kunnen testen. Het is zeer de vraag of business stakeholders de vaardigheden hebben om deze test scenario's aan te passen, laat staan dat het gewenst is om ze hiertoe bevoegdheid te geven. Een gebruiksvriendelijke mogelijkheid om middels zelf opgegeven of gegenereerde voorbeelden een deel en een gehele decision table te kunnen testen zou hierbij een uitkomst zijn.

De vrijheid die de Aquima Studio bij modificaties biedt en het gebrek aan laagdrempelige test mogelijkheden zorgen er voor dat implementatie risico's grotendeels afhankelijk zijn van de kundigheid van de onderhoudsmedewerker. Dit maakt de Aquima Studio in onze ogen niet geschikt voor onderhoud door minder ervaren onderhoudsmedewerkers zoals business stakeholders.

5.6 Samenvatting analyse House casus

De onderhoudsmogelijkheden voor business stakeholders in het House onderhoudsproject worden bepaald door de kennis en capaciteiten van de business stakeholders zelf, de onderhoudstaken, de implementatie en de ontwikkelomgeving. De onderhoudstaken zijn hierbij uitgangspunt geweest aangezien zij het kader bepalen van onze analyse, zij bepalen welke onderhoudsmogelijkheden er maximaal zijn voor business stakeholders. De implementatie, ontwikkelomgeving en de kennis en capaciteiten van de business stakeholders zelf bepalen uiteindelijk welke mogelijkheden hiervan over blijven.

In paragraaf 5.1 hebben we vastgesteld dat voor het onderhouden van een implementatie over het algemeen kennis nodig van vier verschillende kennisgebieden: business kennis, functionele kennis, implementatie kennis en kennis van software ontwikkeling. In diezelfde paragraaf hebben we vastgesteld dat de business stakeholders in het House project vaak maar over maximaal twee van deze kennisgebieden beschikken, namelijk kennis van het business domein en kennis van de functionaliteit van de applicatie.

Uit onze analyse van de verschillende wijzigingen binnen het House onderhoudsproject kwamen drie type onderhoudstaken naar voren. Van elk van de drie onderhoudstaken hebben we bekeken welke handelingen er nodig waren op de huidige implementatie om ze te kunnen uitvoeren. Aan de hand van deze handelingen hebben we een inschatting kunnen maken over de kennis en capaciteiten die er nodig zijn om de onderhoudstaken te kunnen uitvoeren.

Hieronder zijn de bevindingen per onderhoudstaak samengevat:

Onderhoudstaak	**Verschillen tussen implementatie en mentale modellen**	**Impact**
Wijzigingen in business logica en business concepten	Klein	Klein bij business rules / Groot bij kleinere business concepten
Wijzigingen in waarneembare delen van de applicatie	Groot	Groot
Wijzigingen in beoordelingsregels	Gemiddeld	Klein

Tabel 4: Eigenschappen onderhoudstaken

Wanneer het verschil tussen de implementatie en de mentale modellen klein zijn dan is het voor een business stakeholder makkelijker om van implementatie elementen een betekenis te vormen en om het doelconcept in de implementatie te herkennen. Bij alle onderhoudstaken zijn de verschillen tussen de implementatie en mentale modellen van de business stakeholders volgens ons te groot.

Deze verschillen zijn volgens ons niet op een makkelijke manier op te lossen. Zo blijkt uit de analyse dat het lastig is om bijvoorbeeld wijzigingen in het waarneembare gedrag onderhoudbaar te maken zonder hierbij de implementatietaal voor dit soort onderhoudstaken aan te passen. Ook het onderhoudbaar maken van de business concepten, business rules en beoordelingsregels is kostbaar. Het aanpassen van de compositie en naamgeving van deze logische elementen voor een betere aansluiting op de mentale modellen van de business stakeholders is gezien de omvang van het hypotheek domein een grote investering.

Een ander belangrijk aspect bij het uitvoeren van de onderhoudstaken is het kunnen begrijpen van de impact van een modificatie. Onze analyse toont aan dat de complexiteit van de House applicatie zorgt dat deze impact bij veel wijzigingen moeilijk te overzien en te begrijpen is. Dit geldt met name voor de onderhoudstaken die betrekking hebben business concepten en het waarneembaar gedrag van de applicatie. Ook hier liggen oplossingen niet voor de hand, voor het reduceren van de impact is vaak enkel het verminderen van de complexiteit van de applicatie een oplossing. In het geval van de House implementatie blijkt deze complexiteit niet een functie te zijn van slecht ontwerp, maar te danken te zijn aan de hoge eisen die tegenwoordig aan dit soort business applicaties worden gesteld. Het reduceren van impact lijkt dus niet realistisch in het geval van de House casus.

De bovengenoemde problemen zijn niet enkel te wijten aan hoe de House applicatie is geïmplementeerd, maar zijn ook inherent aan de manier waarop applicaties kunnen worden opgebouwd in de Aquima Studio. Doordat de type elementen waaruit Aquima implementaties bestaan van vrij elementair niveau zijn, kunnen vrijwel alle type applicaties worden gerealiseerd. Deze flexibiliteit komt ook met een nadeel, een applicatie is al snel opgebouwd uit vele duizenden elementen die met elkaar in relatie staan. Hoewel dit voor onderhoud aan de goed gescheiden business logica niet zo'n probleem vormt, maakt dit onderhoud aan het waarneembare gedrag van de applicatie al snel complex.

De Aquima Studio biedt voor business stakeholder verder weinig analyse mogelijkheden om met deze complexiteit om te kunnen gaan. Het in kaart brengen van de gevolgen van een modificatie is een intensieve taak in de Aquima Studio. Ook op het gebied van ondersteuning bij het uitvoeren van een modificatie wordt op dit moment uitgegaan van de expertises van een ontwikkelaar in plaats van die van een business stakeholder.

Al met al lijken er voor alle geïdentificeerde onderhoudstaken verschillende problemen te zijn die de onderhoudsmogelijkheden voor business stakeholders op dit moment in de weg staan. In hoofdstuk 6 zullen we wat dieper ingaan op de problemen die we hebben gedefinieerd en zullen we mogelijk oplossingen voor deze problemen bespreken.

6. Problemen en oplossingen bij onderhoud door business stakeholders

6.1 Huidige onderhoudssituatie versus wensen business stakeholders

Bij complexe implementaties zoals de onderzochte House implementatie lijken er twee grote krachten te spelen welke ontwikkelen van oplossingen voor business stakeholders tegenhouden.

Zo zijn er het huidige projectmanagement en de IT stakeholders binnen het onderhoudsproject. Zij zijn gewend aan de huidige manier van onderhoud, de verdeling van de taken, de kosten en de manier waarop met risico's wordt omgegaan. Ze zijn zich verder bewust van de complexiteit van de implementatie en het feit dat bij het implementeren van veel wijzigingen een goede kennis van de implementatie en ontwikkelervaring noodzakelijk is. Zij zien de huidige kosten en risico's als een soort baseline en zullen hun best doen om dit niveau minimaal te handhaven. Elke ontwikkeling welke de risico's of de kosten van onderhoud mogelijk kan verhogen wordt gezien als een bedreiging.

Ander de andere kant zijn er business stakeholders welke graag onderhoudstaken zelf willen kunnen doen omdat ze op deze manier hun werk sneller en beter te doen. Zij bekijken de onderhoudsactiviteiten vaak met een iets grotere afstand en zijn zich wat minder bewust van de risico's en technische implicaties van onderhoud. Zij zien de tijd van een wijzigingsverzoek tot een oplevering vaak als te lang en vragen zich hierdoor hardop af waarom ze met wat technische oplossingen dit onderhoud niet zelf kunnen doen. Deze business stakeholders hebben vaak geen kennis van de implementatie van de applicatie en hebben geen ervaring op het gebied van software ontwikkeling, maar zien het vergaren van deze kennis aan de andere kant ook niet als hun kerntaak.

Deze twee krachten lijken ervoor te zorgen dat het ontwikkelen van oplossingen die mogelijk onderhoud in de handen van de business stakeholders kan leggen maar moeizaam verloopt.
De vrees voor hogere risico's en onderhoudskosten zorgt voor conservatieve houding wat betreft het aanbieden en ontwikkelen van onderhoudsmogelijkheden. Zo worden er enkel delen van de implementatie ter configuratie aangeboden waarvan de risico's met grote zekerheid kunnen worden ingeschat en beheerst. Het ontwikkelingen van meer geavanceerdere mogelijkheden die uitgaan van de huidige kennis en capaciteiten van de business stakeholders worden al gauw als onhaalbaar aangedaan. Dit komt mede ook door de weinige kennis die beschikbaar is over de problemen die hierin een rol spelen.

Gedurende dit onderzoek en de analyse van de House casus hebben we hier echter een beter beeld gekregen van de problemen en mogelijkheden. Deze zullen we nu gaan bespreken.

6.2 Deelproblemen en oplossingen

Onze analyse van de House casus heeft aangetoond dat bij onderhoud op business applicaties al snel veel kennis van de implementatie vereist is om zonder veel risico's op fouten onderhoudstaken te kunnen uitvoeren.

Aan de andere kant lijken de change management documenten en de informatie uit de interviews aan te tonen dat we op dit vlak weinig hoeven te verwachten van de business stakeholders. De lange inwerktijd van business engineers in het klantteam van Everest geeft aan dat een opleiding in de House implementatie en de Aquima Studio waarschijnlijk voor vele business stakeholders te hoog gegrepen is. Daarnaast lijkt er ook vanuit business stakeholders een weerstand te bestaan tegen het opdoen van kennis buiten hun eigen vakgebied.

Om het gat tussen de benodigde kennis en capaciteiten en de beschikbare kennis en capaciteiten te kunnen overbruggen, is de inzet van ondersteunende technologie in veel gevallen dus noodzakelijk.

Een manier om dit gat te overbruggen is om bijvoorbeeld een abstractie laag aan te brengen op de implementatie net zoals de huidige BPM suites abstracties zijn op 3GL ontwikkeltalen. Hoewel er waarschijnlijk wel mogelijkheden liggen om de user interface en modellen van het BPM suites zoals Aquima 'business-vriendelijker 'te maken, gaan grote wijzigingen op dit gebied al snel ten kosten aan de expressiviteit van het platform. Bij het aanbrengen van abstracties op het gebied van implementatietalen gaat namelijk altijd informatie en dus ook expressiviteit verloren. Met 3GL implementatie talen kunnen in die zin een breder scala aan applicaties worden ontwikkeld dan met de huidige BPM suites.

Dat deze expressiviteit belangrijk is, bewijst de House implementatie. De implementatie van de House applicatie is namelijk niet zo complex vanwege slecht ontwerpkeuzes, maar vanwege de vele eisen die er worden gesteld met betrekking tot de dynamiek en mogelijkheden van de applicatie. Ons lijkt daarom een generieke abstractie laag bovenop de huidige generatie BPM suites enkel voor het kunnen realiseren van simpele applicaties nuttig, waarbij standaardoplossingen voldoende zijn.

Het ontwikkelen van onderhoudstaak specifieke oplossingen waarbij de huidige implementatietaal van BPM suites behouden en toegankelijk blijft, lijkt in dit geval meer perspectief te bieden. Door specifieke binnen de grenzen van onderhoudstaak een oplossingen te ontwikkelen kan er ook qua abstractie meer afstand worden genomen van de implementatie.

Als we de onderhoudstaken in de House casus als voorbeeld nemen, dan zou een abstractie die specifiek gericht is op een bepaalde set regels, zoals beoordelingsregels, er uit kunnen zien als een template van een beoordelingsregel in de natuurlijke taal, waarbij de invoermogelijkheden op basis van context zijn beperkt. Door deze invoermogelijkheden te beperken zou het mogelijk zijn om de implementatie details automatisch af te leiden.

Om onderhoudstaak gerichte oplossingen te ontwikkelen die ook voor business stakeholders kunnen worden gebruikt zijn echter wel een aantal problemen moeten die moeten worden opgelost. Deze problemen zullen we in paragraaf 6.2.1 t/m 6.2.3 bespreken. Vervolgens zullen we in paragraaf 6.2.4 ingaan op hoe het ontwikkelen van dergelijke onderhoudsoplossingen volgens ons het best kan worden aangepakt.

6.2.1 Representaties

Voor het aanbieden van representaties die effectief zijn is het belangrijk om de representaties aan te sluiten bij de mentale representatie van de onderhoudstaak van de gebruiker, in dit geval de business stakeholder die de onderhoudstaak uitvoert.

Dit kan op de meeste simpele manier door bijvoorbeeld de taal van de interface aan te sluiten op het taalgebruik waarin de onderhoudstaak wordt bedacht of opgeschreven. Zo zal het bij een onderhoudstaak die betrekking heeft op het wijzigen van een business rule effectief zijn om de representatie van de business rule te laten lijken op het mentale model wat de business stakeholder heeft van een dergelijke business rule.

Met name in complexe en omvangrijke implementaties waarin veel logica wordt gebruikt zoals House is gebleken dat het belangrijk is om naast zinvolle representaties van de te modificeren elementen ook representaties aan te bieden welke het makkelijk maken om de te modificeren elementen te benaderen en te herkennen.

Zo kan het zinvol zijn om een business stakeholder, zoals een gebruiker van de applicatie, de logica te laten benaderen via de front-end, omdat hij gewend is te werken met de applicatie en logica aan bepaalde momenten in de interactie relateert. Het simpel weg zoeken naar een naam van het element in de implementatie zal voor een dergelijke business stakeholder veel moeilijker zijn.

De logica die kan worden benaderd moet in dit soort gevallen echter wel aansluiten bij de voor de business stakeholder bekende representaties. In de House casus is gebleken dat dit bij bijvoorbeeld minder concrete specificaties zoals wijzigingen in de het waarneembare gedrag van de applicaties erg lastig is. Een functionaliteit van de applicatie is vaak een samenspel tussen verschillende elementen in de implementatie. De business stakeholders beschikken over representaties van een dergelijke functionaliteit, maar kunnen zonder uitgebreide kennis van de implementatie deze functionaliteit niet relateren aan de verschillende elementen die dit mogelijk maken. Bij business logica is het verschil tussen de mentale representaties van de onderhoudstaak vaak kleiner. De business logica vertegenwoordigen een domein wat bekend is bij de business stakeholders. Wanneer de implementatie qua compositie en naamgeving aansluit bij hoe de corresponderende business concepten en business rules in de praktijk worden gecommuniceerd, is het voor business stakeholders makkelijk om deze implementatie elementen te relateren aan hun interne representaties van het business domein.

Helaas blijkt uit onze analyse van de House implementatie dat de huidige business logica implementatie vaak niet overeenkomt met deze business concepten en business rules. Ook leert de analyse van de House implementatie ons dat het ontwerpen van de business logica op deze manier, zeer veel discipline en kosten met zich mee kan brengen. Dit geld al helemaal voor complexe business domeinen zoals het hypotheek domein.

Het is in de praktijk moeilijk om deze investeringen te verantwoorden als deze business logica enkel voor één applicatie is opgezet. Wanneer de implementatie echter voor meerdere applicatie kan

worden gebruikt en ook voor documenterende doeleinden kan worden ingezet, dan kan het goed opzetten en onderhouden van een dergelijk geformaliseerd business domein veel voordelen voor een organisatie bieden.

De SBVR standaard voor formele specificatie van business concepten en business rules (SBVR) is speciaal voor dit doel ontwikkeld. Door gebruikt te maken van deze specificatie standaard zou de resulterende implementatie makkelijker kunnen worden uitgewisseld met andere applicaties en kunnen zelf beleidshandboeken en business glossaries kunnen worden gegenereerd. Daarnaast zijn de voordelen van een dergelijk standaard dat je niet meer vast zit aan een leverancier specifieke implementatie, maar gebruik kan maken van vele ondersteunende tools die voor de SBVR standaard zijn ontworpen. Het onderbrengen van dit geformaliseerde business domein in externe tools brengt echter ook problemen met zich mee. Het is bijvoorbeeld een stuk moeilijker om bij een wijziging in een business rule inzicht te krijgen in de impact van deze wijziging op de implementatie van gebruikmakende applicaties. Een geïntegreerde oplossing binnen een BPM suites zoals Aquima Studio zal dus wenselijk zijn.

6.2.2 Analyse

Een ander aspect wat heel belangrijk is gebleken tijdens ons onderzoek is het ondersteunen van business stakeholders met betrekking tot de analyse activiteit. De verschillende onderhoudstaken in de House casus blijken namelijk vaak veel impact te hebben op andere delen van de implementatie. Het kunnen inschatten van de gevolgen van een modificatie is een belangrijke taak in het evalueren van een mogelijke oplossing en het testen van reeds uitgevoerde modificaties.

De huidige trend is dat de front-end van applicaties steeds rijker en dynamischer worden door de vraag naar hogere proces efficiëntie en toename van self servicing mogelijkheden voor klanten. Waar vroeger voor het regelen van bankzaken een bezoek aan het bankkantoor nodig was, kunnen klanten tegenwoordig hun bankzaken steeds meer zelf regelen zonder tussenkomst van personeel van de bank. Ditzelfde geldt voor middle office systemen zoals House, waarbij intermediairs zelf direct hypotheken kunnen opvoeren in de systemen van Westland Utrecht Bank. Personeel wordt in beide gevallen vervangen door schermen die slimme beslissingen kunnen maken en een soort micro proces vormen. BPM suites spelen op deze vraag in door tools te bieden waarin geavanceerde interactie op invoerveld niveau kan worden gedefinieerd.

Het gevolg hiervan is dat logica steeds vaker wordt aangeroepen vanuit de front-end. Waar vroeger de schermen enkel de invoer regelden en logica enkel werd aangeroepen wanneer er bijvoorbeeld op een knop werd gedrukt, wordt logica tegenwoordig tijdens het invullen van een formulier al aangeroepen en passen de schermen zich constant aan de ingevulde data.

Dit maakt het onderhoud van dit soort applicaties complexer. Wanneer een wijziging in de onderliggende logica plaatsvindt zal deze impact dan ook groter zijn bij dit soort systemen. Het kan bijvoorbeeld zijn dat bepaalde interactie elementen die afhankelijk zijn van deze business logica plotseling niet meer worden getoond. Maar ook andersom is de impact van een wijziging in de interactie moeilijker in te schatten en kunnen kleine wijzigingen in de schermen er bijvoorbeeld voor

zorgen dat bepaalde business logica niet meer wordt aangeroepen of dat deze logica ongewenste resultaten gaan opleveren.

Huidige tools die het apart beheren van bijvoorbeeld business logica mogelijk maken, lijken geen rekening te houden met deze trend en gaan nog uit van de oude gedachte dat business rules en business concepten enkel worden aangeroepen op vaste beslismomenten op proces niveau. Omdat de impact van een wijziging in deze situatie niet heel groot is, bieden deze tools vaak maar beperkte analyse mogelijkheden aan. Ook in het Aquima platform zijn de analyse mogelijkheden op dit gebied beperkt.

De geboden analyse mogelijkheden zijn vaak syntactisch van aard. Zo laten de relationele schema's van de elementen enkel zien dat er een relatie bestaat, zonder informatie te weer te geven over wat deze relatie inhoud (zoals de dependancy viewer in Aquima Studio). Daarnaast wordt er gebruik gemaakt van syntactische impact analyses waarbij niet de werkelijke impact van een modificatie te zien is, maar enkel de potentiële impact, die veel groter is en in veel gevallen niet eens relevant is. Deze analyse mogelijkheden maken het inschatten van impact een intensieve taak voor business stakeholders. Zij hebben geen kennis van de implementatie en zijn dus afhankelijk van deze analyse mogelijkheden voor het vormen van een betekenis van de applicatie.

Voor business stakeholders lijken meer geavanceerde analyse mogelijkheden dus noodzakelijk. Hoewel met hun kennis van het business domein, het begrijpen van business logica elementen minder moeite kost, is het vormen van betekenissen van bijvoorbeeld applicatie logica of andere element typen erg lastig aangezien deze niet gerelateerd kunnen worden aan representaties uit hun mentale modellen. Ook de hoeveelheid elementen die mogelijk moeten worden begrepen kunnen de capaciteiten van de business stakeholders op de proef stellen.

Huidige BPM suites lenen zich echter heel goed voor meer geavanceerdere analyse mogelijkheden. De element types hebben vaak een specifiek doel en hebben een standaard set van eigenschappen. Deze semantische informatie kan worden gebruikt om de analyse te ondersteunen.

Een geavanceerd analyse hulpmiddel kan bijvoorbeeld een relationeel schema zijn waarbij niet alleen de relatie tussen elementen wordt weergegeven, maar waarbij ook extra informatie wordt getoond op basis van de semantiek. Zo kan bijvoorbeeld worden weergegeven dat het resultaat van een decision table wordt gebruikt voor het bepalen van een standaardwaarde van een invulveld welke zich bevindt op een bepaalde pagina. Deze informatie is ook nuttig bij geautomatiseerde impact analyse technieken. Zo kan je bij een modificatie op worden geattendeerd dat de logica die je probeert te wijzigen wordt gebruikt bij het berekenen van het rente percentage van de hypotheekaanvraag. Daarnaast kunnen deze automatische impact analyses slimmer worden gemaakt door gebruikt te maken van bestaande testscenario's van de elementen. Met deze data zou bijvoorbeeld een betere inschatting kunnen worden gemaakt op de daadwerkelijke impact van een modificatie.

Ook zou het mogelijk zijn om elementen te annoteren met extra informatie. Zo zou een rente percentage attribuut kunnen worden geannoteerd als zijnde gevoelige data, waarnaar wijzigingen die

impact hebben op dit element een extra validatie nodig is van een business engineer.

Bovenstaande verbeteringen kunnen business stakeholders helpen bij het begrijpen van elementen, het inschatten van impact, maar ook bij het fijnmaziger kunnen beheersen van mogelijke implementatie risico's.

6.2.3 Modificatie

Voor het uitvoeren van een modificatie op de bestaande implementatie moet een business stakeholder zich kunnen uitdrukken in de implementatietaal. Dit vormt een probleem aangezien business stakeholders niet gewend zijn om in formele talen te specificeren maar in natuurlijke taal.

Specificaties die door business stakeholders zijn opgesteld blijken vaak weinig precies, er worden soms onbekende business concepten gebruikt, bepaalde feiten worden niet expliciet genoemd en ook de logica kan soms op meerdere manieren worden geïnterpreteerd. In de formele implementatie talen moeten expressies exact worden geformuleerd en moeten ze syntactisch en semantisch correct zijn.

Ondersteuning van deze business stakeholders om deze formele expressies te formuleren lijkt daarom noodzakelijk. Het stellen van restricties aan de mogelijke invoer kan helpen de business stakeholder te sturen in het formuleren van correcte expressies. Deze restricties kunnen echter ook frustrerend zijn voor de business stakeholders en op deze manier creativiteit belemmeren. Voor het bieden van ondersteuning op dit vlak moet er dus een balans worden gevonden tussen te accepteren risico’s en deze restricties.

Moderne ontwikkelomgevingen hebben tegenwoordig diverse functies in huis om de ontwikkelaar te ondersteunen in het formuleren van correcte statements, zoals syntax-highlighting en auto completion. Hoewel deze technieken ondersteunen in het opschrijven van correcte statements, zijn ze niet dwingend, de ontwikkelaar heeft de vrijheid om fouten te maken of af te wijken van de suggesties.

Bovenstaande mogelijkheden zijn eigenlijk alleen van toepassing als de business stakeholder ook een goed beeld heeft van de daadwerkelijke oplossing. Bij onderhoudstaken gericht op enkel de business logica is dit vaak niet zo’n groot probleem aangezien er vaak maar één oplossing mogelijk is en de modificaties vaak corresponderen met maar één implementatie element. Bij onderhoudstaken gericht op functionele wijzigingen aan een applicatie zijn de wijzigingsverzoeken vaak minder concreet omdat er vele oplossingen mogelijk zijn. Ook zijn de onderliggende logica elementen welke een bepaalde functionaliteit mogelijk maken onbekend voor de business stakeholders. Ondersteuning bieden voor het vinden van een correcte oplossing is in deze gevallen zeer lastig.

Tot een zeker hoogte zou het naspelen van de communicatie, die normaal gesproken tussen business en IT stakeholders plaatsvindt, kunnen helpen bij het oplossen van dit soort problemen. Door gericht suggesties te doen of vragen te stellen tijdens het uitvoeren van een modificatie of bij het selecteren van implementatie elementen kunnen business stakeholders worden gestuurd in het implementeren

van gewenste oplossingen. Hoe verder echter het beeld van de business stakeholder verwijderd is van de juiste oplossingen hoe complexer deze dialogen moeten worden. Bij grote verschillen is het ontwikkelen van dit soort dialogen, gezien het probleem dat moet worden opgelost, waarschijnlijk niet realistisch. Meer onderzoek naar de mentale modellen van de business stakeholders kunnen echter helpen bij het identificeren van veel voorkomende vergissingen. Anderzijds kunnen ook zelflerende systemen helpen bij het opbouwen van complexe dialoog structuren.

6.2.4 Ontwikkelen van onderhoudsoplossingen

Als we kijken naar de oplossingen die genoemd zijn in de voorgaande paragrafen en de problemen die we tijdens onze analyse van de House casus hebben geïdentificeerde, dan kunnen we concluderen dat voor het geschikt maken van de er serieuze investeringen in zowel het aanpassen van de implementatie als het ontwikkelen van ondersteunende technieken nodig zijn om de onderhoudstaken voor business stakeholders geschikt te maken.

Een gerichte aanpak voor een beperkte set van logica lijkt ons op daarom het meest effectieve aanpak. Voor de selectie van logica zou er gekeken moeten worden naar hoe vaak gedurende de onderhoudsfase plaatsvinden, de mogelijkheid om de impact van de wijziging te kunnen overzien en de kosten die nodig zijn om de logica begrijpelijk en modificeerbaar te maken voor de business stakeholders. Het is hierbij belangrijk om de implementatie risico's zoveel mogelijk te beheersen. Met name de impact van wijzigingen zien we als een problematisch omdat dit de scope, van wat er van de implementatie begrepen moet worden, erg kan vergroten. Bij een hoge impact is het dus zowel qua kosten als mogelijkheden lastiger om oplossingen te ontwikkelen die dit soort wijzigingen beschikbaar zouden maken voor business stakeholders.

Voor de onderhoudstaken in de House casus zou een aanpak gericht op het onderhoudbaar maken een specifieke set van veel gewijzigde business rules of een selectie van de beoordelingsregels het meest zinvol zijn. Bij dit type logica is de impact zeer klein en zijn de verschillen met de mentale modellen van de business stakeholders niet heel groot.

7. Conclusie

Aan de hand van de analyse en de bestudeerde literatuur uit de voorgaande hoofdstukken zullen we in dit hoofdstuk proberen een antwoord te geven op de door ons gestelde hoofdvraag en zullen we mogelijkheden voor toekomstig onderzoek aanwijzen.

7.1 Antwoord op de onderzoeksvraag

De hoofdvraag de we proberen te beantwoorden in deze thesis is als volgt:

"Hoe kunnen we het onderhoud van logica door business stakeholders mogelijk maken?"

We hebben in deze thesis middels de analyse van de House casus inzicht gegeven in verschillende problemen die op dit moment het onderhouden van logica door business stakeholders in de weg staan. Enerzijds hebben we in hoofdstuk 4 geïdentificeerd welke problemen op dit moment de ontwikkelingen op dit gebied tegenhouden, anderzijds hebben we door de House casus goed te analyse inzicht gegeven in de mogelijke problemen die zouden kunnen optreden als business stakeholders wel deze mogelijkheden zouden hebben. In hoofdstuk 6 hebben we de geïdentificeerde problemen samengevat en voorzichtig oplossingsrichtingen aangewezen voor deze problemen.

We kunnen uit onze bevindingen concluderen dat het onderhoud van logica door business stakeholders een veelzijdig probleem is. Er is niet een enkel knelpunt dat moet worden opgelost, maar er zijn vele factoren die een rol spelen. Zo wijst onze analyse erop dat er meer aandacht nodig is voor het ondersteunen van de cognitieve stappen van het onderhoudsproces binnen ontwikkelomgevingen. Tijdens onze analyse hebben we op dit vlak verschillende problemen in de representatie, analyse en modificatie functionaliteiten van Aquima Studio kunnen identificeren die het onderhouden van logica door business stakeholders kunnen bemoeilijken. Ook de implementatie van logica blijkt een grote rol te spelen in de onderhoudsmogelijkheden voor business stakeholder. Niet alleen het aanhouden van bekende software design principes is hierin belangrijk, maar ook de daadwerkelijke compositie en naamgeving van implementatie elementen kunnen een groot verschil maken in de onderhoudsmogelijkheden voor business stakeholders. Onze ervaring met de House implementatie heeft geleerd dat het ontwerpen van logica voor onderhoudbaarheid lastig is bij implementaties van een dergelijke omvang en in bedrijfsomgevingen waar er onder een constante tijdsdruk moet worden gewerkt.

Voor het uiteindelijk ontwikkelen van een onderhoudsoplossing voor business stakeholders die daadwerkelijk voordelen biedt boven de huidige manier van werken, zullen veel van bovenstaande problemen te gelijk moeten worden opgelost. Sommige oplossingen voor deze problemen zijn echter dermate complex dat we beseffen dat niet voor elke onderhoudstaak en voor elke business stakeholder een dergelijke onderhoudsoplossing zal bestaan of op haalbare manier ontwikkeld kan worden. Door echter gericht te ontwikkelen voor een bepaald type onderhoudstaak en voor een specifieke groep business stakeholders, zijn er volgens ons zeker successen te behalen. Zo zien we op basis van onze analyse van de House casus zeker een mogelijkheid om business analisten en product managers delen van de beoordelingsregels te laten onderhouden.

De resultaten uit deze thesis kunnen in dit licht dienen als handige referentie bij het ontwikkelen van dergelijke onderhoudsoplossingen voor business stakeholders.

7.2. Toekomstig werk

De problemen die we hebben geïdentificeerd zijn gebaseerd op enerzijds een literatuur studie naar de verschillende activiteiten binnen het onderhoudsproces en anderzijds de analyse van de House casus. Middels de literatuur studie hebben we een ons een voorstelling gemaakt over hoe en welke onderhoudsactiviteiten er plaatsvinden in het onderhouden van applicaties die gerealiseerd zijn middels BPM suites zoals Aquima. De definities die we hiervoor hebben geïntroduceerd hebben we vervolgens toegepast tijdens onze analyse van de House casus.

Hoewel we slechts één casus op deze manier hebben kunnen analyseren lijkt het model en de onderliggende literatuur een goed handvat te zijn om onderhoudsprojecten te analyseren op mogelijke knelpunten. Met name de gebruikte theorieën over cognitieve fit waarbij uitgaan wordt van mentale modellen van zowel de applicatie als de onderhoudstaak blijken zeer verhelderend en passend. Gezien de exploratieve aard van dit onderzoek hebben we de inhoud van deze mentale modellen gebaseerd op informatie uit de interviews, de documentatie en voor een deel ook op basis van intuïtie.

Onderzoek gericht op het in beter in kaart krijgen van deze mentale modellen van business stakeholders kan volgens ons waardevol inzicht geven in de problemen die er spelen en de sleutelrol spelen in het ontwikkelen van geschikte onderhoudsoplossingen voor deze groep stakeholders.

Daarnaast kan door onderzoek naar andere praktijk situaties bijdragen aan het begrijpen van de problemen. Zo zou onderzoek naar onderhoudsprojecten waarbij business stakeholders op dit moment al succesvol onderhoud uitvoeren een beter beeld geven van welke problemen mogelijk meer belangrijk zijn dan andere. Ook zou onderzoek naar kleinschaligere onderhoudsprojecten bijvoorbeeld inzicht kunnen geven in de invloed van omvang op de door ons geïdentificeerde problemen.

Referenties

(Dualthinkers, 2010)
Groot tekort aan Ict'ers verwacht. (2010, December 6), Geraadpleegd op 10 Juli, 2012, van Vkbanen.nl: http://www.vkbanen.nl/banen/artikel/groot-tekort-aan-icters-verwacht/213228.html

(Henderson, 1992)
J. Henderson & N. Venkatraman, "Strategic Alignment: A model for organisational transformation through information technology," in T. Kochan & M. Unseem, eds, Transforming Organisations, Oxford University Press, NY, 1992.

(IEEE, 1998)
IEEE. IEEE Standard for Software Maintenance (IEEE Std 1219–1998). Institute for Electrical and Electronic Engineers: New York NY, 1998;

(Letovsky, 1986a)
Stanley Letovsky. 1986. Cognitive processes in program comprehension. In *Papers presented at the first workshop on empirical studies of programmers on Empirical studies of programmers*, Elliot Soloway and Sitharama Iyengar (Eds.). Ablex Publishing Corp., Norwood, NJ, USA, 58-79.

(Letovsky, 1986b)
S. Letovsky and E. Soloway. 1986. Delocalized Plans and Program Comprehension. *IEEE Softw.* 3, 3 (May 1986), 41-49.

(Lientz, 1978)
B. P. Lientz, E. B. Swanson, and G. E. Tompkins. 1978. Characteristics of application software maintenance. Commun. ACM 21, 6 (June 1978), 466-471

(Mayrhauser, 1995)
Anneliese von Mayrhauser and A. Marie Vans. 1995. Program Comprehension During Software Maintenance and Evolution. *Computer* 28, 8 (August 1995), 44-55.

(Pennington, 1987)
Pennington, N. 1987. Stimulus structures and mental representations in expert comprehen-sion of computer programs. Cognitive Psychology, 19, 295}341.

(Robson, 1991)
D. J. Robson, K. H. Bennett, B. J. Cornelius, and M. Munro. 1991. Approaches to program comprehension. *J. Syst. Softw.* 14, 2 (February 1991), 79-84.

(Shaft&Vessey, 2006)
Shaft, T.M. and I. Vessey, The Role of Cognitive Fit in the Relationship between Software Comprehension and Modification. MIS Quarterly, 2006. 30(1)

(Standish, 1984)

Thomas A. Standish. 1984. An Essay on Software Reuse. *IEEE Trans. Softw. Eng.* 10, 5 (September 1984), 494-497.

(Swanson, 1976)
E. Burt Swanson, The dimensions of maintenance. Proceedings of the 2nd international conference on Software engineering, San Francisco, 1976, pp 492-497

(Vessey, 1986)
I Vessey. 1986. Expertise in debugging computer programs: an analysis of the content of verbal protocols. IEEE Trans. Syst. Man Cybern. 16, 5 (September 1986), 621-637

(Vessey,1991)
Iris Vessey and Dennis Galletta, Cognitive Fit: An Empirical Study of Information Acquisition. Information Systems Research, 1991 2:63-84