EQuA – Symbiosis: multi user aspects

Maurice Hagemeijer
Afstudeerscriptie
Eindhoven 04-06-2013

Gegevens stage bedrijf:

Naam:	ISAAC
Adres:	Marconilaan 16 5621 AA Eindhoven
Telefoonnummer:	+31 (0)40 290 89 79
E-mailadres:	info@isaac.nl

Bedrijfsbegeleider:

Naam:	Ronald Middelkoop
E-mailadres:	ronald.middelkoop@isaac.nl
Telefoonnummer:	040 215 53 66

Gegevens docentbegeleider:

Naam:	Coen Crombach
E-mailadres:	c.crombach@fontys.nl
Telefoonnummer:	+ 31 8850 89241

Gegevens stagiair:

Voorletters:	M.L.
Naam:	Hagemeijer
Adres:	Strijlant 42 5641PT Eindhoven
Telefoonnummer:	+ 31 (0) 6 212 22 939
E-mailadres:	m.hagemeijer@student.fontys.nl
Studentnummer:	2156000
Afstudeerrichting:	Software Engineering
Opleiding:	Fontys Eindhoven
Begin stage:	11 februari 2013
Einde stage:	28 juni 2012
Werkdagen:	85 dagen

Getekend voor gezien door bedrijf begeleider:

Datum getekend: Naam en handtekening bedrijf begeleider:

Documenthistorie

Versie	Status	Datum	Wijzigingen
0.1.0	Concept	2013-03-01	Eerste opzet
0.1.1	Concept	2013-03-05	Uitbreiding document
0.1.2	Concept	2013-03-07	Opmaak aangepast + uitbreiding
0.1.3	Concept	2013-03-19	Uitbreiding document
0.1.4	Concept	2013-04-15	Onderzoeksvragen + uitwerking toegevoegd
0.1.5	Concept	2013-04-18	Bronvermelding volgens APA richtlijnen gemaakt
0.1.6	Concept	2013-05-22	Uitbreiding document
0.2.0	Concept	2013-06-04	Afronding concept versie document
1.0.0	Versie 1	2013-06-11	Afgeronde versie.

Goedkeuring

Versie	Datum	Naam	Functie	Paraaf
		Ronald Middelkoop	Bedrijfsbegeleider	
		Coen Crombach	Docentbegeleider	

Distributielijst

Versie	Datum verzending	Naam	Functie
		Ronald Middelkoop	Bedrijfsbegeleider
0.1.4	15-04-2013	Coen Crombach	Docentbegeleider
0.1.4	15-04-2013	Rene van der Heijden	Stage Coördinator S
1.0.0	11-06-2013	Stage bureau	stage bureau

Voorwoord

Deze scriptie is tot stand gekomen tijdens mijn afstudeerstage voor de opleiding HBO-ICT afstudeerrichting Software Engineering. De afstudeerstage is uitgevoerd bij ISAAC Software Solutions BV. Dit project heeft een looptijd gehad van 20 weken. Binnen de scriptie wordt het proces, de producten en resultaten van mijn afstudeerstage beschreven.

Graag zou ik deze mogelijkheid nemen om een aantal mensen te bedanken die me hebben geholpen of begeleid tijdens mijn stage. Om te beginnen wil ik graag mijn opdrachtgever Frank Peeters bedanken voor het aanbieden van de opdracht en de prettige samenwerking. Daarnaast wil ik mijn docent begeleider Coen Crombach bedanken voor de nuttige tips, de voortgang bewaken en feedback te geven op de documenten. Verder wil ik natuurlijk iedereen bij ISAAC bedanken voor de plezierige tijd en in het speciaal Ronald Middelkoop. Ronald is tijdens mijn stage zeer betrokken geweest in de technische begeleiding, feedback geven op de documenten en het bewaken van de voortgang.

Maurice Hagemeijer
Eindhoven, juni 2013

Samenvatting

Gedurende mijn afstudeerstage heb ik gewerkt bij ISAAC Software Solutions in opdracht van het EQuA project. Het EQuA project richt zich op het verbeteren van de kwaliteit van software in een zo vroeg mogelijk stadium van het ontwikkelproces. Binnen dit project is er een tool in ontwikkeling die de synchronisatie tussen requirements en het objectmodel regelt genaamd Symbiosis. Voor deze tool heb ik onderzoek gedaan hoe deze multi-user gemaakt kan worden en deze oplossing geïmplementeerd. Tevens is er gekeken naar een alternatieve GUI toolkit en het integreren van de applicatie als Eclipse plug-in.

Als eerste heb ik gekeken naar de alternatieven voor de GUI toolkit aangezien er een flinke contributie aan de GUI nodig was binnen de tool. De tool maakt gebruik van het Swing framework welke niet verder ontwikkeld wordt. De bekeken alternatieven SWT en Java FX 2.0 zijn vergeleken met het Swing framework en er zijn prototypes gemaakt van enkele componenten van de tool om te vergelijken hoeveel werk het opnieuw bouwen van de GUI in een andere techniek zou kosten. Hieruit bleek dat het ombouwen naar een andere GUI techniek niet de moeite waard zou zijn in verband met de geringe voordelen en enorme hoeveelheid tijd dat het zou kosten.

Vervolgens heb ik gekeken naar de mogelijkheden voor het integreren van de Symbiosis applicatie als Eclipse plug-in. Aangezien de Symbiosis applicatie een andere GUI toolkit gebruikt dan dat Eclipse vereist voor zijn plug-ins als ze een toevoeging aan de GUI maken moest er onderzocht worden of het technisch mogelijk was deze te integreren of dat hier een aparte GUI voor ontwikkeld moest worden. Het bleek technisch mogelijk maar hier kwamen een hoop problemen bij kijken. Ten eerste leunde de applicatie erg op het gebruikte docking framework en waren er veel afhankelijkheden van het hoofdframe die niet bestond binnen Eclipse. Dit is opgelost door de applicatie MVC op te zetten en een hoop code te refactoren. Ten tweede kwamen er veel threading issues zodra je Swing en SWT door elkaar gaat gebruiken waardoor de applicatie vastliep. Door gebruik te maken van wrapper classes heb ik dit opgelost en de overige problemen tot een minimum kunnen beperken en werkt de volledige applicatie nu ook als Eclipse plug-in.

Als laatste kwamen de multi-user changes aan bod. Hier heb ik gekeken naar zowel de technische aspecten, functionele eisen als de usability wensen. Eerst heb ik gekeken naar de functionele eisen. Er moest namelijk een requirements editor en validator toegevoegd worden aan de applicatie. Aan de hand van het aangeleverde requirements document heb ik de gewenste functionaliteit geïmplementeerd. Om te kijken of deze gebruiksvriendelijk waren is hiervoor een usability test gepland. Aan de hand van deze test zullen we te weten komen of de gebruikers tevreden zijn met de aanpassingen. Deze resultaten zullen helaas pas beschikbaar zijn na de deadline van de scriptie.

Toen de extra functionaliteit geïmplementeerd was kon ik aan de slag met het technisch mogelijk maken met meer gebruikers tegelijk in hetzelfde project te werken. Hier heb ik goed gekeken naar de mogelijkheden om een applicatie multi-user te laten communiceren en het persisteren van data. Overwogen opties voor de communicatie van data waren web services, JMS, RMI en een centrale database. Aangezien een splitsing van de code ongewenst was vanwege de onderhoudbaarheid van de applicatie is er gekozen voor de centrale database. Hierdoor viel de keuze van het persisteren van de data natuurlijk ook op de centrale database. Andere technieken die overwogen waren zijn XML, JSON en serialisation. Voor de implementatie van de centrale database is gebruik gemaakt van JPA. JPA biedt veel voordelen omdat je direct de objecten naar tabellen kan mappen. Helaas heeft de techniek ook enkele nadelen zoals overhead en kan het niet omgaan met alle software design keuzes zoals fields van een interface en inner classes. Deze design keuzes waren wel gemaakt binnen Symbiosis. Hierdoor was een aanpassing in het model nodig. Na deze aanpassing was de weg vrij om de implementatie succesvol te laten zijn.

Summary

During my internship I've worked at ISAAC Software Solutions. The assignment was commissioned by the EQuA project. The EQuA project focusses on improving the quality of software in the software production process. Within this project there is a tool in development which focusses on the synchronisation between the requirements and the object model called Symbiosis. The main part of the assignment was to research how to make the application multi-user and implement it in the tool. Another part of the assignment was researching alternative GUI toolkits and the integration of the Symbiosis application as an Eclipse plug-in.

The first part I've taken a look at was the alternative GUI toolkit. Since there needed to be made additional views for the application I thought it was wise to look at this first. The Symbiosis application uses the Swing framework which is not actively developed so the client wanted to know what alternatives there were and how long it would take to recreate the application windows in this alternative GUI toolkit. I've taken a look at 2 alternatives namely the SWT framework and the Java FX framework. These GUI toolkits have been compared to the Swing framework and there were small prototypes build of each framework to get an indication how much time it would take. In the end we've decided to stick to the Swing framework since rebuilding the application would take too much time and there were few benefits.

Next I've taken a look at the possibilities to integrate Symbiosis as an Eclipse plug-in. Since Eclipse forces developers to use the SWT toolkit when making contributions to the user interface and Symbiosis uses the Swing framework it was needed to research if it was technically possible to use a Swing GUI with the Eclipse IDE or that a completely separate GUI was needed for the Eclipse version. After some research it seemed possible to integrate a Swing GUI in an SWT component however this would lead to some complications. First of all the Symbiosis application had lots of references to the master frame of the application which wasn't present in the Eclipse plug-in. I took care of this problem by implementing the Model View Controller design pattern in the application. Second there were lots of threading issues when using both techniques side by side. By wrapping the Swing components in custom made wrapper classes I've managed to solve lots of these problems. Today the application is completely usable from within Eclipse.

Lastly it was time to tackle the multi-user changes. It was needed to take a look at the technical side, the extra functionality and the usability of the multi-user changes. First I took a look at the functional requirements. The application needed to be extended with an requirements editor and validator. The requirements for this where delivered by the client. These requirements and associated extra windows for the user interface were implemented and needed to be tested. To see if the new requirements editor and validator were user friendly, a usability test was designed. This test will tell us if the application is user friendly and if modifications are needed. Unfortunately these results will be available after de deadline of this thesis. With all the wanted functionality implemented it was time to take a look at the technical side of multi-user application design. I needed to research the ways to let applications communicate data and the persistence of data. For the communication I've taken a look at web services, JMS, RMI and a database. All the options apart from the database required a splitting of the code into client and server. This wasn't wanted and we chose to go with the database. This would mean that the persistence choice would be easy, this would also be done with a database. Other alternatives I've looked at were XML, JSON and serialisation. The implementation of the database has been done with JPA. JPA gives you many advantages because you can map objects directly to database tables. However JPA does have its downsides like overhead and it can't cope with all the software design patterns like field of an interface and inner classes. These design choices were made in Symbiosis which required us to rewrite some of the code. After this refactoring the implementation was successful.

Begrippenlijst

Begrip	Type	Beschrijving
AWT	GUI Toolkit	Toolkit met daarin hele basic Java componenten voor GUI design.
Behaviour		Beschrijving van de methodes, getters, setters en constructoren van een class.
Constraint		Een beperking op een veld. Bijvoorbeeld dat deze uniek moet zijn.
Eclipse	IDE	Ontwikkel omgeving.
Editor window	Onderdeel Eclipse	Een Editor window wordt gebruikt binnen de Eclipse IDE om source files weer te geven.
EQuA	Projectnaam	Staat voor Early Quality Assurance in Software Production.
EXPO	Modelleer techniek	EXPO is een modelleer techniek wat EQuA als basis gebruikt.
Fact breakdown		De fact breakdown is een analyse die uitgevoerd kan worden op een fact requirement. Hieruit ontstaan fact en object types.
Fact type		Wordt gebruikt binnen de Symbiosis applicatie. Houdt informatie vast over het model, fields, constraints en overerving.
GUI	Term	Staat voor "Graphical User Interface" en hiermee worden alle grafische interfaces bedoeld waar de gebruiker mee in aanraking komt.
HQL	Taal	Staat voor Hibernate Query Language en wordt onder andere gebruikt om data uit Hibernate / JPA op te halen.
IDE	Ontwikkelomgeving	Een tool waarmee programma's ontwikkeld kunnen worden.
Java FX	GUI toolkit	GUI toolkit ontwikkeld door Oracle.
JBoss	Applicatieserver	JBoss is een applicatie server gericht op het draaien van Java EE applicaties.
JPA	Object relational mapping	Een framework om je Java objecten naar database tabellen kan mappen.
JSON	File type	JSON staat voor JavaScript Object Notation en wordt gebruikt om data over het web te sturen. Veel gebruikt in web services en mobiele telefoons voor data overdracht.
Library	Software	Externe software die gebruikt kan worden binnen een andere applicatie.
Mock-up scherm	Prototyping tool	Snel gemaakt prototype van een scherm. Wordt gebruikt om de bruikbaarheid te testen voordat de schermen echt gemaakt worden.
Multi-user		Multi-user staat simpel weg voor gebruik door meerdere mensen tegelijk.
MVC	Design patroon	Model View Controller patroon. Een patroon om de logica van de views te scheiden.
Object type		Wordt gebruikt binnen de symbiosis applicatie. Houdt informatie vast over het model, fields, constraints en overerving.
Participant	User type	Een participant is een deelnemer van een project binnen Symbiosis. Een Participant kan een projectmember of een stakeholder zijn.
Persistentie		Gegevens die na het afsluiten en heropenen van een applicatie nog steeds beschikbaar zijn. Dit worden persistente gegevens genoemd.
Projectmember	Participant type	Een projectmember is een type participant die alles kan binnen symbiosis behalve valideren van requirements.
Plug-in	Software	Een tool die binnen een andere applicatie kan draaien.
ReviewState	Staat van requirement	Een ReviewState wordt gebruikt om de staat van een requirement aan te geven en heeft de volgende opties: Approved, Rejected, Added, Removed, Add Rejected, Remove Rejected, Changed en change rejected

Begrip	Type	Beschrijving
Role		Geeft de rol van een waarde aan in een fact en object type.
Rollback		Het terugdraaien van een aanpassing.
Symbiosis	Tool	Naam van de tool in ontwikkeling bij Frank Peeters. Gebruikt als prototype tool voor requirements en model in sync te houden.
Serialisatie	Opslag methode	Het binair wegschrijven van gegevens in een lokaal bestand.
Swing	GUI Toolkit	GUI toolkit gebouwd op het AWT framework. Biedt veel uitgebreidere functionaliteit als AWT.
SWT	GUI Toolkit	GUI Toolkit gebouwd als alternatief op het Swing framework. Maakt gebruik van native library's.
Symbiosis	Naam	Naam van de tool in ontwikkeling bij Frank Peeters.
UML	Modelleer techniek	Staat voor "Unified Modeling Language" en wordt gebruikt voor het creëren van diagrammen en modellen gebruikt bij software ontwerp.
Web service	Service	Een aan te spreken service om data op te halen of weg te schrijven via XML.
Viewpart	Onderdeel Eclipse	Een dockable paneel binnen de Eclipse IDE. Viewparts kunnen volledig geprogrammeerd worden en kunnen dan allerlei functionaliteiten hebben.
XML	File type	XML is een file type die veel gebruikt word bij web services.

Inhoudsopgave

1 Inleiding

Software is in bijna alle gevallen onderhevig aan aanpassingen. Dit kan op allerlei momenten in een ontwikkel en beheer traject plaatsvinden. Vaak hebben veel changes een negatieve werking op de kwaliteit van het op te leveren product. Zodra een applicatie ontwikkeld wordt, worden hier allerlei documenten en ontwerpen voor gemaakt. Een groot probleem bij een wijziging in het ontwerp zodra deze documenten gemaakt zijn binnen een lopend ontwikkel traject is dat deze ontwerpdocumenten vaak niet geüpdate worden en hierdoor niet in lijn zijn met het product dat uiteindelijk opgeleverd wordt. Het EQuA project probeert dit probleem aan te pakken. Binnen het EQuA project zijn er meerdere gebieden die aangepakt worden. Het gebied waar de opdrachtgever Frank Peeters zich op richt is: Changing requirements & design.

Om dit probleem aan te pakken is er een tool in ontwikkeling genaamd Symbiosis. Met deze tool is het mogelijk om vanuit de requirements een objectmodel te genereren. Later is het zelfs de bedoeling dat de tool code gaat genereren naar de meest gebruikte programmeer talen. Dit project zal zich echter richten op de usability en gebruik van de applicatie door meerdere gebruikers. De Symbiosis applicatie is momenteel niet bruikbaar door meer mensen tegelijk, niet erg tijdens de ontwikkeling van de tool maar een echte hindernis zodra de tool op de markt zou komen.

Deze afstudeeropdracht is uitgevoerd bij ISAAC Software solutions welke technische ondersteuning hebben verleend bij vragen en problemen. ISAAC loopt zelf ook vaker tegen het probleem aan dat requirements veranderen en opdrachtgevers functionaliteit willen toevoegen en veranderen. Hierdoor is ISAAC geïnteresseerd in dit project en wilde hieraan meewerken. Het project heeft verder geen direct nut voor ISAAC. In hoofdstuk 2 komt de organisatie kort aan bod.

Het probleem waaruit dit project is ontstaan wordt nader beschreven in hoofdstuk 3. Hier wordt er dieper op het probleem ingegaan en wordt de doelstelling van het project beschreven.

De onderzoeksvragen gesteld bij dit project worden kort beschreven in hoofdstuk 4. Tevens wordt er hier in het kort aangegeven hoe deze vraag beantwoord zal worden. De gebruikte aanpak binnen het project wordt in hoofdstuk 5 beter uitgelegd. Binnen dit project is gebruik gemaakt van het tien stappen plan welke daar uitwerkt staat.

In hoofdstuk 6 wordt er gekeken naar mogelijkheden om de bruikbaarheid van de applicatie te verhogen, het implementeren van requirements aangeleverd door de opdrachtgever en wordt er gekeken naar GUI toolkit alternatieven. In hoofdstuk 7 wordt er verder ingegaan op de technische mogelijkheden tot het multi-user maken van de applicatie en wordt er gekeken naar de persistentie mogelijkheden. In hoofdstuk 8 wordt er gekeken naar de mogelijkheden om de Symbiosis applicatie als plug-in in Eclipse te gebruiken. Hier zal vooral naar de technische kant gekeken worden.

Als laatste wordt er in hoofdstuk 9 gekeken naar alle multi-user aspecten binnen de EQuA methodiek en welke veranderingen er door de gebruiker gemaakt kunnen worden. Dit zal van groot belang zijn binnen de multi-user aanpassingen. In hoofdstuk 10 reflecteer ik nog kort op de opgeleverde producten en sluit af met een persoonlijke evaluatie.

2 ISAAC Software Solutions B.V.

In dit hoofdstuk volgt een korte beschrijving van het bedrijf waar mijn stage opdracht heeft plaatsgevonden.

2.1 Algemene informatie

ISAAC Software Solutions is in 1999 opgericht door drie studenten van de Technische Universiteit van Eindhoven. ISAAC staat voor Internet Strategy And Automation Company. Sinds het ontstaan van ISAAC is het bedrijf enorm gegroeid en vandaag de dag telt het bedrijf 45 werknemers.

De sleutelwoorden bij ISAAC zijn:

- Innovative
- Customer Intimate
- Responsible
- Committed
- Mutuality

2.2 Producten

Als je kijkt naar de producten en diensten die ISAAC aanbied zie je dat zich met name op open source software richten. ISAAC heeft een heel compleet aanbod van diensten:

- Front end
  - Concept & design
  - Webdesign
- Back end
  - Middleware
  - Consultancy
- Mobile
- Webshops

Sinds kort heeft ISAAC ook eigen producten (applicaties) die ze verkopen op licentie basis.

2.3 Locatie

ISAAC is gevestigd in Eindhoven. Ze bezetten de vierde en vijfde verdieping van het kantoorpand Obelisk. Op de vijfde verdieping bevind zich de receptie, administratie en designers. En op de vierde verdieping bevinden zich zowel de front-end als de backend developers.

3 Opdracht: EQuA multi-user aspecten

Dit hoofdstuk beschrijft de opdracht en de achtergrond welke tot de opdracht heeft geleid.

3.1 Achtergrond

Deze stage is tot stand gekomen vanuit het EQuA project. Het EQuA project richt zich op het verbeteren van de kwaliteit van software in een zo vroeg mogelijk stadium van ontwikkeling. De afkorting EQuA staat dan ook voor **E**arly **Q**uality **A**ssurance in software production. Het EQuA project is een RAAK-PRO project met een looptijd van 4 jaar. Het project is gestart in het najaar van 2010 en loopt dus tot het najaar van 2014. Het EQuA project is een samenwerking tussen 8 partners: 2 hogescholen, 3 wetenschappelijke instellingen en 3 bedrijven. (Brunekreef, 2013)

Een van de deelprojecten van het EQuA project betreft de synchronisatie tussen requirements en objectmodel. Hiervoor is inmiddels een prototype-tool, met de naam Symbiosis, ontworpen. Aan deze applicatie heb ik al eerder gewerkt binnen een schoolproject. Dit vond ik een zeer interessant project om aan te werken en ik was aangenaam verrast toen Frank mij voor deze stageplek vroeg.

3.2 Probleemstelling

De constructie van requirements model en objectmodel wordt normaliter niet door één en dezelfde persoon uitgevoerd. Binnen een ontwerpproject zijn requirements engineers, software engineers, project management en niet te vergeten stakeholders betrokken. Soms is het praktisch dat sommige betrokkenen tegelijkertijd onderdelen uit het ontwerpproject, conform de EQuA ontwerpmethodiek, kunnen bewerken. Bewerken moet hierbij worden opgevat als feitdecompositie, typeconfiguratie, invoer, wijziging en validatie van requirements, model checking.

Met de huidige opzet van de tool is het echter niet mogelijk om met meerdere mensen tegelijk in te werken. Hier zullen dus aanpassingen gemaakt moeten worden zodat dit wel mogelijk is.

3.3 Beginsituatie

Zoals aangegeven is er een prototype-tool in ontwikkeling bij Ir. Frank Peeters. Binnen deze tool is het mogelijk om fact requirements in te voeren, te analyseren en daar een objectmodel uit te genereren. Met dit objectmodel is het dan mogelijk om een class-diagram te genereren met alle benodigde fields en methods.

De tool is ontwikkeld in Java SE en maakt gebruik van een Swing GUI. Symbiosis is een standalone applicatie gebouwd voor Windows, Mac en Linux. De applicatie is single user en er is dan ook nog geen rekening gehouden met multi-user aspecten.

3.4 Doelstelling

Binnen 20 weken moet de huidige versie van de tool compleet verwerkt zijn als Eclipse plug-in. Tevens moet de tool multi-user gemaakt worden en er dus meerdere gebruikers tegelijk in de applicatie kunnen werken. Als laatste zal de tool een eenvoudig te gebruiken requirements editor hebben. Aan het einde van week 17 zullen de resultaten van dit project worden gepresenteerd. De laatste 3 weken zullen voor nazorg gereserveerd worden.

3.5 Resultaten

Hier onder vindt u een overzicht van de hoofdpunten. Voor een gedetailleerde lijst van producten verwijs ik u door naar de **W**ork **B**reakdown **S**tructure (WBS) binnen het PID (Bijlage A).

- Scriptie (met onder andere):
    - Mogelijkheden onderzoeken om geen gebruik meer te maken van Swing en over te stappen naar modernere GUI techniek
    - Onderzoek MVC maken applicatie
    - Onderzoek multi-user maken applicatie
- Symbiosis integreren als plug-in in Eclipse
- Multiuser versie Symbiosis
- Symbiosis uitbreiden met MVC patroon
- Documentatie
- Gebruiksvriendelijke requirements model editor en –validator (binnen de tool)

4 Onderzoek

5.1 Onderzoeksvragen:

5.1.1 Hoofdvraag:

H1 - Welke multi-user activiteiten, rekening houdend met de EQuA methodiek, zijn benodigd of kunnen beter vermeden worden tijdens het ontwerpen van het domeinmodel?

Om antwoord te krijgen op deze vraag zullen we deze opdelen in enkele deelvragen. In hoofdstuk 9 zal de hoofdvraag nader bekeken worden.

5.1.2 Deelvragen:

Deelvraag 1:

1 - Hoe kunnen we de bruikbaarheid van de Symbiosis applicatie voor requirement engineers, ontwikkelaars en stakeholders verbeteren door de user interface te veranderen?

Om antwoord te krijgen op deze vraag zal ik de ontwerpmethoden van het boek "User interface design" van (Leuesen, 2005) gebruiken voor ontwerp richtlijnen. Om te verifiëren dat de bruikbaarheid verbeterd is zal er een vernieuwde versie van de GUI getest worden onder ontwikkelaars en een software engineer klas op de Fontys Hogeschool en zal worden gekeken of het aan de gestelde kwaliteitseisen voldoet.

Aan deze vraag zitten nog enkele sub vragen vast.

1.1 – Welke GUI mogelijkheden zijn er en wat zijn de voor en nadelen hiervan?
1.2 - Hoe kunnen we de bruikbaarheid van de Symbiosis applicatie voor requirement engineers, ontwikkelaars en stakeholders verbeteren door de user interface te veranderen?

Dit deel van het onderzoek wordt verder uitgewerkt in hoofdstuk 6.

Deelvraag 2:

2 - Wat is de beste manier om projectleden te laten samenwerken binnen de tool terwijl ze parallel aan hetzelfde project werken, zonder vertragingen of data corruptie veroorzaakt door de multi-user conflicten?

Aan deze vraag zitten nog enkele sub vragen vast.

2.1 - Wat is de impact als er een requirement gewijzigd wordt?

Om antwoord te krijgen op deze vraag zal ik goed de bestaande code en kijken wat er allemaal kan veranderen (ook in andere onderdelen van het requirement en object model) zodra alle mogelijke wijzigingen op een requirement worden uitgevoerd. Dit deel van het onderzoek zal in hoofdstuk 7.2 worden behandeld.

2.2 - Hoe kun je het beste meerdere gebruikers met dezelfde data, geproduceerd door Symbiosis, laten werken op verschillende locaties?

Om hier inzicht in te krijgen zal ik een vergelijking maken tussen een aantal opties die deze mogelijkheden bieden. Dit zal worden besproken in hoofdstuk 7.3

2.3 - Hoe kun je het beste de data opslaan als er meerdere gebruikers tegelijk in werken?

Om hier antwoord op te kunnen geven zal ik een vergelijking maken tussen een aantal opties die mogelijkheden bieden om de data op te slaan. Dit zal worden besproken in hoofdstuk 7.4

Deelvraag 3:

3 - Hoe kunnen we de Symbiosis applicatie binnen Eclipse integreren?

Aan deze vraag zitten nog enkele sub vragen vast.

3.1 - Wat is de techniek gebruikt voor Eclipse plug-ins en is deze compatible met de Symbiosis applicatie?

Om hier antwoord op te kunnen geven zal ik eerst desk research doen en hierna een prototype maken zodat ik een goed beeld krijg van de mogelijkheden en beperkingen van het Eclipse platform. Dit zal worden besproken in hoofdstuk 8.2.

3.2 - *In hoeverre is het mogelijk een Swing GUI binnen SWT te laten draaien?*

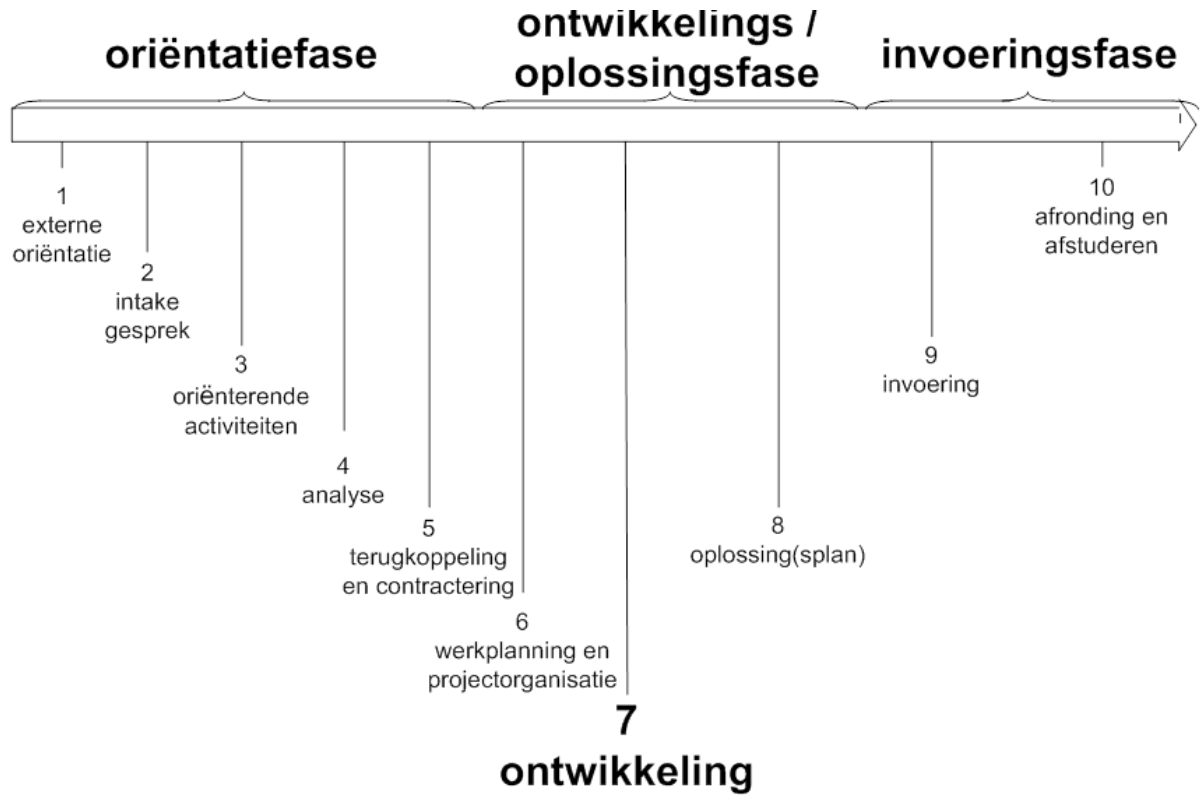
Om hier antwoord op te krijgen zal ik, net als bij de vorige deelvraag beginnen met desk research en daarna een prototype maken. Dit zal ik zo proberen op te zetten dat er niet voor alles nieuwe code geschreven moet worden. Uitwerkingen van dit wordt besproken in hoofdstuk 8.3

5 Aanpak

Dit hoofdstuk bespreekt de gekozen aanpak binnen het project. Van het zoeken van de stageplaats tot het afronden ervan. De planning van het project is ook te vinden in dit hoofdstuk.

5.1 Tien stappen plan

Vanuit de opleiding werd er sterk aangeraden om met het tien stappen plan te gaan werken. Het tien stappen plan is een hulpmiddel bij stages en heeft, zoals de naam zegt, 10 stappen die de stagiair begeleiden. Onderstaande afbeelding geeft een globaal overzicht van het 10 stappen plan aangepast op software ontwikkelaars. Het 10 stappen plan zal later in dit hoofdstuk in detail worden beschreven.



5.2 Projectopzet

Binnen het project is met scrum gewerkt. Om goed in te kunnen springen op wijzigingen vanuit de opdrachtgever is besloten met sprints van 2 weken te werken. Deze sprints werden bijgehouden op de online tool Acunote. Binnen deze tool is de dagelijkse planning en de urenverantwoording opgenomen.

5.3 Tien stappen plan: stap voor stap

5.3.1 Stap 1: Externe Oriëntatie

Het doel van de externe oriëntatie is een inzicht krijgen in de drijfveren en belangen van de opdrachtgever. Ook oriënteren op het bedrijf en de bedrijfstak hoort hier bij.

Voor ik met het afstuderen begon werd ik aangesproken door Ir. Frank Peeters, de opdrachtgever binnen dit project. Frank had de opdracht bij ISAAC neergelegd voor het door ontwikkelen van de Symbiosis applicatie. Dit leek mij een erg leuke opdracht om aan te werken aangezien ik al eerder binnen dit project gewerkt had en het een leuke technische uitdaging had.

Na het lezen van de vacature, de voorwaarden en goed te hebben gekeken naar ISAAC heb ik besloten te solliciteren op de stage. Hier kreeg ik snel reactie op en kreeg direct een uitnodiging om een keer op het bedrijf te komen kijken.

5.3.2 Stap 2: Intake gesprek

Het doel van het intake gesprek is dat zowel de stageplek als de stagiair een goed beeld van elkaar kunnen krijgen en onduidelijkheden in de opdracht weg kunnen werken.

Tijdens het intake gesprek had ik meteen een goed gevoel bij het bedrijf. Iedereen was heel vriendelijk en goed op de hoogte van de opdracht. Aangezien ik al goed op de hoogte was van de opdracht en een aanbeveling van Frank had gekregen was het meer een formaliteit.

5.3.3 Stap 3: Oriënterende activiteiten

Het doel van de oriënterende fase is het in kaart brengen van de bestaande situatie, het inlezen in de opdracht, software en hardware, interviews houden met betrokkenen en een mening vormen over de behoefte van de opdrachtgever.

Aangezien ik al eerder aan de applicatie gewerkt had was deze stap niet erg moeilijk. Hierdoor kende ik een groot deel van de code al en aangezien ik al eerder met de opdrachtgever overlegd had wist ik ook wat hij verwachtte. Om mijn begeleider wat meer inzicht te geven en mijn kennis van de applicatie te verversen hebben we in de eerste weken 2 workshops gehad van de opdrachtgever om te leren hoe de tool werkt en wat de mogelijkheden zijn.

5.3.4 Stap 4: Analyse

Het doel van deze fase is de opdracht goed kunnen definiëren, het opstellen van een Project Initiatie Document (PID) en het maken van een planning.

Deze stap ging heel voorspoedig. Na het opzetten van het PID is deze doorgenomen door mijn projectbegeleider en hier waren enkele kleine aanmerkingen op. Na het verbeteren van deze punten was hij direct akkoord met het PID en nadat ik deze opgestuurd had naar mijn docent begeleider werd deze ook goedgekeurd.

5.3.5 Stap 5: Terugkoppeling en contractering

Het doel van deze stap is een goedkeuring krijgen op het PID en het eerste bedrijfsbezoek van de docentbegeleider. Hier kunnen ook afspraken gemaakt worden over terugkoppel momenten in de toekomst.

Dit gesprek ging prima, echter waren er nog enkele onduidelijkheden met betrekking tot het onderzoekscomponent binnen de stage. Na dit besproken te hebben met de docent begeleider was het een stuk duidelijker. De afspraken met de docent begeleider waren al gemaakt dus hier hoefde verder geen actie ondernomen worden.

5.3.6 Stap 6: Werkplanning en projectorganisatie

Het doel van deze stap is het uitwerken van de planning.

Dit loopt bij mijn project iets anders aangezien ik agile met sprints van 2 weken werk. Hierdoor kan ik eenvoudig aanpassingen maken in de planning en zo inspelen op de directe wensen van de opdrachtgever. Uiteraard heb ik geprobeerd zoveel mogelijk aan de globale planning te houden voor zo ver mogelijk.

5.3.7 Stap 7: Ontwikkeling

Het doel van deze stap is het ontwikkelen van de applicatie en het uitvoeren van het onderzoek.

Deze stap is heel voorspoedig gelopen. Het ontwikkelen van de Eclipse applicatie is zeer voorspoedig gegaan en was hier voor op schema mee klaar. De requirement editor changes namen echter iets meer tijd in beslag door missende requirements en use cases in het aangeleverde requirements document. Aangezien ik tijd over had van de Eclipse plug-in was ik hier alsnog prima op schema mee klaar.

Na de Eclipse plug-in was het tijd om met de multi-user requirements aan de slag te gaan. De multi-user changes zijn eigenlijk verdeeld over 3 items. Het samenwerken met dezelfde data, het centraal beschikbaar maken van de data en extra functionaliteit te implementeren.

5.3.8 Stap 8: Oplossingsplan

Het doel van deze stap is afspraken maken betreffende de overdracht van het werk en de invoering hiervan.

Hiervoor hebben de opdrachtgever en ik een afspraak gemaakt zodat de overdracht zo soepel mogelijk zou verlopen.

5.3.9 Stap 9: Invoering

Het doel van deze stap is het behoud van het product en opgedane kennis voor de opdrachtgever.

Voordat mijn stage afgelopen was heb ik er voor gezorgd dat de opdrachtgever de technische implementatie van JPA en de multi-user aspecten bekend was met de implementatie en hebben we samen een hoop code doorlopen om te kijken of er nog vraagstukken waren.

Voor overdracht van de Eclipse kennis is er in hoofdstuk 8 een passage opgenomen hoe de Eclipse plug-in opgezet is en hoe deze te gebruiken. Tevens stel ik mij na mijn stage periode nog 3 weken beschikbaar voor support en overdracht. In deze tijd zal ik geen nieuwe changes doorvoeren.

5.3.10 Stap 10: Afronding en afstuderen

Het afronden van de scriptie en de presentatie staan centraal in deze stap.

De resultaten van de dit project zal ik presenteren tijdens mijn afstudeerzitting welke op 26 juni ’13 zal plaatsvinden op de Fontys Hogeschool ICT in Eindhoven.

6 Deelvraag 1: De bruikbaarheid verhogen

6.1 Onderzoeksvragen

1 - Hoe kunnen we de bruikbaarheid van de Symbiosis applicatie voor requirement engineers, ontwikkelaars en stakeholders verbeteren door de user interface te veranderen?

1.1 – Welke GUI mogelijkheden zijn er en wat zijn de voor en nadelen hiervan?

1.2 - Hoe kunnen we de bruikbaarheid van de Symbiosis applicatie voor requirement engineers, ontwikkelaars en stakeholders verbeteren door de user interface te veranderen?

6.2 Welke GUI mogelijkheden zijn er en wat zijn de voor en nadelen hiervan?

Om antwoord te kunnen geven op deze vraag moeten we eerst weten op welk platform we gaan werken. Het ontwikkel platform dat gebruikt is in de huidige Symbiosis applicatie is het Swing platform. Swing is een toevoeging op het oude AWT framework en is geschreven in Java SE. De applicatie wordt ontwikkeld voor de Windows, Linux en Mac platformen.

Nu dit bekend is kunnen we gaan kijken naar de GUI opties. Naast de gebruikte Swing GUI zijn er namelijk nog veel meer opties met hun eigen voor en nadelen. Hieronder is een overzicht van de veelgebruikte GUI technieken gericht op Java SE applicaties. Hierbij wordt rekening gehouden met de wensen voor de tool, de huidige staat van de applicatie, de toekomstbestendigheid en leer curve.

6.2.1 Swing

Swing is een veelgebruikt framework in de Java SE industrie gebouwd bovenop het AWT framework. Op de opleiding software engineering op de Fontys wordt ook gebruikt gemaakt van swing componenten. Swing is ook een lange tijd de officiële GUI techniek op het Java SE platform geweest. Voor informatie over Swing heb ik gebruik gemaakt van het boek "Java Swing, O'Reilly" (Elliott, 2002) en heb ik gebruik gemaakt van verschillende internet bronnen. (authors, 2013), (Fowler)

Voordelen	Nadelen
- De huidige GUI van Symbiosis is geschreven in Swing.	- Legacy GUI techniek, vervangen door JavaFX 2.x
- Swing is een volwassen platform met veel 3th party library's beschikbaar.	- Hoge leer curve, echter al veel ontwikkelaars mee bekend.
- Swing werkt goed op alle target platformen.	- Relatief trage GUI door software emulatie, dit is echter met de hedendaagse computers geen probleem.
- Meest gebruikte GUI techniek	- Werkt niet direct samen met Eclipse door het hanteren van een ander threading model als SWT.
- Gemakkelijk ontwikkelaars te vinden die bekend zijn met Swing.	

6.2.2 SWT

SWT is een GUI techniek gemaakt door IBM als alternatief op AWT en Swing. SWT is ook gebruikt binnen de Eclipse IDE. SWT gebruikt een ander threading model als AWT/Swing. SWT heeft net als Swing GUI builders en is eenvoudig in gebruik. Voor informatie over SWT heb ik gebruik gemaakt van het boek "The definitive guide to SWT and JFace" (Harris, 2004) en enkele internet bronnen. (Vogel, 2012), (authors, 2013), (Eclipse)

Voordelen	Nadelen
- SWT maakt gebruik van native JNI calls en gebruikt de native look en feel	- Symbiosis is in Swing gemaakt, vereist het herschrijven van alle GUI code.
- Eenvoudig te integreren als Eclipse plug-in.	- Minder ontwikkelaars bekend met deze techniek.
- Werkt goed op de target platformen, heeft alleen een aparte library nodig per platform.	- Moet voor elk platform apart gebuild worden.
- Lage leercurve	- Resources moeten handmatig vrijgegeven worden.
- Veel 3th party library's	
- Heeft AWT bridge voor gebruik Swing componenten (beperkt en veel haken en ogen)	

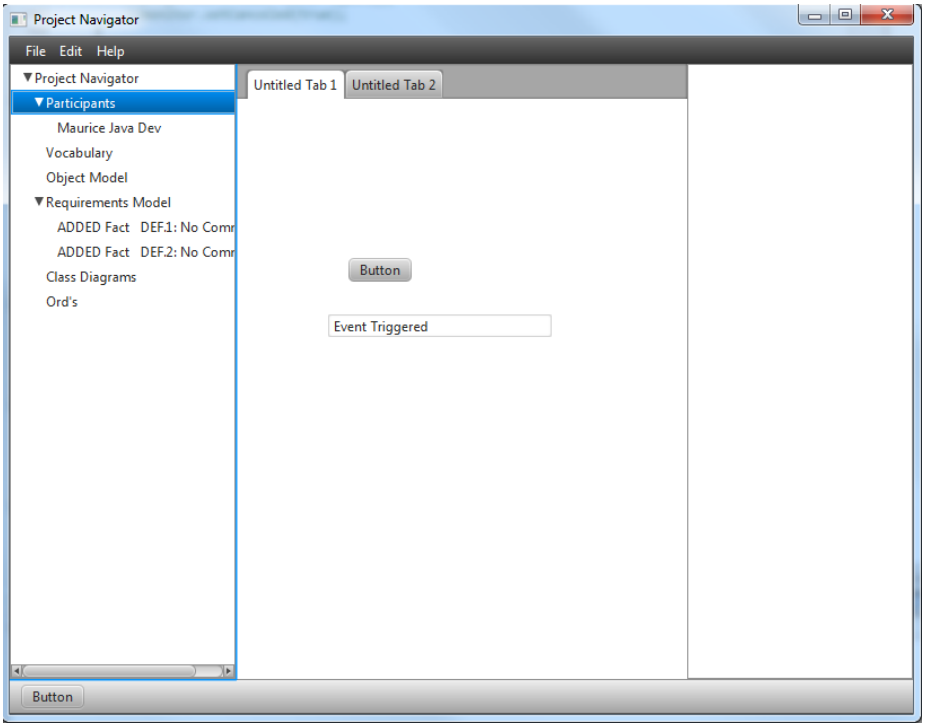
6.2.3 JavaFx 2.x

Na een moeilijke start van het Java FX platform als web script taal hebben ze de hele taal omgegooid en wordt deze nu gebruikt als GUI techniek voor Java SE applicaties. Java FX is momenteel de officiële GUI techniek van Java en wordt actief onderhouden door Oracle. Sinds Java SE 7 update 6 is Java FX opgenomen binnen de JDK en kan dus native gebruikt worden. Voor informatie over Java FX 2 heb ik gebruik gemaakt van het boek "Pro JavaFX 2: A Definitive Guide to Rich Clients with Java Technology" (Chin, 2012) en enkele internet bronnen. (Oracle), (Oracle BV), (authors, 2013)

Voordelen	Nadelen
- Java FX maakt gebruik van hardware acceleratie	- Symbiosis is in Swing gemaakt, vereist het herschrijven van alle GUI code.
- Relatief eenvoudig te integreren als Eclipse plug-in, minder goed als SWT, beter als Swing.	- Minder ontwikkelaars bekend met deze techniek. Nog relatief nieuw.
- Werkt goed op de target platformen.	- Nieuwe techniek, zitten nog wat kinderziektes in.
- Lage leercurve	
- Heeft AWT bridge voor gebruik Swing en SWT componenten (beperkt en veel haken en ogen)	
- Heel toekomstbestendig	

6.2.4 Prototypes

Zoals te zien in de tabellen in de vorige paragrafen heeft elke techniek wel voor en nadelen. Aangezien Java FX en SWT beiden zeer valide alternatieven zijn voor Swing heb ik voor beiden GUI technieken een prototype gemaakt van een onderdeel van de applicatie en deze herschreven in de nieuwe GUI techniek. Hierdoor kon ik een goede inschatting maken over hoe lang het omschrijven van de applicatie zou duren mochten we hiermee doorgaan. De simpele componenten waren vlug nagemaakt maar er waren ook een groot aantal complexe componenten en deze zouden zeker veel tijd in beslag gaan nemen.



[Voorbeeld Java FX prototype]

6.2.5 Conclusies en aanbevelingen

Na het doornemen van alle resultaten met mijn begeleider waren we het er over eens dat de huidige Swing GUI prima voldoet en het veel teveel tijd gaat kosten om de gehele GUI om te gooien naar Java FX of SWT en de voordelen wegen niet op tegen de tijd die erin gestopt dient te worden. Ook heb ik inmiddels een AWT/SWT wrapper class geschreven waardoor Swing ook te gebruiken is binnen Eclipse met enkele beperkingen (deze worden verder besproken in hoofdstuk 9). Dit voldoet nog aan de eisen van de opdrachtgever.

De opdrachtgever heeft ook ingestemd met de aanbeveling en we zullen Swing blijven gebruiken.

6.3 Hoe kunnen we de bruikbaarheid van de Symbiosis applicatie voor requirement engineers, ontwikkelaars en stakeholders verbeteren door de user interface te veranderen?

Aangezien bruikbaarheid, ook wel usability genoemd, niet voor iedereen gelijk is zal ik me hier aan de gestelde richtlijnen voor GUI ontwerpen houden. Hiervoor gebruik ik het boek "User interface design" (Leuesen, 2005). In dit boek beschrijft Leuesen algemene richtlijnen waar je op moet letten voor een goede user interface.

Om te kunnen verifiëren of de usability is verbeterd zal de nieuwe user interface getest moeten worden. Voor deze test was een klas beschikbaar op de Fontys Hogeschool Eindhoven die begin Mei 2013 met de tool aan de slag zouden moeten zijn gegaan. Helaas is deze test enkele weken vertraagd waardoor de resultaten niet op tijd voor de laatste inleverdatum van mijn scriptie beschikbaar zijn. Na de ervaringen met de tool wordt er een enquête afgenomen onder de leerlingen. De resultaten hiervoor zullen tijdens mijn presentatie wel beschikbaar zijn. Met deze steekproef kunnen we controleren of de ontwikkelaars tevreden zijn met de nieuwe GUI.

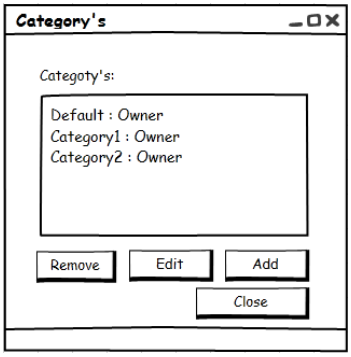
6.3.1 Ontwerpen user interface

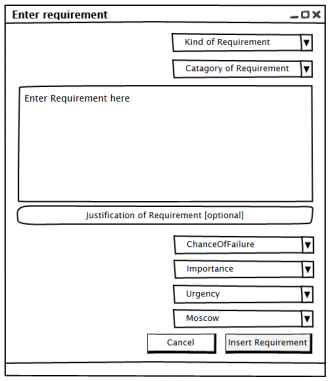
Om er achter te komen welke views er gemaakt moeten worden zullen we eerst goed naar de requirements en use cases moeten kijken en wat er precies allemaal gevraagd wordt. De requirements en use cases waren op voorhand aangeleverd echter zijn gaande de stage vorderde nog aangepast/aangevuld na overleg met de opdrachtgever en missende functionaliteiten. De laatste versie van het requirements document is als bijlage B opgenomen.

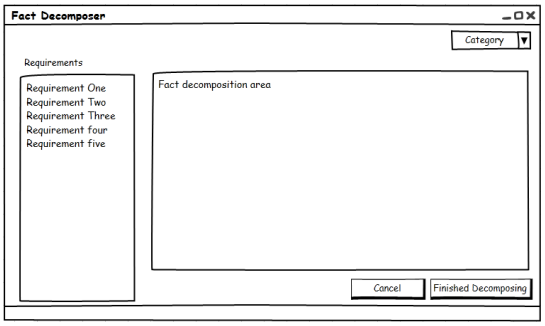
Na het doornemen en bespreken van het document had ik een aardig beeld van wat de opdrachtgever verwachtte. Nu kon ik dus aan de slag met het maken van mock-up schermen. Hiervoor heb ik de tool Mockup Builder gebruikt om een "screen prototype" te maken.

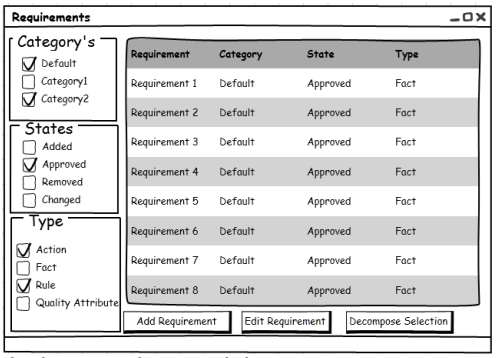
Screen prototype. The screens are shown on the real computer screen, but they have little functionality. Typically, the user may enter data into some of the fields, but when he pushes a button or selects a menu point, nothing happens by itself. Instead, the designer has built a standard functionality into the system that allows him to use some 'secret' keys to open and close specific windows. (Leuesen, 2005)

Hieronder vindt je een overzicht van enkele van de gemaakte screen prototypes. Deze mock-up schermen zijn voorgelegd aan de opdrachtgever en er is gekeken of alle functionaliteit erin zat en of de indeling prettig was. De opdrachtgever had enkele opmerkingen welke ik meegenomen heb in mijn uiteindelijke design.









Het uiteindelijke design heb ik direct binnen de ontwikkel omgeving gemaakt zodat hier enkel de functionaliteit nog aan toegevoegd diende te worden. Aangezien deze ook nog binnen de Eclipse ontwikkel omgeving gebruikt moesten worden en hier ook beperkingen aanzitten (welke beschreven worden in hoofdstuk 8) zijn de vensters als JPanel gemaakt. Deze kunnen op een Frame geplaatst worden voor de Eclipse plug-in en als dockable worden verwerkt binnen de standalone versie van Symbiosis.

6.3.2 Functionaliteit

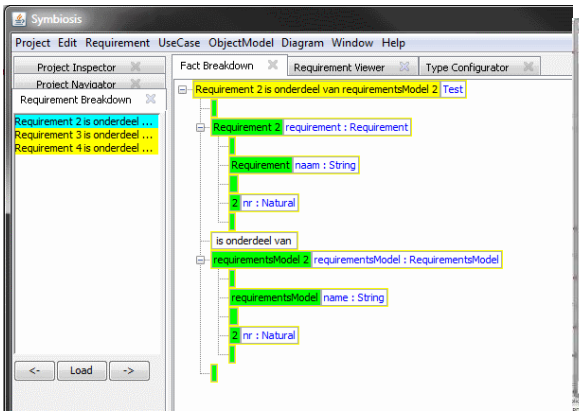
Nu de views een design hadden kon ik beginnen met het implementeren van de functionaliteit van de nieuwe views. De categorie en requirement views waren vlug gemaakt aangezien de functionaliteit hier beperkt is. Het meeste werk waren de requirement viewer en de fact breakdown. De requirement viewer had een requirement dat er eenvoudig gefilterd kon worden op allerlei verschillende aspecten van de requirement en reviewstate. Later zijn hier nog een aantal extra filters bijgekomen namelijk de "Owner" en "Realized" filters.

Vanuit de requirement viewer moest het ook mogelijk zijn de ReviewState van een requirement aan te passen, een bestaande requirement aan te passen en nieuwe requirements in te voeren. Voorheen was het niet mogelijk requirements in te voeren zonder direct een breakdown uit te voeren. Dit is nu losgekoppeld om Stakeholders de mogelijkheid te geven om een requirement goed te keuren voordat deze direct in het model opgenomen wordt. De fact breakdown moest hierdoor ook flink aangepakt worden.

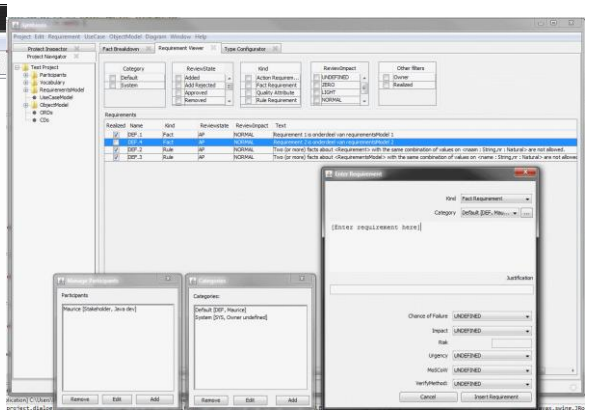
6.3.3 Fact Breakdown

Nu de fact breakdown gescheiden is van het invoeren van requirements moest er een nieuwe manier komen om dit te initialiseren. Ook was er de wens dat de fact breakdown niet telkens opnieuw geïnitialiseerd hoeft te worden. Om deze reden is het mogelijk gemaakt om meerdere requirements te selecteren en deze direct na elkaar af te handelen.

In mijn originele ontwerp zat dit in een los venster. Om het meer een geheel te maken heb ik deze later verwerkt als dockable in het docking framework van de applicatie. Nu is het één geheel in plaats van allemaal losse vensters.



[Fact breakdown]



[Requirement viewer]

6.3.4 Usertest

Om te testen of de interface voldoet aan de kwaliteitseisen gesteld in het requirements document (Bijlage B) is hier een test voor opgezet. Deze test wordt afgenomen binnen de Excellent klas van de Fontys Hogeschool ICT Eindhoven. Na het uitvoeren van de test worden de gebruikers gevraagd een enquête in te vullen met hun bevindingen. In deze enquête komen alle kwaliteitsattributen aan bod en enkele extra vragen met betrekking tot de achtergrond van de afnemer. De enquête en bijbehorende test case is opgenomen in bijlage C.

Zoals al aangegeven in het begin van dit hoofdstuk komen de resultaten niet op tijd om nog binnen deze scriptie te verwerken. Deze zal ik echter tijdens mijn presentatie wel beschikbaar hebben.

6.3.5 Conclusie en aanbevelingen

Bij het ontwerpen van de nieuwe schermen is er zoveel mogelijk rekening gehouden met de wensen van de opdrachtgever en zijn de richtlijnen van het boek “User Interface design” (Leuesen, 2005) zoveel mogelijk aangehouden. Totdat de nieuwe user interface getest is kan ik echter nog niet stellen dat de interface aan de kwaliteitseisen voldoet.

7 Deelvraag 2: Samenwerken

Aangezien het samenwerken binnen de applicatie een zeer belangrijk doel is van de applicatie zullen we moeten onderzoeken hoe dit het beste aangepakt kan worden. We zullen de bestaande applicatie moeten onderzoeken en de technieken die multi-user mogelijk maken bekijken.

7.1 Onderzoeksvragen

2 - Wat is de beste manier om projectleden te laten samenwerken binnen de tool terwijl ze parallel aan hetzelfde project werken, zonder vertragingen of data corruptie veroorzaakt door de multi-user conflicten?

2.1 - Wat is de impact als er een requirement gewijzigd wordt?

2.2 - Hoe kun je het beste meerdere gebruikers met dezelfde data, geproduceerd door Symbiosis, laten werken op verschillende locaties?

2.3 - Hoe kun je het beste de data opslaan als er meerdere gebruikers tegelijk in werken?

7.2 Wat is de impact als er een requirement gewijzigd wordt?

Om hier een eenduidig antwoord op te kunnen geven zullen we eerst wat af moeten bakenen.

7.2.1 Afbakening

Wat wordt bedoeld met "impact"?
Impact moet hierbij worden opgevat als wat voor veranderingen het teweeg kan brengen **binnen** de applicatie. Hier wordt dus niet mee bedoeld wat voor impact een requirement kan hebben binnen een project of looptijd.

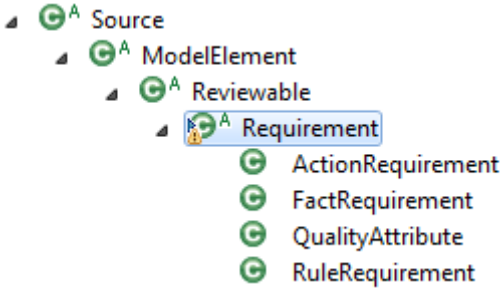
Wat wordt verstaan onder "requirement"?
Met "requirement" wordt een requirement ingevoerd in de Symbiosis tool bedoeld. Dit kan zijn een Action, Fact, Quality Attribute of een Rule requirement. Hiermee wordt dus niet bedoeld op project niveau maar op applicatie niveau.

Wat wordt bedoeld met een "wijziging" van een requirement?
Wijziging moet hierbij worden opgevat als feitdecompositie, typeconfiguratie, invoer, wijziging (inhoud) en validatie van requirements, model checking.

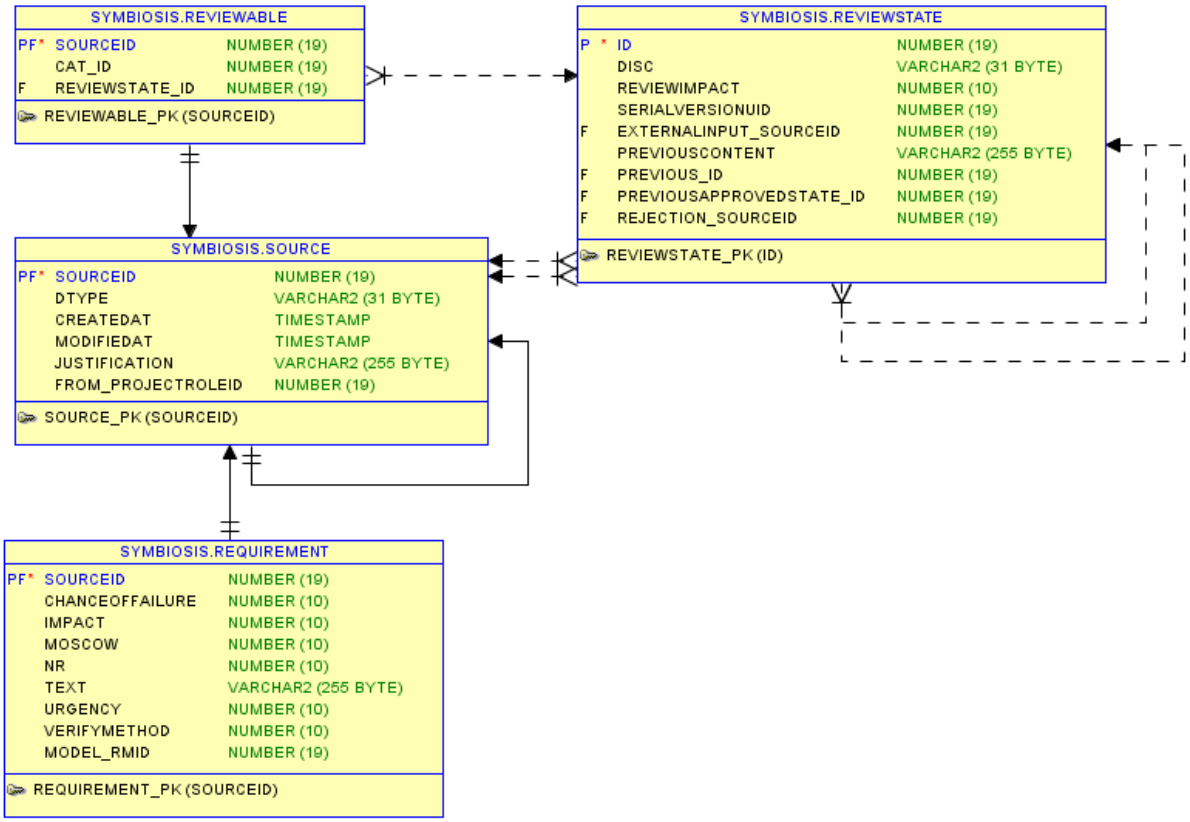
7.2.2 Opzet requirement

Om te weten wat voor veranderingen er kunnen plaatsvinden als er een requirement aangepast wordt moeten we weten hoe een requirement in elkaar zit in de applicatie.

Onderstaand figuur geeft de opzet van een requirement weer.

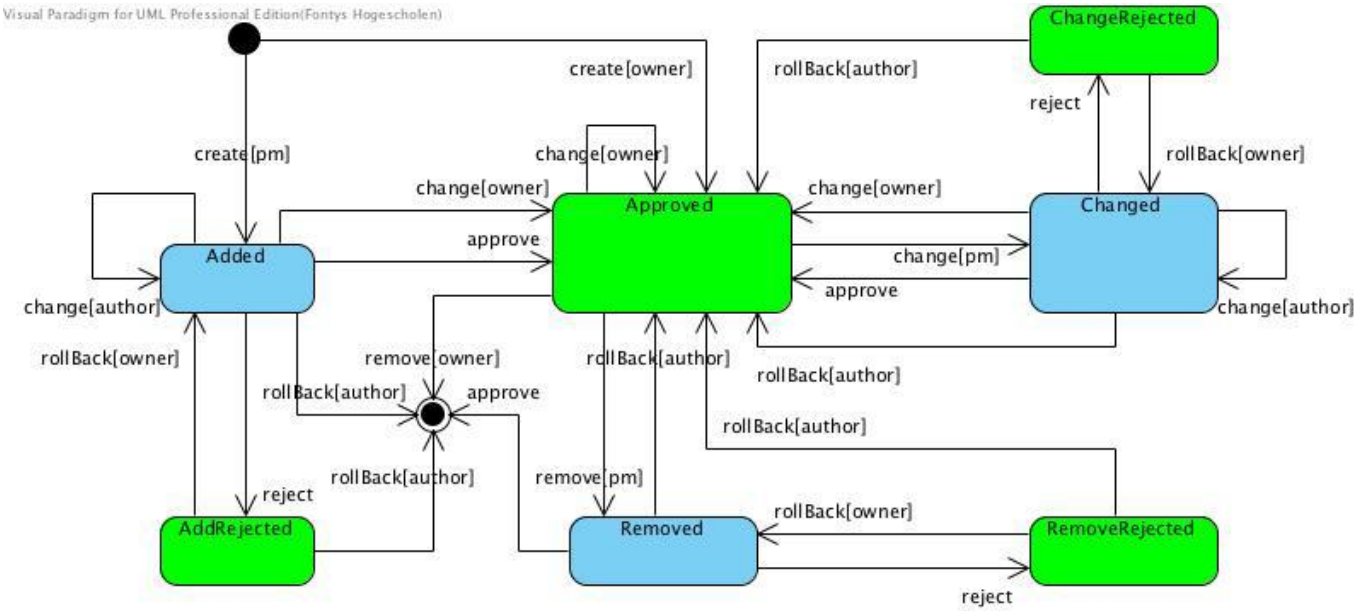


De velden die van belang zijn bij een requirement staan in onderstaand database diagram weergegeven. De belangrijkste hierbij zijn de Tekst van de requirement en de gelinkte reviewstate. Zodra deze veranderen kan dit een flinke impact hebben op de status van de requirement.



7.2.3 Wat gebeurd er met de requirement zodra er een aanpassing gemaakt wordt?

Om dit duidelijk in beeld te krijgen zullen we per mogelijke actie bekijken wat er aangepast wordt. Aangezien er meerdere rollen zijn binnen de applicatie zullen we ook moeten kijken of er verschillen zijn zodra er een aanpassing gemaakt wordt in een requirement door een andere rol. Er wordt onderscheid gemaakt tussen 3 verschillende rollen: Projectmember, Projectmember (Author) en Stakeholder. Hieronder is een weergave het geïmplementeerde state transition diagram van de reviewstates van een requirement.



Zodra er een wijziging wordt gemaakt aan de requirement zal dit een wijzigingsverzoek sturen die beoordeeld moet worden door een stakeholder. Er zal dus niet direct een wijziging doorgevoerd worden tenzij het door de eigenaar van de categorie van de requirement gebeurd. Door deze implementatie kunnen er niet direct door meerdere mensen changes gemaakt worden zonder dat deze beoordeeld zijn door de desbetreffende stakeholder. Zodra er echter een wijziging gemaakt wordt aan een requirement die al gerealiseerd is (lees: al in het domein opgenomen is) zal de inhoud niet zomaar aangepast mogen worden. Zodra deze aangepast wordt gaat de link met de FactType verloren en zullen er inconsistenties ontstaan in het model. Om dit te voorkomen zijn er 2 opties:

- Blokkeren
- Rollback breakdown en breakdown opnieuw uitvoeren

7.2.4 Conclusies en aanbevelingen

De impact van veranderingen aan de requirement is in de meeste gevallen erg laag. Zo kan alleen de eigenaar van een requirement direct een verandering doorvoeren en sturen de overige participants een wijzigingsverzoek naar de eigenaar van de requirement. De eigenaar kan in deze situatie bepalen wat hij met het verzoek doet. Zodra een requirement gerealiseerd is veranderd de situatie en kan deze niet zomaar aangepast worden zonder dat er inconsistenties in het model komen. Dit kan voorkomen worden door deze actie te blokkeren of de realisatie terug te draaien en dan de gewenste aanpassingen te maken.

7.3 Hoe kun je het beste meerdere gebruikers met dezelfde data, geproduceerd door Symbiosis, laten werken op verschillende locaties?

Om te kijken op wat voor manier we het beste de applicatie op meerdere plekken beschikbaar te maken zullen we moeten onderzoeken welke manieren er zijn om een applicatie multi-user te maken. Elke manier heeft zo zijn eigen voor en nadelen. Hieronder heb ik een overzicht gemaakt van de door mij bekeken technieken. Voor informatie over de gerefereerde artikelen heb ik gebruik gemaakt van de volgende bronnen:

"Java persistence with hibernate" (Christian Bauer, 2006)
"Create stand-alone Web services applications" (Lam & Robertson, 2008)
"Pro JPA 2: Mastering the Java™ Persistence API" (Mike Keith, 2009)

7.3.1 Java Standard Edition: Database (remote)

Een centrale database is natuurlijk een handige manier om met meer mensen dezelfde data te gebruiken. Door de vele overervingen in het model is het echter de vraag of het snel genoeg blijkt zodra er meer mensen in de applicatie gaan werken. Dit is bij databases een relatief zware operatie door de vele joins die uitgevoerd moeten worden.

Voordelen	Nadelen
Centrale opslag	Veel werk om te implementeren
Mogelijkheid tot locken van data	Internet/intranet is een vereiste
Eenvoudig met meer mensen in hetzelfde project werken	Database moet instelbaar zijn voor de gebruikers (extra werk)
De database kunnen andere applicaties eventueel ook nog gebruik van maken.	

7.3.2 Java Enterprise Edition: Web-services / JMS

Java EE is een veelgebruikte techniek om grootschalige Enterprise applicaties te schrijven en heeft ook de mogelijkheid tot het wegschrijven naar een database. Een groot voordeel is dat de data door 1 applicatie aangepast wordt in tegenstelling tot clients die direct met de database praten. Helaas zitten er ook een hoop nadelen aan Java EE welke hieronder beschreven staan.

Voordelen	Nadelen
Heel schaalbaar	Klanten zullen een applicatieserver moeten draaien.
Eenvoudig opzetten web-services / JMS	Internet/intranet is een vereiste
Eenvoudig met meer mensen in hetzelfde project werken	Redelijk wat configuratie door de gebruiker.
Web-services zouden later andere applicaties van data kunnen voorzien.	Huidige versie van de applicatie is Java SE. Dit zou moeten worden herschreven naar service based.

7.3.3 Java Standard Edition: Web-services / JMS

Web-services en JMS zijn niet gebonden aan Java EE maar kunnen ook in Java SE gebruikt worden. Dit heeft echter een flinke impact op het design van de applicatie. Zo heb je een server nodig wat fragmentatie van de code zou betekenen.

Voordelen	Nadelen
Slechts de server heeft directe toegang tot de data.	Veel werk om te implementeren
Web-services zouden later andere applicaties van data kunnen voorzien.	Internet/intranet is een vereiste
Eenvoudig met meer mensen in hetzelfde project werken.	Lastig instelbaar door gebruikers
	Code fragmentatie
	Niet schaalbaar

7.3.4 Java Standard Edition: Remote Method Invocation

De laatste methode die ik bekeken heb is RMI. Met RMI kun je directe methodes aanroepen binnen andere Java applicaties. Ook hier heb ik de voor en nadelen van bekeken.

Voordelen	Nadelen
Slechts de server heeft directe toegang tot de data.	Code fragmentatie
Eenvoudig te realiseren voor middel van enkele interfaces te definiëren.	Internet/intranet is een vereiste
Eenvoudig met meer mensen in hetzelfde project werken	Niet schaalbaar
	Lastig instelbaar door gebruikers

7.3.5 Conclusie

Als we enkel kijken naar welke techniek het meest geschikt is voor het multi-user maken van de applicatie en buiten beschouwing laten hoe de applicatie nu is opgebouwd dan zou Java EE de meest logische optie zijn. Java EE is een extreem schaalbare optie en zou later dus ook eenvoudig door grotere bedrijven in gebruik genomen kunnen worden. Een ander voordeel van Java EE is dat het een centrale applicatie is en alle changes via de server zouden lopen.

Het is echter zo dat de applicatie al compleet geschreven is in Java Standard Edition en het ombouwen hiervan enorm veel tijd gaat kosten. De opdrachtgever is ook niet bekend met Java EE en hierdoor viel deze optie dus af. Het scheiden van code naar een client-server applicatie geeft een hoop fragmentatie van de code. Hierdoor blijft er nog 1 optie over die deze nadelen niet heeft en dat is de centrale database. Het probleem dat deze optie niet via 1 centraal punt loopt en er dus eerder fouten in de database ontstaan zal met transacties opgelost kunnen worden. Welke keuze we maken is echter ook afhankelijk van de volgende deelvraag.

7.4 Hoe kun je het beste de data opslaan als er meerdere gebruikers tegelijk in werken?

Om te kijken welke architectuur we kiezen voor de multi-user versie van de applicatie is het ook belangrijk om te kijken naar verschillende manieren om gegevens op te slaan en deze te vergelijken. Momenteel wordt de data geproduceerd door Symbiosis serialized opgeslagen in 1 file. Niet erg geschikt om met meer mensen in te werken. Hieronder zal ik dus een vergelijking maken tussen de verschillende opties die ik heb bekeken. Voor informatie over onderstaande technieken heb ik gebruik gemaakt van de volgende bronnen:

"JSON – JavaScript Object Notation" (JSON)
"Pro JPA 2: Mastering the Java™ Persistence API" (Mike Keith, 2009)
"Java persistence with hibernate" (Christian Bauer, 2006)
"Understanding JPA" (Aditi, 2008)
"JPA Tutorial with Examples using Hibernate in Standalone" (Verstrynge, 2012)

7.4.1 XML / JSON

XML is een mark-up taal origineel ontwikkeld voor het grootschalig aanpakken van elektronisch publiceren wordt tegenwoordig steeds vaker gebruikt voor de communicatie van data over het web. (W3C)

JSON is een lichtgewicht text-based open standaard ontworpen voor leesbare datacommunicatie over het web. JSON maakt gebruik van 2 verschillende primaire datastructuren namelijk een array en een object met name/value waardes. (JSON)

De reden dat ik deze samen bekijk is omdat beiden standaarden een vergelijkbaar doel hebben op dezelfde manier toegepast zouden worden.

Voordelen	Nadelen
Eenvoudig te implementeren met tools als XStream	Geen version control, zodra 2 personen tegelijk een aanpassing maken zal dit conflicten opleveren.
Leesbaar buiten de applicatie om	Aan te passen buiten de applicatie om.
Minder gevoelig voor source changes als serialisatie	
Veelgebruikte web technieken zou goed samenwerken met web-services	

7.4.2 Serialisatie

Momenteel maakt de applicatie al gebruik van serialisatie echter slaat deze alles op in 1 bestand. Het nadeel aan alles in 1 bestand opslaan is dat het zeer moeilijk wordt wijzigingen te delen doordat er zodra 1 iemand iets wijzigt er meteen een conflict is. Het is echter ook mogelijk om per bestand te serialiseren. Hieronder staan de voor en nadelen van deze techniek.

Voordelen	Nadelen
Eenvoudig te implementeren, applicatie is al serialiseerbaar.	Geen version control, zodra 2 personen tegelijk een aanpassing maken zal dit conflicten opleveren.
Niet leesbaar buiten de applicatie.	Zeer afhankelijk van source versie. Updates maken bestanden onbruikbaar.
Niet aan te passen buiten de applicatie.	

7.4.3 Centrale database

Als laatste wil ik nogmaals kijken naar de centrale database maar nu puur voor persistentie. Databases heb je in alle soorten en maten. Ik heb hier enkel gekeken naar de relationele database. De overige databases zijn buiten beschouwing gelaten.

Voordelen	Nadelen
Transactie management (afhankelijk van gekozen database)	Vereist enkele aanpassingen in het model.
Eenvoudig te doorzoeken met HQL.	Database server vereist.
Iedereen werkt met dezelfde server geen gedoe met versie controle.	
Relatief eenvoudige implementatie met JPA	

7.4.4 JPA

Aangezien de centrale database mijn voorkeur heeft indien JPA een optie is zal ik even moeten kijken naar de toepassing van JPA binnen een Java SE omgeving. Aangezien je binnen Java SE geen container hebt die de JPA instance beheert zal je dat binnen Java SE zelf moeten regelen. Binnen JPA is het echter niet mogelijk om alle, door Symbiosis gebruikte, ontwerpkeuzes te persisteren. Zo is het niet mogelijk om inner classes te persisteren en is het tevens niet mogelijk om fields van een interface te persisteren.

Deze 2 problemen zijn beiden aan te pakken. Zo kun je inner classes externalizen en de lifetime dependency anders te regelen. Bij fields die gebruik maken van een interface is het in veel gevallen mogelijk om een abstracte class te introduceren welke wel te mappen zijn.

7.4.4 Conclusies en aanbevelingen

Terugkomend op de conclusie van hoofdstuk 7.3.5 waar er nader onderzoek nodig was hebben we hier de persistentie mogelijkheden bekeken bij multi-user gebruik. Alle bekeken opties zijn geldige manieren de applicatie multi-user te maken. Als je echter geen fragmentatie van de code wilt door het client server te maken en geen eigen versie control systeem wilt schrijven, wat veel te veel tijd in beslag zou nemen, blijft er slechts 1 logische optie over en dat is de centrale relationele database.

Ik heb er dan ook voor gekozen om een Oracle 11g database te gebruiken voor de applicatie. De Oracle 11g database ondersteund transacties en is dus prima geschikt voor dit doel. Een groot nadeel hieraan is dat het model op enkele plaatsen veranderd moet worden aangezien JPA niet met alle constructies om kan gaan zoals een inner class of fields van een Interface. Hier zijn echter workarounds voor beschikbaar.

8 Deelvraag 3: Eclipse integratie

8.1 Onderzoeksvragen

3 – How can we incorporate the Symbiosis application within Eclipse?

3 - Hoe kunnen we de Symbiosis applicatie binnen Eclipse integreren?

Aan deze vraag zitten nog enkele sub vragen vast.

3.1 - Wat is de techniek gebruikt voor Eclipse plug-ins en is deze compatible met de Symbiosis applicatie?

3.2 - *In hoeverre is het mogelijk een Swing GUI binnen SWT te laten draaien?*

8.2 Wat is de techniek gebruikt voor Eclipse plug-ins en is deze compatible met de Symbiosis applicatie?

Om antwoord op deze vraag te vinden ben ik begonnen met wat desk research naar Eclipse aangezien deze IDE nog onbekend was bij mij. Op voorhand had ik enkel ervaring binnen Netbeans IDE. Evenals Netbeans is Eclipse een veelgebruikte IDE voor Java ontwikkeling. Eclipse blijkt zelfs een groter marktaandeel te hebben dan Netbeans. Dit verschil is zelfs zeer significant, 7 van de 10 maakt gebruik van Eclipse of een variant van Eclipse. Netbeans zit slechts op 8% marktaandeel. (Tee, 2012)

De keuze voor welke IDE een plug-in te maken ligt dan ook vrij voor de hand.

8.2.1 Eclipse IDE

Door een plug-in voor Eclipse te maken kun je veel meer ontwikkelaars bereiken. Tijd om dus dieper in Eclipse te duiken. Eclipse is geschreven in Java en de plug-ins dienen ook in Java geschreven te worden. Om contributies te maken aan de GUI van Eclipse ben je echter gebonden aan de SWT UI toolkit. Zoals al eerder besproken in hoofdstuk 6.2 betreft Symbiosis een Swing applicatie en zijn Swing en SWT niet direct met elkaar te gebruiken omdat ze een ander threading model gebruiken. Hier zit dus een issue die nader bekeken moet worden. Dit probleem wordt verder beschreven in hoofdstuk 8.3.

Eclipse heeft voor zijn plug-ins 2 mogelijkheden om views toe te voegen aan zijn user interface namelijk viewparts en editor windows. Beiden werken anders dan een standaard Java frame in Swing of SWT. Deze worden hieronder in detail uitgelegd.

8.2.2 Viewparts

Een viewpart is een veelgebruikt window binnen Eclipse. Een viewpart bestaat uit 2 standaard onderdelen.

Het eerste onderdeel is de SWT composite. Binnen deze SWT composite kun je andere SWT componenten opnemen en is het dus mogelijk je eigen forms te maken.

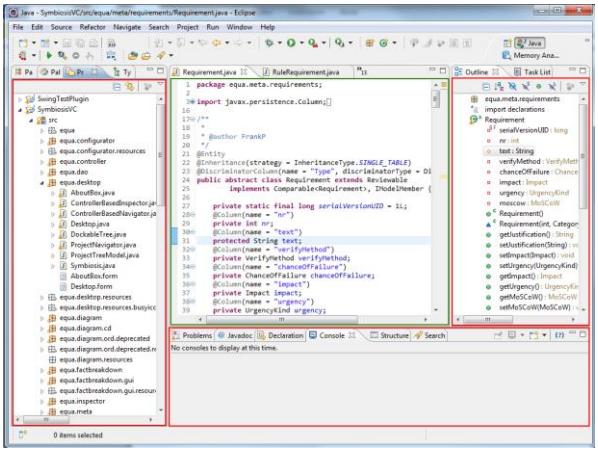
Het tweede onderdeel is de toolbar. De toolbar kun je vullen met Actions. Deze actions kunnen methodes aanroepen binnen de java file.

Viewparts zijn binnen Eclipse dockable op bijna alle plekken behalve de main toolbar en de editor window. Deze zijn ook stackable.

8.2.3 Editor Windows

Een editor window is naast de viewpart de enige manier om contributies aan de Eclipse GUI te maken. Het grote verschil tussen editor windows en viewparts is dat een editor window standaard een source file heeft. Dit kan bijvoorbeeld een XML file zijn maar ook een java file. Binnen de editor window is standaard een tekst editor beschikbaar.

Een editor window heeft verder geen toolbar. Wel is het mogelijk om SWT componenten te gebruiken binnen een editor window. Met deze componenten kun je een form maken waarmee je de source file aan kunnen passen.



[Eclipse IDE: rood geeft viewparts weer, groen het editor window]

8.2.4 Conclusie en aanbevelingen

Eclipse plug-ins maken net als Symbiosis gebruik van Java en daarom kan Symbiosis prima gebruikt worden binnen Eclipse als plug-in. Eclipse maakt echter gebruik van een andere UI toolkit dan dat in Symbiosis gebruikt wordt. Deze zijn niet standaard met elkaar te gebruiken en dit heeft verder onderzoek nodig. Dit wordt bekeken in hoofdstuk 8.3.

8.3 In hoeverre is het mogelijk een Swing GUI binnen SWT te laten draaien?

Door de fundamentele verschillen tussen Swing en SWT is het moeilijk de 2 door elkaar te gebruiken. Echter is er sinds versie 3 van Eclipse een swt-awt bridge geïntroduceerd waardoor het mogelijk is de 2 technieken naast elkaar te gebruiken. Hier zitten echter nog een hoop haken en ogen aan waar je rekening mee moet houden. Eclipse heeft een artikel gepubliceerd waarin de grote valkuilen staan welke ik als basis heb genomen voor het beantwoorden van deze vraag. (Hirsch, 2007) Een ander boek wat ik heb gebruikt om deadlocks te voorkomen is "Java concurrency in practice" (Brian, et al., 2006).

In de basis is het integreren van een AWT/Swing component binnen SWT zeer eenvoudig en met 2 regels code te realiseren. [zie snippet 1]

```
[Snippet 1]

Composite composite = new Composite(parent, SWT.EMBEDDED | SWT.NO_BACKGROUND);
Frame frame = SWT_AWT.new_Frame(composite);
```

Met de code uit snippet 1 krijg je een awt frame binnen een swt composite. De events van zowel de Swing componenten als de Eclipse componenten zullen netjes afgehandeld worden. De problemen ontstaan zodra er vanuit een awt naar een swt component een call wordt gemaakt. Door de verschillende threading modellen kunnen deze deadlocks veroorzaken indien direct aangeroepen. Om dit probleem op te lossen dien je de call in de juiste eventQueue van het targetobject te plaatsen.

Om een call op de AWT event thread te plaatsen gebruik je:

- javax.swing.SwingUtilities.invokeLater()
- javax.swing.SwingUtilities.invokeAndWait()

Om een call op de SWT thread te plaatsen gebruik je:

- org.eclipse.swt.widgets.Display.asyncExec()
- org.eclipse.swt.widgets.Display.syncExec()

Het gebruik van invokeAndWait() in combinatie met syncExec() geeft echter een deadlock. Beiden threads zullen hierdoor in een blocking state komen en hier niet meer uit kunnen komen. Bijvoorbeeld het openen van een nieuw Jdialog binnen een SWT applicatie geeft dit probleem. Om dit probleem op te lossen moet de SWT thread tijdelijk geblokkeerd worden door syncExec() aan te roepen en in de aangeroepen Swing thread na het uitvoeren van zijn call de SWT thread weer vrijgegeven worden. [Snippet 2]

Hiermee los je een hoop van de threading problemen op. Echter zitten er ook nadelen aan en bijvoorbeeld dat alle Swing runnables moeten als final gedeclareerd worden binnen de SWT component class. Een ander nadeel is dat het de performance niet ten goede komt aangezien de threads steeds op elkaar moeten wachten.

Enkele problemen die er niet mee opgelost worden zijn het openen van popup menu's en window Events. Zodra er een Swing popup geopend is en je opent een SWT popup zullen deze beiden zichtbaar zijn. Dit is enkel op te lossen door deze handmatig te sluiten. Hetzelfde geldt voor window events.

```
[Snippet 2]
      EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    // do swing work...
                    runnable.run();
                } finally {
                    display.asyncExec(new Runnable() {
                        public void run() {
                            // Unblock SWT
                            SwtInputBlocker.unblock();
                        }
                    });
                }
            }
        });

        // Prevent user input on SWT components
        SwtInputBlocker.block();
```

8.3.1 Look en feel

Het gebruik van Swing en SWT samen geeft het probleem dat de look en feel van beiden UI toolkits verschillend zijn. SWT zal altijd de look en feel van het host operating system pakken aangezien SWT native library's gebruikt. Swing gebruikt echter een geëmuleerde look en feel die op ieder host operating system hetzelfde eruit ziet.

Om dit probleem op te lossen is het nodig om bij de Swing componenten de look en feel gelijk te zetten aan de look en feel van het host operating system.

Een ander probleem is dat bij het resizen van een embedded frame er background flicker optreed. Dit wordt veroorzaakt doordat het AWT window resize event constant de background probeert te repainten.

Dit is echter enorm te verminderen door [Snippet 3] te gebruiken binnen het SWT component. Dit is echter enkel effectief op het Windows platform. Op de andere platformen is hier helaas geen oplossing voor. Het gebruik van deze property geeft echter wel weer andere problemen. Zodra een window groter wordt gemaakt kan het zijn dat niet alles goed weergegeven wordt. Dit probleem is echter op te lossen door een resize listener aan de SWT composite te hangen en zodra het window groter gemaakt word de background te fillen. Dit is bij Symbiosis echter geen issue.

```
[Snippet 3]

System.setProperty("sun.awt.noerasebackground", "true");
```

De anti aliassing van Swing componenten binnen SWT gaat ook niet standaard goed. Dit is echter op te lossen door [Snippet 4]

```
[Snippet 4]

System.setProperty("swing.aatext", "true");
```

8.3.2 Wrapper classes

Om het geheel wel beheerbaar te houden is het echter niet handig om voor elk Swing component custom code te moeten schrijven om de events af te handelen en door te sturen naar juiste SWT threads. Om deze reden heb ik gebruik gemaakt van wrapper classes die 90% van de standaard events afvangen en de rest met final runnables beschikbaar maken.

Het aanmaken van een Swing window gaat hierdoor anders maar heeft verder geen custom code meer nodig. Hierdoor kan je dezelfde Swing windows gebruiken als de Symbiosis applicatie wat een hoop tijd scheelt. Mochten er later nieuwe windows bijkomen kost dit zeer weinig tijd om deze ook beschikbaar te maken binnen de Eclipse plug-in.

8.4 Hoe kunnen we de Symbiosis applicatie binnen Eclipse integreren?

Met de wrapper classes zijn we bijna klaar om Symbiosis beschikbaar te maken als Eclipse applicatie. Het enige wat nog moet gebeuren is het herschrijven van de applicatie naar het MVC (Model View Controller) patroon. In de standaard Swing applicatie zaten erg veel referenties naar het main window. Deze is niet beschikbaar binnen de Eclipse versie aangezien deze een eigen docking systeem heeft met zijn viewparts. In de originele versie van de applicatie zaten de referenties naar het project en user en het hele model in het main window verwerkt. Vanuit de Eclipse plug-in kan je hier dus niet bij.

Voor informatie betreffende het MVC patroon heb ik de volgende bronnen gebruikt:
http://www.oracle.com/technetwork/articles/javase/index-142890.html (Eckstein, 2007)
Het boek: "Design Patterns: Chapter 1" (Erich, Richard, Ralph, & John, 2009)

8.4.1 Ombouw naar MVC

Het ombouwen van Symbiosis moet gebeuren om de volgende redenen:

- Het MVC patroon zorgt ervoor dat de Views losgekoppeld worden en niet meer direct de data opvragen. Alles loopt via de Controller classes welke alle input afhandelen, de data uit de Model classes ophalen en serveren aan de Views. Dit maakt het geheel een stuk flexibeler en beter onderhoudbaar.
- De huidige versie van de applicatie haalt direct alle data op binnen de views. Aangezien de applicatie tevens tot een Eclipse plug-in omgebouwd (naast de standalone applicatie) is dit geen mogelijkheid. De Eclipse plug-in vereist namelijk andere views dan de standalone applicatie. Zou je dit niet ombouwen zou dit een ramp worden om te onderhouden.

8.4.2 Betreft classes/packages

Aanpassingen nodig in bestaande Packages/Classes:

- Equa.decomposition.gui
  - Een hoop classes uit deze package gebruiken de Desktop class. Deze moeten omgebouwd worden zodat deze de controller classes gebruiken.
- Equa.desktop
  - Desktop.java
    - Beiden versies
      - Een hoop van de code dient verplaatst te worden naar de controller. (e.g. Save, Load en Timer functies)
      - Referenties van het project en overige views dienen verplaatst te worden naar de controller.
    - Eclipse specifiek
      - Docking is overbodig in de Eclipse versie, Eclipse werkt met View-parts die al een vorm van docking gebruiken.
    - Standalone versie
      - Docking code aanpassen om met de controller te werken ipv de desktop class.
  - ProjectNavigator
    - Moet controller gebruiken ipv Desktop
  - ProjectInspector
    - Moet controller gebruiken ipv Desktop
- Equa.diagram.cd
  - ClassDiagramPanel.java
    - Referentie naar equa.desktop.Desktop.java gebruikt voor PropertyEditor. Moet controller gebruiken.
- Equa.diagram.ord
  - ObjectRoleDiagramPanel.java

Referentie naar equa.desktop.Desktop.java gebruikt voor PropertyEditor. Moet controller gebruiken.

8.4.3 Nieuwe packages

Om alle views aan te sturen wil ik een nieuwe package introduceren met daarin de volgende classes:

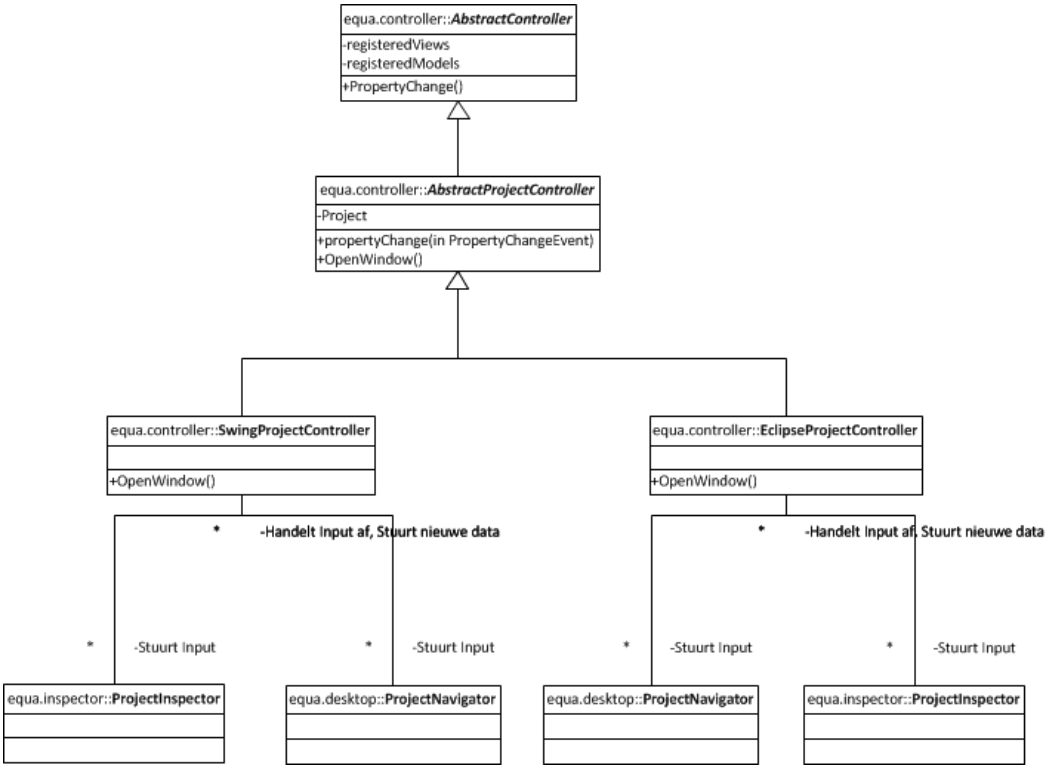
Equa.controller:

- AbstractController.java
  - Deze class zal alle standaard controller class functies bevatten. (Eventlisteners)
- AbstractProjectController.java
  - Deze class zal alle standaard non view related functies bevatten.
- SwingProjectController.java extends AbstractProjectController
  - Deze class zal alle aanroepen naar de standaard Swing views doen en openen.
- EclipseProjectController.java extends AbstractProjectController
  - Deze class zal alle aanroepen naar de gewrapte Swing views doen en openen.

Equa.test:

- In deze package zullen unit tests komen voor alle **toegevoegde** classes die getest moeten worden.

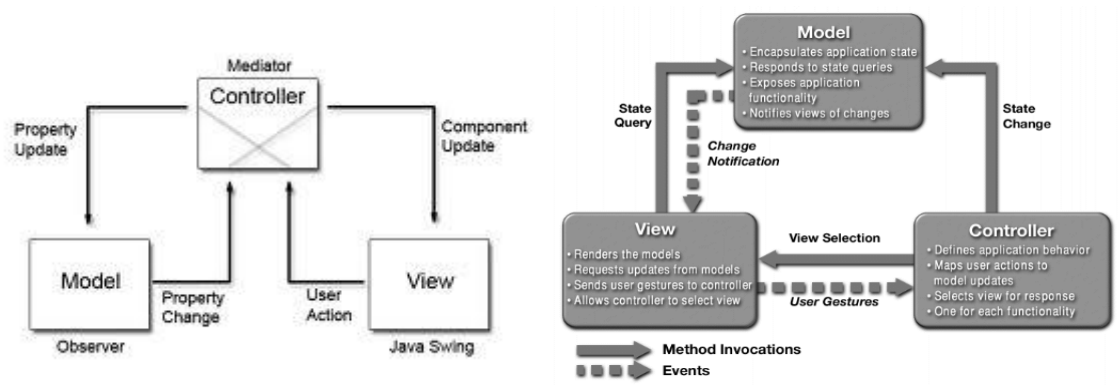
Er is een scheiding gemaakt tussen de Swing applicatie controller en de Eclipse plug-in controller. Dit is gedaan omdat de Eclipse plug-in de views op een andere manier moet openen (wrapper classes) aangezien er anders deadlocks optreden. Om het geheel wel onderhoudbaar te houden zullen alle standaard non view related functies worden opgenomen binnen een abstracte class welke word extended door beiden versies.



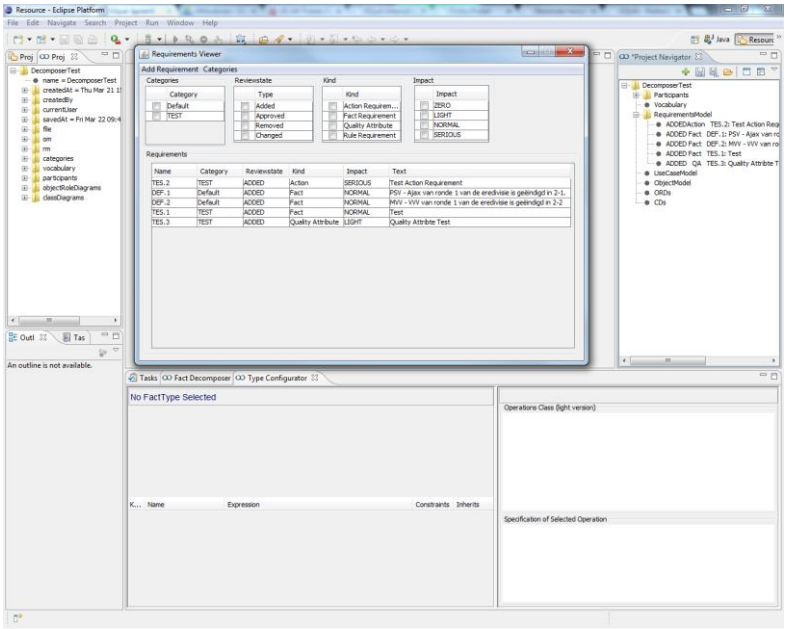
8.4.4 Gekozen vorm MVC

Er zijn meerdere manieren om MVC te implementeren. Zo kan het via het observer patroon waar de views zich abonneren op changes van de model en de input afgehandeld wordt door de controller. Een andere manier is om alles via de controller laten lopen zoals in onderstaand diagram. Beiden technieken hebben zo hun voor- en nadelen.

Aangezien



[2 implementaties van het MVC pattern]



[Voorbeeld Eclipse versie]

9 Hoofdvraag: Multi-user aspecten

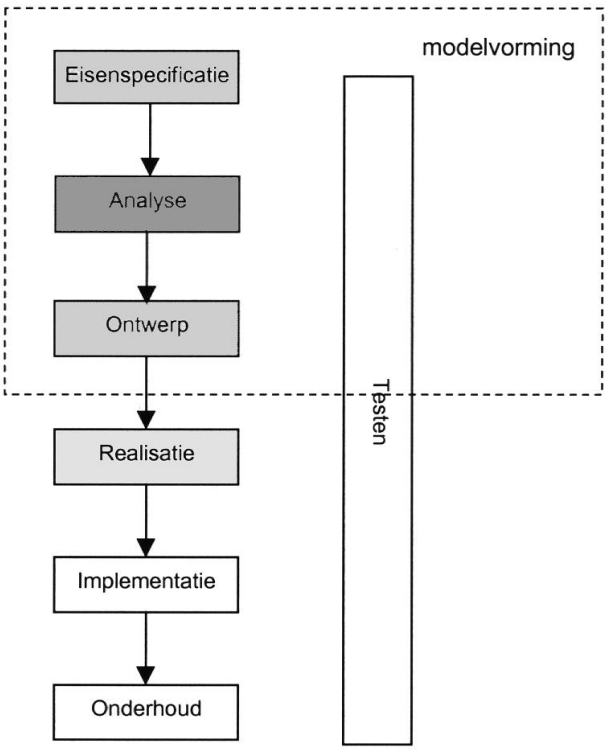
9.1 onderzoeksvragen

H1 - Welke multi-user activiteiten, rekening houdend met de EQuA methodiek, zijn benodigd of kunnen beter vermeden worden tijdens het ontwerpen van het domeinmodel?

9.2 EQuA Methodiek

Nu we antwoord hebben op de deelvragen kunnen we een dieper in de hoofdvraag duiken. Om op deze vraag goed antwoord te kunnen geven zullen we ons eerst moeten verdiepen in de EQuA methodiek en de code van de applicatie. Het boek waar deze methodiek op gebaseerd is heet "Objectgeörienteerde domeinanalyse" en is geschreven door Frens Vonken en Frank Peeters, mijn opdrachtgever (Vonken & Peeters, 2001). In dit boek wordt de EXPO methode beschreven die aan de basis staat van de Symbiosis applicatie.

Welke multi-user activiteiten komen dan aan bod tijdens het ontwerpen van het domeinmodel of beter gezegd tijdens de modelvorming met deze methode? De eerste fase van het modelvormingsproces vindt plaats buiten de Symbiosis applicatie. In deze eerste fase worden de requirements en eisen van de applicatie verzameld en gestructureerd. Hier heeft het dus geen enkele impact en dit kan door meerdere mensen tegelijk afgehandeld worden zonder dat hier conflicten optreden. Nadat alle eisen verzameld zijn zullen deze in de applicatie ingevoerd worden door een projectmember of een stakeholder. Deze kunnen onafhankelijk van elkaar requirements invoeren zonder dat dit impact op elkaar heeft.



Alvorens deze requirements goedgekeurd zijn door een stakeholder zal het echter beperkt blijven tot invoer en zal de volgende fase niet mogelijk moeten zijn. Dit moeten we dus beperken binnen de applicatie. Dit zou namelijk onnodig werk opleveren zodra de requirement afgekeurd of gewijzigd wordt. Nadat de requirements goedgekeurd zijn kan er overgegaan worden naar fase 2.

Tijdens fase 2, de analyse, wordt er na aanleiding van de requirements en eisenspecificatie gekeken uit welke onderdelen het te ontwikkelen systeem moet bestaan. Zodra dit bekend is moet er worden gekeken welke objecten een rol spelen om aan de gesteld eisen te voldoen. Ook is het belangrijk om te weten wat de eigenschappen van deze objecten zijn en waar deze een rol vervullen.

Met deze gegevens is het mogelijk om de objecten vorm te geven en fase 3, de ontwerpfase, in te gaan. Voordat dit in de applicatie kan gebeuren zal er een goedgekeurde fact requirement moeten zijn waar een breakdown op uitgevoerd zal moeten worden. Aangezien een fact breakdown, waarmee een nieuw FactType en eventueel ObjectTypes geïntroduceerd worden, een grote impact op het model heeft zal dit maar door 1 persoon tegelijk mogen gebeuren. Als meerdere projectmembers hier

tegelijk een breakdown op uit zouden voeren kunnen er dubbele FactTypes en Objecten aangemaakt worden. Dit zal dus voorkomen moeten worden. Dit kan op meerdere manieren.

Het is bijvoorbeeld mogelijk om een requirement te locken op applicatie niveau zodra een breakdown gestart wordt door een van de projectmembers. Dit heeft het voordeel dat deze requirement niet veranderd kan worden door iemand anders in de tussentijd en de breakdown dus niet dubbel uitgevoerd kan worden. Een nadeel hieraan is wel dat als iemand dit per ongeluk open laat staan de requirement een hele tijd op locked kan komen te staan zonder dat er iets mee gebeurd.

Een andere manier om dit aan te pakken is om projectmembers verantwoordelijk te maken voor een categorie net zoals dat mogelijk is voor stakeholders. Een stakeholder die verantwoordelijk is voor een categorie heeft de mogelijkheid om binnen die categorie requirements goed te keuren. Als je ook een projectmember aan een categorie toe kan wijzen dan is deze verantwoordelijk voor de breakdown van deze requirements. Het voordeel hieraan is dat je weet wie verantwoordelijk is voor de breakdown en zodra er iets misgaat wie de fout gemaakt heeft. Een nadeel aan deze opzet is natuurlijk dat de verantwoordelijk niet aanwezig is en het stil blijft liggen tot hij/zij weer terug is. Om dit nadeel aan te pakken zou je, net als bij de stakeholder een 2^{e} projectmember kunnen aanstellen als back-up. Op deze manier zal het veel minder vaak voorkomen dat er iets blijft hangen. Het is dan echter aan de gebruikers dat ze niet tegelijk in de requirements van eenzelfde categorie gaan werken om conflicten te voorkomen.

Na de fact requirement breakdown ontstaan er allemaal FactTypes en ObjectTypes. Deze moeten nog verder geconfigureerd worden. Het configureren van FactTypes en ObjectTypes heeft impact op het gedrag van de uiteindelijke objecten. Deze veranderingen zijn echter zeer klein en hebben pas daadwerkelijk een impact zodra het gedrag gegenereerd wordt. Hier kunnen dus prima meerdere mensen tegelijk in werken zonder dat er issues ontstaan. Hier hoeft verder ook niets beperkt of uitgebreid te worden binnen de applicatie.

Op dit punt is de ontwerpfase van het domeinmodel klaar. Natuurlijk kan het nog verder verfijnd worden en zijn de fases opnieuw door te lopen zodra er veranderingen zijn in de requirements. Met de nieuwe changes in de applicatie is het ook mogelijk om deze fases door elkaar te doorlopen. Terwijl bijvoorbeeld het model geanalyseerd wordt is het nog steeds mogelijk nieuwe requirements in te voeren.

9.3 Multi-user Activiteiten

Om te weten waar er aandacht benodigd is bij het multi-user maken van de applicatie zullen we moeten weten wat er voor een gebruiker allemaal mogelijk is in de applicatie en of dit enige impact op het model heeft. Om hier achter te komen zullen we de informatie die we bij deelvraag 1.2: De aanpassingen op de User Interface hebben opgedaan, de bestaande views van de applicatie en de extra requirements terug te vinden in het requirements document uit bijlage B.

9.3.1 Activiteiten

Hier onder staat een overzicht van alle mogelijke activiteiten binnen Symbiosis. Voor elke activiteit zal bekeken moeten worden of hier een aanpassing in moet komen zodra deze multi-user gemaakt moet worden.

Project activiteiten:

- New Project
- Open Project
- Save Project
- SaveAs Project

Symbiosis was origineel gebouwd om 1 project heen gebouwd zowel in het model als in de views was geen rekening gehouden met de mogelijkheid tot meerdere projecten. Hier moesten dus aanpassingen in gemaakt worden.

Bij het maken van een nieuw project zal er weinig verschil zijn. De gebruiker zal nog steeds de wizard moeten doorlopen echter zal deze zodra de wizard afgerond is ook het nieuwe project naar de database weg moeten schrijven.

Het openen van het project ging voorheen door het openen van een geserialiseerd bestand. Nu de applicatie dus multi-user moet worden moet dit aangepast worden en zullen de projecten moeten worden opgehaald uit de database en in een nieuwe view gezet worden waar de gebruiker zijn project kan selecteren.

Het opslaan van een project wordt in de nieuwe situatie overbodig. Het project wordt direct in de database opgeslagen en veranderingen hierop zullen ook direct naar de database geschreven moeten worden. Een losse save functie is dus overbodig zodra je multi-user gaat werken echter zal deze er tijdens de ontwikkelperiode in blijven zitten.

Categorie activiteiten:

- Categorie: toevoegen/verwijderen
- Categorie: wijzigen
    - Naam wijzigen
    - Owner wijzigen

Symbiosis maakte al gebruik van categorieën echter waren deze statisch en hier was nog geen beheer over. Om dit aan te pakken zijn er enkele views bijgekomen die deze wijzigingen mogelijk maken. Wijzigingen, toevoegingen en verwijderingen van categorieën worden uitgevoerd en zullen weggeschreven worden naar de database.

Participant activiteiten:

- Participant: toevoegen/verwijderen
- Participant: wijzigen
  - Projectrole wijzigen
  - Naam wijzigen
- Participant: login

Participants bestonden al binnen Symbiosis echter waren deze opgenomen binnen het project, kon een gebruiker niet bij meerdere projecten betrokken zijn en was er geen beveiliging. Aangezien het wenselijk is als een participant bij meerdere projecten betrokken kan zijn zullen de participants buiten het project gehaald moeten worden en zal er moeten worden ingelogd voordat de gebruiker de projecten waarvan hij deel uitmaakt te zien krijgt.
Wijzigingen, toevoegingen en verwijderingen van participants worden uitgevoerd en zullen weggeschreven worden naar de database.

Requirement activiteiten:

- Requirement: toevoegen/verwijderen
- Requirement: wijzigen
  - Requirement: kind wijziging
  - Requirement: tekst wijziging
  - Requirement: categorie wijziging
  - Requirement: attributen wijziging
- Requirement: breakdown
- Requirement: validatie
  - Approve requirement
  - Reject requirement
  - Rollback requirement
  - Remove requirement

Om het toevoegen, aanpassen, verwijderen en valideren van requirements geschikt te maken voor gebruik door meer gebruikers zal de invoer gescheiden moeten worden van de breakdown. Als dit niet gebeurd zal de gehele validatie stap niet uitgevoerd kunnen worden. De invoer en breakdown zullen dan ook gescheiden moeten worden.

Het toevoegen en verwijderen zal direct doorgevoerd moeten worden en weggeschreven worden naar de database zodat iedereen een up-to-date requirementsmodel heeft. Bij het wijzigen van een requirement is het afhankelijk van de wijziging wat er moet gebeuren. Zodra de kind van een requirement wordt veranderd (bijvoorbeeld van action naar fact) zal de originele requirement verwijderd worden en een nieuwe requirement aangemaakt worden in het model. Door het verwijderen van de originele requirement raak je wel de geschiedenis van die requirement kwijt. Dit zal niet het geval zijn zodra je de tekst, categorie of andere attributen gaat wijzigen. Deze wijzigingen zullen moeten worden doorgevoerd in het model en naar de database weggeschreven moeten worden.

Tijdens de breakdown van een requirement zullen allemaal objecten ontstaan. Deze objecten zijn tijdelijk voor de vorming van de fact en object types en worden na de breakdown niet meer gebruikt. Hier zullen dus geen items buiten de fact en object types naar de database weggeschreven hoeven worden.

Her valideren van een requirement op wat voor manier dan ook zal moeten worden weggeschreven naar de database. Het valideren van een requirement kan echter leiden tot de verwijdering van de requirement. Als deze requirement al gerealiseerd was zal de gebruiker hiervan op de hoogte moeten worden gebracht en indien deze doorgaat met de verwijdering zullen alle fact en object types ontstaan uit deze requirement onbetrouwbaar moeten worden gemarkeerd.

Objectmodel activiteiten:
Abstract object type: toevoegen/verwijderen
Inheritance: toevoegen/verwijderen
Behaviour: toevoeren/verwijderen (met en zonder registries)

Deze acties waren allemaal al aanwezig binnen Symbiosis. Het toevoegen en verwijderen van een abstract object type en inheritance zullen moeten worden weggeschreven naar de database. Behaviour zal niet worden opgeslagen omdat dit afleidbaar is. Deze moet echter wel worden gegenereerd. Zodra deze niet meer klopt met het model zal deze onbetrouwbaar gemarkeerd worden.

Diagram activiteiten:
Class diagram: toevoegen/verwijderen
Class diagram: wijzigen

De mogelijkheid om een class diagram toe te voegen, wijzigen en verwijderen was al aanwezig. Aangezien behaviour echter niet opgeslagen wordt omdat dit afleidbaar is en wel gebruikt wordt binnen een class diagram zal dit voordat de class diagrammen geopend worden het gedrag gegenereerd moeten worden. Het wegschrijven naar de database zal echter minder gemakkelijk gaan aangezien er veel gebruik gemaakt wordt van een externe library voor de weergave. De opdrachtgever neemt echter genoegen met de locatie van de class boxes.

Type configuration:
Fact Type: set (derivable/immutable/value type/comparable)
Fact Type: typeExpression wijzigen
Fact Type: naam wijzigen
Fact Type: verwijderen
objectify/deobjectify: Fact type
Fact Type: constraint base type wijzigen
Fact Type: inheritance toevoegen/verwijderen

Role: uniqueness constraint toevoegen/verwijderen
Role: mandatory constraint toevoegen/verwijderen
Role: overige constraints toevoegen/verwijderen
Role: naam wijzigen
Role: overige wijzigingen

Als laatste werpen we een blik op de type configuration. De activiteiten die hier uitgevoerd worden zijn allemaal wijzigingen op de fact en object types. De multi-user afhandeling is hier ook niet verschillend en zal na een wijziging, toevoeging en verwijdering weggeschreven moeten worden naar de database.

9.4 Conclusies en aanbevelingen

Nu we goed hebben gekeken naar de EQuA methodiek, de bestaande applicatie en de gewenste toevoegingen voor de applicatie kunnen we een goede bepaling maken welke delen van de applicatie aangepast moeten worden om goed multi-user te kunnen werken.

Enkele functionaliteiten zoals het opslaan van een project worden overbodig bij het multi-user maken van Symbiosis. De wijzigingen in het model zullen na een wijziging direct naar de database gepusht worden. Hiervoor zijn enkele views nodig.

Een specifiek geval is het class diagram. Deze is afhankelijk van afgeleide data die niet opgeslagen wordt. Hier zal dus eenmalig een extra stap uitgevoerd moeten worden zodra de applicatie opgestart wordt namelijk het genereren van behaviour.

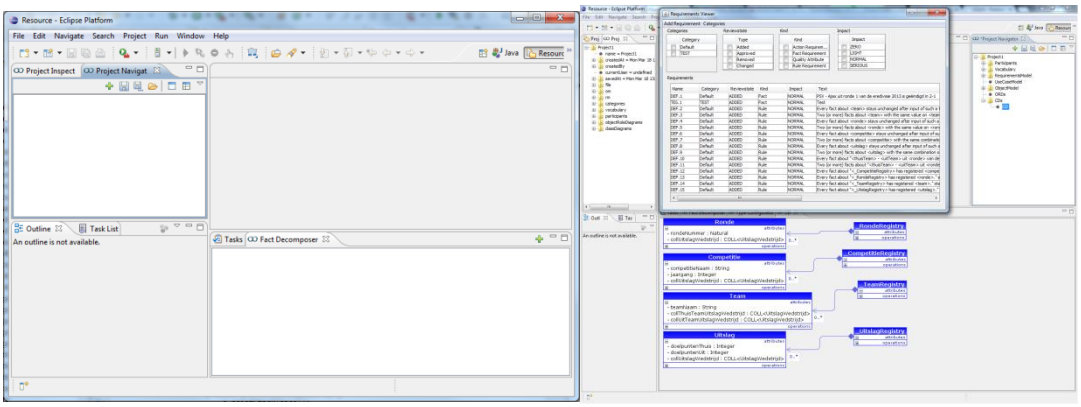
Een ander belangrijk punt is dat de participants buiten de projecten (vanuit de code gezien) gehaald worden waardoor ze deel kunnen nemen aan verschillende projecten. Graag zou ik hier ook aanstippen dat zodra er met meerdere mensen in een project gewerkt wordt beveiliging wel een belangrijker punt wordt. Mijn aanbeveling hierbij is om een gebruiker login minimaal met een password te beveiligen in plaats van direct toegang te geven.

10 Resultaten

Na aanleiding van dit onderzoek zijn er ook een aantal producten opgeleverd. Deze resultaten komen in dit hoofdstuk aan bod.

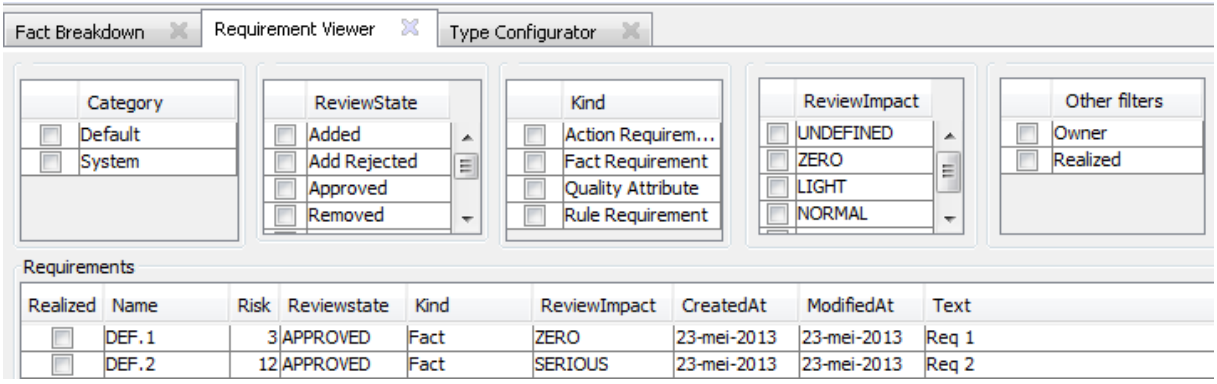
10.1 Eclipse plug-in

De Eclipse plug-in heeft alle functionaliteit van de standalone applicatie maar is binnen de Eclipse omgeving te gebruiken. Hierdoor kunnen ontwikkelaars gebruik maken van de Symbiosis applicatie zonder hun ontwikkel omgeving te verlaten. Voordat deze gemaakt kon worden moest de applicatie echter wel omgebouwd worden naar het MVC patroon aangezien enkele afhankelijkheden niet beschikbaar waren.



10.2 Requirement editor & validator

Er moest een hoop extra functionaliteit opgeleverd worden die nog niet in de applicatie zat. Deze zijn uiteindelijk verwerkt in de requirement editor & validator.



10.3 Multi-user versie Symbiosis

De multi-user versie van Symbiosis ziet er in de meeste opzichten identiek uit aan de standalone versie. Een kleine aanpassing die gedaan is binnen de multi-user versie is het inladen van de projecten. Aangezien deze nu niet meer via een bestand ingeladen worden maar via de database moest hier een extra view voor komen. Verder is de werking identiek aan de standalone versie echter slaat de applicatie wijzigingen op in de centrale database.

11 Evaluatie

11.1 Zelfreflectie

Terugkijkende op afgelopen half jaar kan ik niet anders dan tevreden zijn met hoe alles verlopen is. Zo begon ik alleen aan een duo-stage opdracht in de veronderstelling dat een master student van de TU/e binnen een maand ook aan dezelfde opdracht zou starten. Toen deze student vertraagd was en ik in die maand al ver voor op schema lag hebben we in overleg met de opdrachtgever besloten dat ik de stage alleen zou uitvoeren. Door de stage alleen uit te voeren nam ik natuurlijk wel het risico dat het niet op tijd af zou zijn. Dit is een punt waar ik in het vervolg op moet blijven letten. In het verleden heb ik al vaker te veel hooi op mijn vork genomen waardoor de werkdruk erg hoog lag. Gelukkig heb ik mij tijdens de stageperiode volledig kunnen focussen op mijn stage opdracht en had ik geen overige verantwoordelijkheden. Dit zorgde ervoor dat de projectdoelen alsnog kon bereiken.

Procesmatig is het gehele project zeer goed verlopen. Alle uren werden netjes bijgehouden via een scrum tool en er is een dagelijks verslag gemaakt op mijn blog. Hierdoor konden mijn begeleiders goed in de gaten houden waar ik mee bezig was en of ik nog wel op schema liep. De wekelijkse meetings met de opdrachtgever en bedrijfsbegeleider waren ook van grote waarde en mocht ik mijn stage overdoen zou ik het weer zo aanpakken. Over het procesverloop ben ik dan ook zeer tevreden.

Een punt wat ik in het vervolg beter aan zal pakken is vaker feedback vragen. In het begin van de stage deed ik dit vaker dan aan het eind. Ook wil ik eerder beginnen aan de scriptie zodat ik hier net wat meer aandacht aan kan besteden.

Over de resultaten kan ik ook niet anders dan tevreden zijn. Zo heb ik veel producten op kunnen leveren. Al met al ben ik zeer tevreden over mijn prestaties. Bij ISAAC waren ze dit gelukkig ook en na mijn stage zal ik daar dan ook werkzaam blijven.

11.2 Persoonlijk ontwikkelingsplan: terugblik

Voordat ik daadwerkelijk op stage ging had ik bekeken welke vaardigheden ik graag zou willen ontwikkelen en deze in een persoonlijk ontwikkelingsplan opgenomen. De vaardigheden die ik wilde ontwikkelen waren:

- Onderzoek vaardigheid
- Professioneel handelen
- Technische kennis uitbreiden

Om mijn onderzoek vaardigheid te verbeteren wilde ik een compleet onderzoek zelf uitvoeren binnen mijn stageperiode. Naar mijn mening ben ik hier zeker in geslaagd. Zo heb ik een hoop desk research uitgevoerd, met name literatuur onderzoek, en een kwantitatief onderzoek gehouden betreffende de bruikbaarheid van de applicatie.

Om de vaardigheid professioneel handelen te verbeteren wilde ik mij gedurende de gehele stage periode aan alle afspraken houden en elke dag op tijd komen. Helaas ben ik 1 maal te laat gekomen maar dit was overmacht. Om dit verder te voorkomen heb ik vanaf dat moment gezorgd dat ik minimaal een half uur voor mijn starttijd aan het werk was.

Als laatste wilde ik mijn technische kennis uitbreiden tijdens mijn stage periode. Gelukkig was dit geen probleem. Binnen de opdracht lagen voldoende technische uitdagingen zoals de Eclipse plug-in, verschillende GUI technieken en JPA binnen een Java SE applicatie waar ik nog geen ervaring mee had.

13 Literatuurlijst

Aditi, D. (2008, Januari 17). *Understanding JPA, Part 1: The object-oriented paradigm of data persistence*. Retrieved from Javaworld: http://www.javaworld.com/javaworld/jw-01-2008/jw-01-jpa1.html?page=3

authors, M. (2013, 3 28). *Java GUI frameworks. What to choose?* Retrieved from Stackoverflow: http://stackoverflow.com/questions/7358775/java-gui-frameworks-what-to-choose-swing-swt-awt-swingx-jgoodies-javafx

Brian, G., Tim, P., Joshue, B., Joseph, B., David, H., & Doug, L. (2006). *Java concurency in practice.* Stoughton: Addison Wesley.

Brunekreef, J. (2013, Mei). *EQuA website*. Retrieved from EQuA website: http://www.equaproject.nl/

Chin, S. (2012). *Pro JavaFX 2: A Definitive Guide to Rich Clients with Java Technology.* New York: Apress.

Christian Bauer, G. K. (2006). *Java Persistence with Hibernate.* Greenwich: Manning.

Eckstein, R. (2007, 03). *Java SE Application Design with MVC*. Retrieved from Java SE Application Design with MVC: http://www.oracle.com/technetwork/articles/javase/index-142890.html

Eclipse. (n.d.). *SWT: The Standard Widget Toolkit*. Retrieved from Eclipse.org: http://www.eclipse.org/swt/

Elliott, J. (2002). *Java Swing, 2nd Edition.* Sebastopol: O'Reilly & Associates.

Erich, G., Richard, H., Ralph, J., & John, V. (2009). *Design Patterns.* Westford: Addison-Wesley.

Fowler, A. (n.d.). *A Swing Architecture Overview*. Retrieved from Oracle: http://www.oracle.com/technetwork/java/architecture-142923.html

Harris, R. (2004). *The definitive guide to SWT and JFace.* New York: Springer-Verlag.

Hirsch, G. (2007). *Swing/SWT Integration*. Retrieved from Eclipse: http://www.eclipse.org/articles/article.php?file=Article-Swing-SWT-Integration/index.html

JSON. (n.d.). Retrieved from JSON - Javascript Object Notation: http://www.json.com/

Lam, F., & Robertson, J. (2008, November 13). *Create stand-alone Web services applications.* Retrieved from IBM: https://www6.software.ibm.com/developerworks/education/ws-eclipse-javase1/ws-eclipse-javase1-pdf.pdf

Leuesen, S. (2005). *User interface design.* Essex, England: Pearson Education Limited.

Mike Keith, M. S. (2009). *Pro JPA 2: Mastering the Java™ Persistence API.* New York: Apress.

Oracle BV. (n.d.). *Overview (JavaFX 2.2)*. Retrieved from Oracle: http://docs.oracle.com/javafx/2/api/

Oracle. (n.d.). *JavaFX - The Rich Client Platform*. Retrieved from Oracle: http://www.oracle.com/technetwork/java/javafx/overview/index.html

Tee, J. (2012, Juli 12). *What's the big IDE? Comparing Eclipse and NetBeans*. Retrieved from TheServerSide: http://www.theserverside.com/feature/Whats-the-Big-IDE-Comparing-Eclipse-vs-NetBeans

Verstrynge, J. (2012, Augustus). *JPA Tutorial with Examples using Hibernate in Standalone*. Retrieved from Technical Notes: http://technotes.tostaky.biz/2012/08/jpa-tutorial-with-examples-using-hibernate-standalone.html

Vogel, L. (2012, 8 22). *SWT Tutorial*. Retrieved from Vogella: http://www.vogella.com/articles/SWT/article.html

Vonken, F., & Peeters, F. (2001). *Objectgeörienteerde domeinanalyse.* Schoonhoven: Academic Service.

W3C. (n.d.). *Extensible Markup Language (XML)*. Retrieved from http://www.w3.org/XML/

14 Bijlagen

Bijlage A: Project Initiatie Document

Bijlage B: Requirements document: ReqUI

Bijlage C: Usability test

BIJLAGEN

Document ID 01

Bijlage A: Project Initiatie Document

Document historie

Versie	Status	Datum	Wijzigingen
0.1.0	concept	2013-02-12	Eerste opzet
0.1.1	concept	2013-02-15	Grote wijzigingen en toevoegingen
0.1.2	concept	2013-02-18	Enkele toevoegingen
0.1.3	concept	2013-02-20	Stijl veranderd in ISAAC stijl. WBS, planning en kosten-baten analyse toegevoegd.
0.1.4	concept	2013-02-21	Kleine verbeteringen + bronnen en links toegevoegd.
0.1.5	concept	2013-02-21	Aanpassingen na review Ronald: Sprints verlengd naar 2 weken, typo's verbeterd en links gelegd tussen informatie.
0.1.6	Concept	2013-03-07	Aanpassingen na feedback Coen.

Begrippenlijst

Begrip	Type	Beschrijving
Agile	Projectmanagement	Agile is een iteratieve vorm van projectmanagement.
Bedrijfsbegeleider	Begeleider	Begeleider vanuit ISAAC. Overziet het proces en helpt met de dagelijkse problemen.
Docentbegeleider	Begeleider	Begeleider vanuit de Fontys. Overziet het proces.
Eclipse	IDE	Ontwikkel omgeving.
EQuA	Projectnaam	Staat voor Early Quality Assurance in Software Production.
IDE	Ontwikkelomgeving	Een tool waarmee programma's ontwikkeld kunnen worden.
JBoss	Applicatieserver	JBoss is een applicatie server gericht op het draaien van Java EE applicaties.
MVC	Design pattern	Model View Controller patroon. Een patroon om de logica van de views te scheiden.
Plug-in	Software	Een tool die binnen een andere applicatie kan draaien.
RAAK-PRO	Organisatie	RAAK-PRO richt zich op het versterken van het praktijkgericht onderzoek aan hogescholen, in samenwerking met de beroepspraktijk, en op het intensiveren van de relaties met andere kennisinstellingen. (meer informatie)
Scrum	Projectmanagement	Scrum is een implementatie van Agile. In dit geval wordt Acunote gebruikt.
Symbiosis	Tool	Naam van de tool in ontwikkeling bij Frank Peeters.
UML	Methodiek	Staat voor Unified Modeling Language, gebruikt voor het maken van ontwerp diagrammen.
Web service	Service	Een aan te spreken service om data op te halen of weg te schrijven via XML.
XML	File type	XML is een file type die veel gebruikt word bij web services.

Managementsamenvatting

Doel van dit document

Dit projectinitiatiedocument (PID) heeft als doel de globale eisen en afspraken en overeenkomsten van de opdrachtnemer en opdrachtgever te documenteren. De opdrachtnemer neemt dit document als basis voor al zijn werkzaamheden. De verschillende hoofdstukken geven stap voor stap aan hoe het gerealiseerd gaat worden. Vooral de planning is belangrijk om in de gaten te houden en regelmatig te wijzigen om aan de (nieuwe) eisen te voldoen.

Aanleiding

Er is behoefte vanuit het EQuA project om de bestaande applicatie multi-user te maken en beschikbaar te maken als Eclipse plug-in. Tevens moet de requirement editor and –validator gebruiksvriendelijker gemaakt worden. Hiervoor is het verzoek bij ISAAC Software Solutions binnen gekomen en zal binnen dit project opgepakt worden.

Producten

Uit dit project komen de volgende producten voort:

- Onderzoeksrapport (met onder andere):
  - Mogelijkheden onderzoeken om geen gebruik meer te maken van Swing en over te stappen naar modernere GUI techniek.
  - Onderzoek MVC maken applicatie.
  - Onderzoek client-server maken applicatie.
- Symbiosis (huidige tool in ontwikkeling bij Ir. Frank Peeters) integreren als plug-in in Eclipse en omschrijven tot client.
- Symbiosis server
- Symbiosis client
- Documentatie
- Gebruiksvriendelijke requirements model editor en –validator (binnen de tool)

Globale aanpak

Scrum

In dit project wordt er met sprints van 10 werkdagen (2 weken) gewerkt. De tool die hiervoor gebruikt wordt is Acunote (equa.acunote.com).

Globale kosten en doorlooptijd

Beschikbaar budget	Beschikbare resources
680 uur (85 werkdagen, 1 persoon)	Werktijd van projectleden (projectleden staan beschreven in Hoofdstuk 2)

Risico's

Omdat dit project afhankelijk is van aangeleverde software en meerdere andere partijen, zijn er enkele risico's waar weinig preventief tegen gedaan kan worden. Hierdoor zou ik het project een gemiddelde risicokans geven en een gemiddelde risico impact. Een gedetailleerde uitwerking van de risico's kunt u terugvinden in het document.

Inhoudsopgave

1 Projectdefinitie

Het EQuA project is een RAAK-PRO project, gesubsidieerd door het Ministerie van Onderwijs, Cultuur en Wetenschap. Het project heeft een looptijd van 4 jaar: van november 2010 tot november 2014. Doel van het project is het ontwikkelen van methoden en technieken voor het vroegtijdig opsporen en verbeteren van fouten bij het ontwikkelen van software.

Dit is een sub-project wat binnen het EQuA project valt waarin voornamelijk wordt gekeken naar de multi-user aspecten van Symbiosis, de requirements editor en de bruikbaarheid binnen een IDE.

1.1 Huidige situatie

De Symbiosis tool die momenteel in ontwikkeling is maakt gebruik van de EQuA ontwerpmethodiek. Deze tool is gericht op het gebruik van een enkel persoon. Symbiosis is een losse tool die buiten een ontwikkelomgeving gebruikt wordt. Deze tool is een standaard Java SE applicatie, ontworpen om op 1 systeem te draaien.

1.2 Probleem

De constructie van een requirements model en objectmodel wordt normaliter niet door één en dezelfde persoon uitgevoerd. Binnen een ontwerpproject zijn requirements engineers, software engineers, project management en niet te vergeten stakeholders betrokken. Soms is het praktisch dat sommige betrokkenen tegelijkertijd onderdelen uit het ontwerpproject, conform de EQuA ontwerpmethodiek, kunnen bewerken. Bewerken moet hierbij worden opgevat als feitdecompositie, typeconfiguratie, invoer, wijziging en validatie van requirements en model checking.

1.3 Gewenste situatie

Het moet mogelijk zijn om met meerdere mensen en locaties tegelijk binnen de tool te werken zonder dat dit conflicten oplevert.

De Symbiosis tool moet tevens worden verwerkt binnen een Eclipse plug-in zodat deze gemakkelijker door ontwikkelaars te gebruiken is.

Als laatste zal de tool een eenvoudig te gebruiken requirements editor hebben.

1.4 Projectdoelstellingen

Binnen 20 weken moet de huidige versie van de tool compleet verwerkt zijn als Eclipse plug-in. Tevens moet de tool multi-user gemaakt worden en er dus meerdere gebruikers tegelijk in de applicatie kunnen werken. Als laatste zal de tool een eenvoudig te gebruiken requirements editor hebben. Aan het einde van week 17 zullen de resultaten van dit project worden gepresenteerd. De laatste 3 weken zullen voor nazorg gereserveerd worden.

1.5 Kwaliteit waarborging

Om de kwaliteit te kunnen waarborgen zal er binnen het project gebruikt worden gemaakt van de procedures en richtlijnen van ISAAC Software Solutions BV. Tevens is er erg veel kennis en ervaring binnen ISAAC en er al veel ervaring is met de gebruikte technieken.

1.6 Gekozen oplossing of aanpak

Het project zal Agile worden ontwikkeld in de Scrum vorm. Dit om snel te kunnen aanpassen aan de wensen van de opdrachtgever. Binnen het project zal met sprints van 10 werkdagen (2 weken) worden gewerkt. Hierdoor kan de opdrachtnemer de kwaliteit van de software en de documenten waarborgen.

1.7 Scope van het project

De Eclipse plug-in zal ontworpen, ontwikkeld en beschikbaar worden gesteld. Hiervoor zal ook een nieuwe requirements model editor en -validator worden ontwikkeld die aan de kwaliteitseisen voldoet (deze staan beschreven in het ReqUI document, zie bronnen). Tevens zal zowel de Eclipse plug-in, als de standalone applicatie omgebouwd worden tot een client/server setup zodat meerdere gebruikers tegelijk in de applicatie kunnen werken. Voor alle bovenstaande systemen zal technische documentatie worden geleverd en tevens een handleiding van de nieuwe systemen. Er wordt geen nazorg geleverd buiten dit project.

1.8 Producten c.q. eindresultaat

Hier onder een overzicht van de hoofdpunten. Voor een gedetailleerde lijst van producten verwijs ik u door naar de **W**ork **B**reakdown **S**tructure (WBS).

- Onderzoeksrapport (met onder andere):
    - Mogelijkheden onderzoeken om geen gebruik meer te maken van Swing en over te stappen naar modernere GUI techniek.
    - Onderzoek MVC maken applicatie.
    - Onderzoek client-server maken applicatie.
- Symbiosis (huidige tool in ontwikkeling bij Ir. Frank Peeters) integreren als plug-in in Eclipse en omschrijven tot client.
- Symbiosis server
- Symbiosis client
- Documentatie
- Gebruiksvriendelijke requirements model editor en –validator (binnen de tool)

1.9 Uitsluitingen

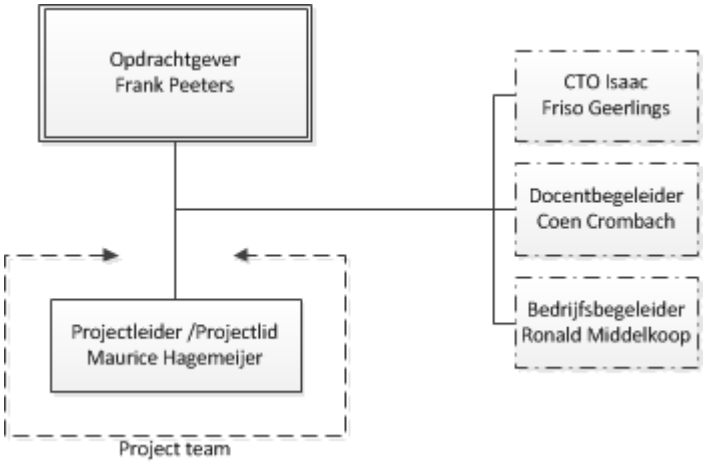
Het project zal de volgende punten **niet** omvatten:

- Er zal geen hosting worden geleverd voor de web-interface
- Er zal geen handleiding en documentatie worden geschreven voor al bestaande onderdelen van de applicatie, enkel de nieuwe onderdelen.

1.10 Randvoorwaarden

- In overleg met de opdrachtgever wordt er een deadline voor het geven van informatie afgesproken. Dit om het project binnen onze planning af te krijgen.
- Er zal elke week een voortgangsvergadering plaatsvinden tussen de projectleden en de bedrijfsbegeleider.
- Er zal elke week een voorgangsgesprek zijn met de opdrachtgever.
- Elk project lid zal dagelijks (werkdagen) zijn email controleren.
- Van de opdrachtgever wordt verwacht dat hij binnen 2 werkdagen een reactie zal sturen op de mails van de projectleider.

2 Project organisatie structuur



2.1 Opdrachtgever

De opdrachtgever is verantwoordelijk voor het verstrekken van informatie. De opdrachtgever controleert ook of het informatiesysteem aan de eisen voldoet.

2.2 Projectsupport

De projectsupport is een expert op een bepaald vakgebied die geraadpleegd kan worden voor ondersteuning met het project.

2.3 Projectleider

De projectleider is de trekker van het project, heeft overzicht, verdeelt taken, stemt deze op elkaar af, stelt actiepunten vast, controleert de voortgang en stelt indien nodig, in overleg met de opdrachtgever, de planning bij. De projectleider moet ook het overzicht van het project behouden en ingrijpen indien nodig.

2.4 Projectlid

Het project lid is de basis van het project: hij werkt mee aan alle fases van het project, waaronder de planning, ontwerpen en uitvoering.

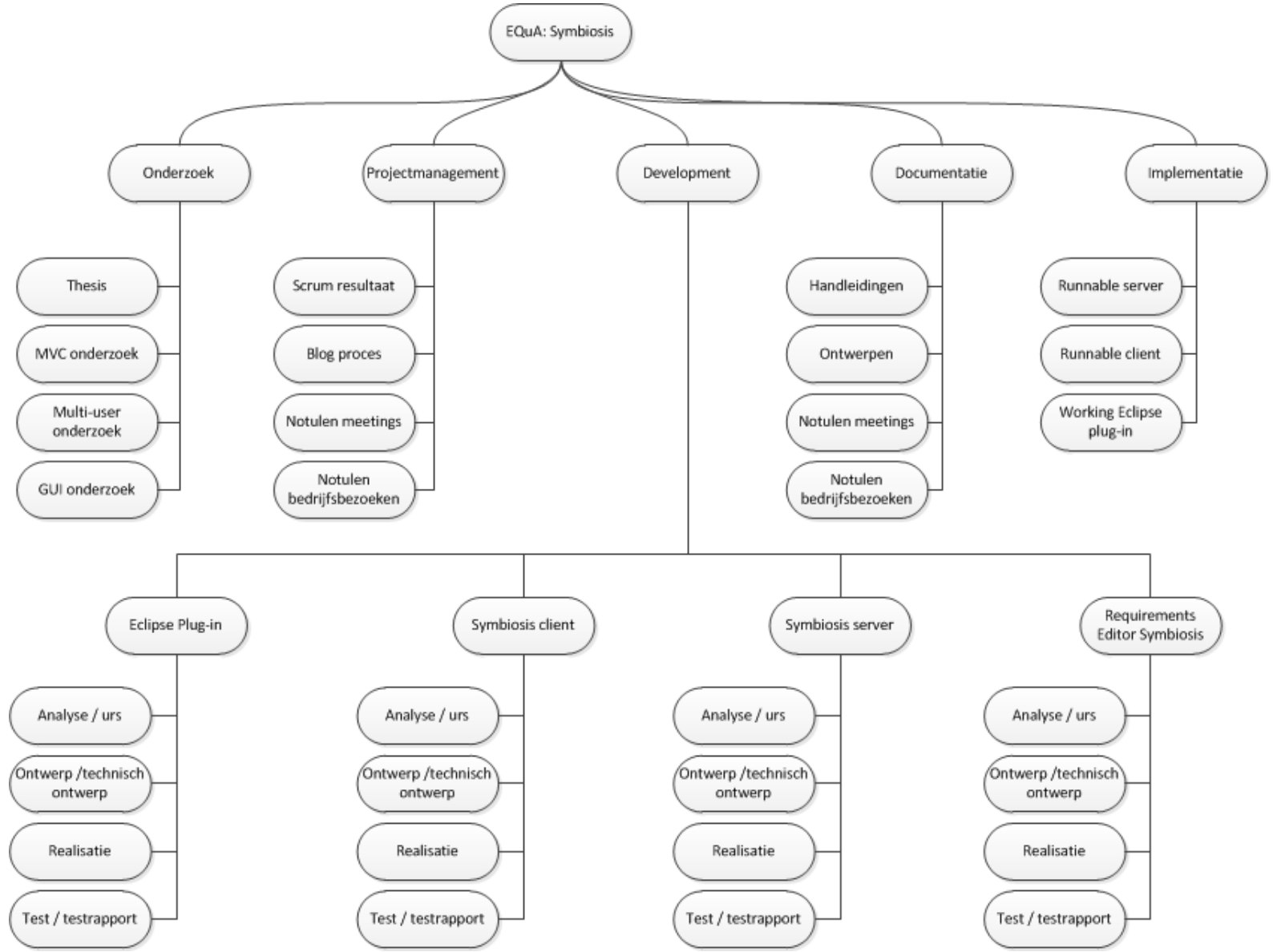
2.4.1 Taken

Werken aan het project:
- Plannen;
- Ontwerpen;
- Uitvoeren;
- Documenteren;
- Presenteren.

3 Afspraken

- Er zal een wekelijkse meeting plaatsvinden tussen de stagiair en de opdrachtgever.
- Er zal een dagelijkse voortgangsrapportage op de blog pagina komen. (www.sharpcoding.nl/EQuA)
- Er zal een wekelijkse meeting plaatsvinden tussen de stagiair en de bedrijfsbegeleider.

4 Work Breakdown Structure (WBS)



5 Planning

Hier onder vind je een overzicht van de planning van het project. Voor een meer gedetailleerde (wekelijkse) planning kun je terecht op de Scrum tool Acunote (vereist autorisatie) op equa.acunote.com.

Een overzicht en schema van de planning zijn beschikbaar op http://sharpcoding.nl/EQuA/Planning/ .

	Naam	Duur	Aanvang	Voltooiing	Voorgangers
1	**Start afstudeer project**	**16d**	**11/02/2013**	**04/03/2013**	
2	Vooronderzoek	10d	11/02/2013	22/02/2013	
3	Gesprek opdrachtgever	1d	18/02/2013	18/02/2013	
4	PID	15d	11/02/2013	01/03/2013	
5	1e bedrijfbezoek (Docentbegeleider)	1d	04/03/2013	04/03/2013	4
6	Proces blog opzetten	1d	14/02/2013	15/02/2013	
7	Scrum tool opzetten	1d	14/02/2013	15/02/2013	
8	Terugkomdag (School)	1d	26/04/2013	26/04/2013	
9	**Uitvoering project**	**67d**	**11/03/2013**	**11/06/2013**	
10	MVC onderzoek + ombouwen applicatie	10d	11/03/2013	22/03/2013	
11	Multi user onderzoek	10d	11/03/2013	22/03/2013	
12	Eclipse plug-in (multi user)	35d	25/03/2013	10/05/2013	10,11
13	Symbiosis server application (multi user)	35d	25/03/2013	10/05/2013	10,11
14	Symbiosis client application (multi user)	35d	25/03/2013	10/05/2013	10,11
15	Client - server tests	3d	15/05/2013	20/05/2013	12,13,14
16	Requirement Editor changes	10d	20/05/2013	31/05/2013	
17	Scriptie	67d	11/03/2013	11/06/2013	
18	2e bedrijfsbezoek	1d	31/05/2013	31/05/2013	
19	**Afronding**	**10d**	**24/06/2013**	**05/07/2013**	
20	Afstudeerzitting	10d	24/06/2013	05/07/2013	

6 Kosten-baten analyse

Mvc maken applicatie	Kosten	Baten
URS	8 uur	Afbakening herstructurering
Technisch ontwerp	8 uur	Technische keuzes en implementatie staan gedocumenteerd.
Realisatie	16 uur	MVC opgezette applicatie (nodig voor Eclipse plug-in / client server)
Huidige applicatie onderzoeken	5 uur	Beter inzicht in de werking krijgen

Eclipse plug-in	Kosten	Baten
URS	8 uur	Afbakening ontwikkeling
Technisch ontwerp	8 uur	Technische keuzes en implementatie staan gedocumenteerd.
Realisatie	24 uur	Symbiosis ombouwen tot Eclipse plug-in
Testen	4 uur	Werking garanderen
Handleiding	2 uur	Betere bruikbaarheid

Eclipse plug-in (Client)	Kosten	Baten
URS	4 uur	Afbakening ontwikkeling
Technisch ontwerp	4 uur	Technische keuzes en implementatie staan gedocumenteerd.
Realisatie	16 uur	Eclipse plug-in als client
Testen	4 uur	Werking garanderen
Handleiding	2 uur	Betere bruikbaarheid

Symbiosis standalone (Client)	Kosten	Baten
URS	4 uur	Afbakening ontwikkeling
Technisch ontwerp	4 uur	Technische keuzes en implementatie staan gedocumenteerd.
Realisatie	16 uur	Symbiosis applicatie als client
Testen	4 uur	Werking garanderen
Handleiding	2 uur	Betere bruikbaarheid

Symbiosis (Server)	Kosten	Baten
URS	8 uur	Afbakening ontwikkeling
Technisch ontwerp	8 uur	Technische keuzes en implementatie staan gedocumenteerd.
Realisatie	40 uur	Symbiosis applicatie als server
Testen	10 uur	Werking garanderen
Handleiding	2 uur	Betere bruikbaarheid

7 Risicoanalyse

Nr.	Risico	Preventie maatregelen	Impact waarde (1-5)	Risico kans (1-5)	Waarde (1-25)
1	Een project lid kan niet meer meewerken aan het project	-	5	1	**5**
2	Verliezen van documentatie of andere tussenproducten	Back-ups maken en houden op meerdere plekken	4	1	**4**
3	Misinterpretatie van de eisen	Agile programmeren en een goede analyse uitvoeren, wekelijkse communicatie met de opdrachtgever.	4	2	**8**
4	Huidige applicatie voldoet niet aan hedendaagse standaard	-	4	3	**12**
5	Er wordt niet volgens de afgesproken procedures gewerkt	Zorgen dat iedereen het eens is met de afgesproken methodes en ze begrijpt	3	2	**6**
6	Een fase loopt uit	Uitlooptijd inplannen, voortgang elke week (Agile) controleren	4	2	**8**
7	De bestaande Symbiosis applicatie heeft bugs	Tijd inplannen voor uitloop.	3	3	**9**
8	Geen toegang tot de source code van een bestaande Symbiosis applicatie.	Oude versie van de applicatie beschikbaar houden.	3	2	**6**
9	De opdrachtgever komt met een wijziging	Agile programmeren en goede procedures opzetten voor wijzigingen	2	4	**8**

8 Bronnen

De gebruikte bronnen zijn:

- EQuA – ReqUI (found on dropbox: EQuA – ISAAC)
- Blokboek afstuderen februari – juli (Fontys portal)
- PID4 reader (Fontys portal)
- Vooronderzoek

9 Links

Gebruikte links voor het project:

- http://sharpcoding.nl/EQuA/Planning/ (Overzicht van de planning van het project)
- http://sharpcoding.nl/EQuA/ (Blog met dagenverantwoording)
- http://equa.acunote.com (Scrumboard, let op vereist login)

Bijlage B: Requirements Document

ReqUI
Requirements editor and validator

Use Cases and Quality Attributes

Version: 0.7
Date: 12-03-13
Author: F.Peeters

Document history

ver	date	modifications	distribution	request
0.1	27-01-13		Ion	
0.2	01-02-13	reviewstate	Ion, Petra	review
0.3	06-02-13	reviewing notes of Petra: versioning, vagueness, spelling/style	Ion Petra Maurice Waldo Ronald	review for information review review for information
0.4	01-03-13	Reviewing notes of Maurice; rectifications are marked in yellow		
0.5	04-03-13	Use case about configuring before decomposing added		
0.6	07-03-13	ReviewImpact deleted ToDo's added; See changes in green	Maurice	review
0.7	12-03-13	Revert ToDos in behalf of designed model elements Delegating vice owner	Maurice	review

1

Table of Content

Introduction

The requirementseditor and validator is a facility in behalf of all participants of a Symbiosis project. We distinguish two kind of participants: those who are responsible to develop parts of the application and those who are responsible to validate the requirements and the application. The developers are categorized as ProjectMember and the externals are called Stakeholder (see figure 1). Every requirement needs exactly one owner, to prevent conflicts. This person should approve his requirements, but he got the opportunity to delegate his responsibility to another stakeholder of the project. Of course, such a person should be expert on the relevant part of the domain.

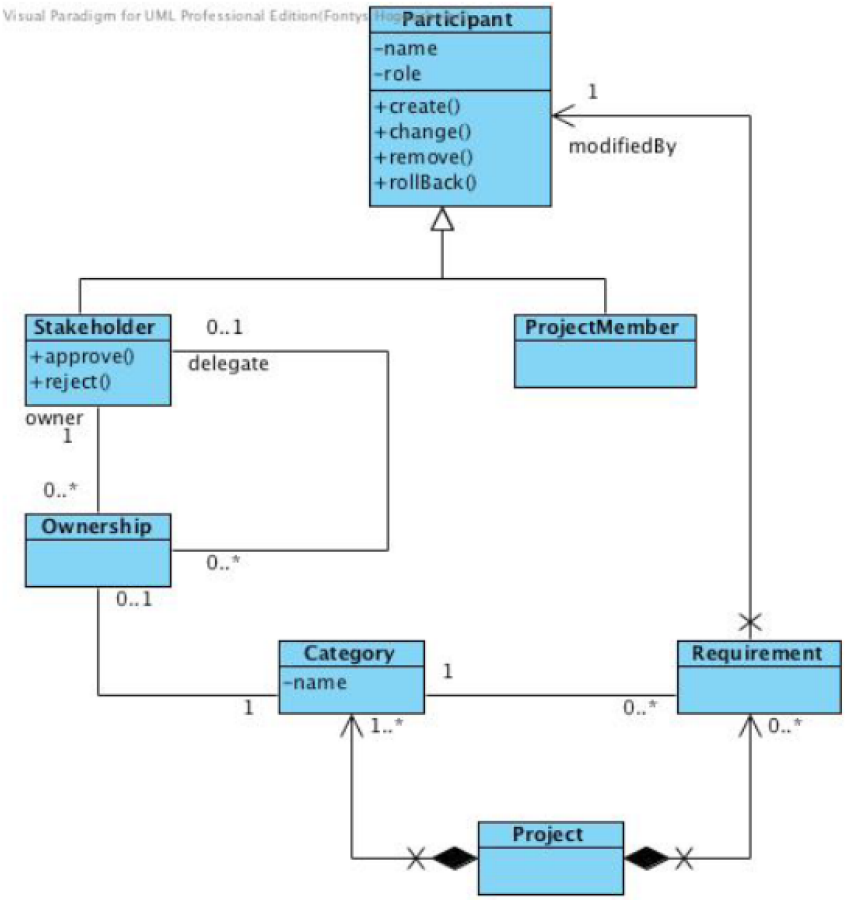


Figure 1: Participants and Responsibilities

Requirements do possess two partitionings: by *kind* and *category*. The kind partitions the requirements in Actions, Facts, Rules and Quality Attributes. The other one partitions requirements in subdomains or something like that. We do not want to be too restrictive, that is why we use the more or less neutral term of Category in stead of Subdomain. Every category needs an assignment to one *responsible* stakeholder; this rule implies that every requirement belongs to one stakeholder. Ownership is the title of such an assignment (see figure 1). Only the owner or the delegate of the category can approve and reject changes in the requirements belonging to this category. Every requirement belongs to exactly one category. As long as such assignment did not took place, the *Default* category will be used.

All participants of the project are allowed to create, change or remove requirements. If project members perform such action, the resulting changes should be validated by the owner or his delegate. If owner stakeholders cause a change, validation is not needed. We distinguish seven reviewing states of a requirement:

1. *Approved*: the requirement has been approved by the owner.
2. *Added*: the requirement has recently been added by a project member; the requirement needs validation by the owner.
3. *AddRejected*: the added requirement is rejected by the owner.
4. *Changed*: the requirement has been changed by a project member; the requirement needs validation by the owner.
5. *ChangeRejected*: the changed requirement is rejected by the owner.
6. *Removed*: the requirement has recently been removed by a project member; the removal needs an approval by the owner.
7. *RemoveRejected*: the proposal to remove the requirement is rejected by the owner.

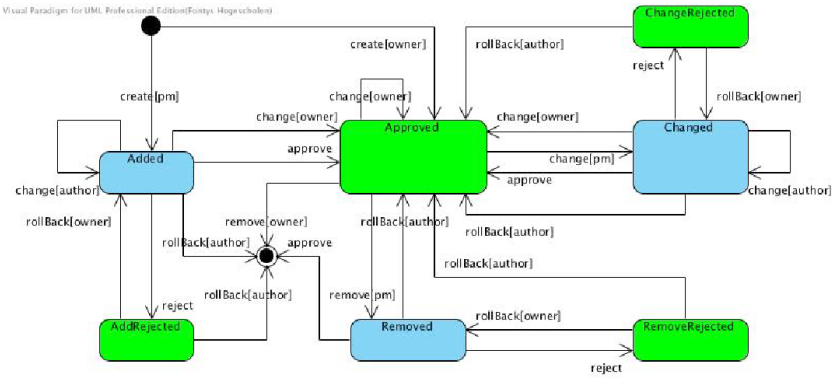


Figure 2.: State Transition Diagram of a Requirement.

In figure 2 one can read which transitions are allowed. The green states are caused normally by the owner and the blue states by a project member. A trigger of a transition can be extended with a guard, mentioned between brackets. The text *owner* refers to the owner stakeholder of the requirement. In that case only the owner or his delegate is allowed to execute the trigger. If the trigger is extended with the code *pm*, only a project member is allowed to produce the trigger. A trigger with an *author* guard requires the author of the concerning requirement. At last, an approval or rejection can only be triggered by the owner stakeholder.

Participants can do a roll back of their own editorial actions on a requirement. An author project member can roll back his creation, change or remove of some requirement. An owner, or his delegate, can roll back his approval or rejection. When an owner adds, changes or removes a requirement, such an action will be accepted unconditionally. If an owner wants to withdraw his change, he cannot use a similar roll back facility. But he will get a facility to return to one of the previous approved versions.

If the state is Removed, RemovedRejected, Changed or ChangedRejected, a requirement is still linked to the previous approved state. One could say, the state of such requirement is a bit dubious, it exists in nowhere land. A requirement in an Added state resists also in nowhere land, but in that case there is of course no previous approved state.

Keep attention: only changes on the content of a requirement, by a participant who is not the owner, need validation. Other modifications are marked but do not need explicit validation by the owner.

Requirement aspects

In addition to the content of a requirement, we want to decorate a requirement with various aspects about importancy, planning, testing, realization etc. The next table summarizes the aspects which are necessary or nice to have.

Aspect	Description	Values
ChanceOfFailure	What is the expected complexity to realize this requirement.	Low (1), Medium (2), High (3)
Importance	How big is the impact if this requirement isn't realized at last.	Zero (0), Light (1), Normal (2), Serious (3)
Risk	How risky is this requirement (Derivable property defined as the product of ChanceOfFailure and Importance)	0..9
Urgency	How early, within the planning, needs this requirement to be realized	Low, Medium, High
Moscow		Must, Should, Could, Wont
Verify method	Normally a requirement needs to be verified eventually.	None, Test, Demonstration, Inspection, Analysis
Realized	Did this requirement already get incorporated within the design; if ReviewState is not approved Realized will be always No. (Derivable property)	Yes, No
ReviewState	See figure 2 (Derivable property)	Approved, Added, AddRejected, Changed, ChangeRejected, Removed, RemoveRejected
ReviewImpact		Zero, Light, Normal, Serious
~~ToDo's~~	~~With a description (and/or cause) and the impact of a to-do~~	~~Zero, Light, Normal, Serious~~
Sources	Every source has a CreatedAt and ModifiedAt timestamp; Every source has a text with for instance the name of the author etc (see derived classes of equa.meta.traceability.Source)	
History	Every approved requirement retains all previous, approved, states of this requirement	

5

Use Cases

'Cancel scenario' means that the execution of the use case scenario did not influence the state of the system at the start van the use case.

Uc1a	Enter requirement
Actor	Project Member
Pre	
Normal flow	1. Actor • selects kind of requirement • selects category of requirement • enters content of requirement • enters justification of requirement (optional) • configures requirement aspects 2. System adds requirement to requirements model of project
Post	New requirement is added; state is ADDED
Alternate flow	.1 Empty content 1. Actor reenters step 1 or cancels scenario

Uc1b	Enter requirement
Actor	Stakeholder
Pre	Stakeholder is owner or delegate of selected category
Normal flow	1. Actor • selects kind of requirement • selects category of requirement • enters content of requirement • enters justification of requirement (optional) • configures requirement aspects 2. System adds requirement to requirements model of project
Post	New requirement is added; state is APPROVED
Alternate flow	.1 Empty content 1. Actor reenters step 1 or cancels scenario

Uc2a	Change requirement
Actor	Project Member
Pre	Selected requirement is ADDED, APPROVED or CHANGED
Normal flow	1. Actor configures kind, content, justification or aspects 2. System saves changes
Post	Changes are stored; If content has been changed requirement is stored as CHANGED, except when precondition is ADDED, then the state stays ADDED
Alternate flow	.1 Empty content 1. Actor reenters step 1 or cancels scenario

Uc2b	Change requirement
Actor	Stakeholder
Pre	Selected requirement is CHANGED, APPROVED or ADDED
Normal flow	1. Actor configures kind, category, content, justification or aspects 2. System saves changes
Post	Changed requirement is stored as APPROVED except when category has been changed then requirement is stored as CHANGED
Alternate flow	.1a Empty content 1. Actor reenters step 1 or cancels scenario. 1b Stakeholder is not the owner of the selected requirement 1. System cancels scenario

6

Uc2c	Revert content requirement
Actor	Stakeholder
Pre	Selected requirement is APPROVED
Normal flow	1. Actor selects previous approved content 2. System saves changes
Post	Changed requirement is stored as APPROVED, dependent model elements got an unreliable state
Alternate flow	

Uc3a	Remove requirement
Actor	Project Member
Pre	Selected requirement is APPROVED
Normal flow	1. Actor removes requirement 2. Actor enters justification (optional) 3. System saves deletion
Post	Requirement is stored as REMOVED
Alternate flow	

Uc3b	Remove requirement
Actor	Stakeholder
Pre	Selected Requirement is APPROVED
Normal flow	1. Actor removes requirement 2. Actor enters justification (optional) 3. System saves deletion
Post	Requirement is deleted from requirements model
Alternate flow	.1 Stakeholder isn't the owner of selected requirement. 1. System cancels scenario

Uc4a	Roll back
Actor	Project member
Pre	Selected Requirement is ADDED, CHANGED or REMOVED
Normal flow	1. Actor rolls back requirement towards previous approved state 2. System executes roll back
Post	Requirement returns to previous approved state
Alternate flow	1 Project member is not the author of selected requirement. 1. System cancels scenario

Uc4b	Roll back
Actor	Stakeholder
Pre	ADD_REJECTED, CHANGE_REJECTED, REMOVE_REJECTED
Normal flow	1. Actor rolls back requirement towards previous state 2. System executes roll back
Post	Requirement returns to previous state
Alternate flow	1 Stakeholder is not the owner of selected requirement. 1. System cancels scenario

Uc5	View requirements
Actor	Participant
Pre	
Normal flow	1. Actor activates filters (Category, Kind, Ownership, ReviewState, …) 2. System shows filtered requirements

7

Post	Selected requirements are shown; for every requirement one can see all aspects, including the progress on the filtered requirements
Alternate flow	

Uc6	Validate requirement
Actor	Stakeholder
Pre	Selected requirement is ADDED, CHANGED or REMOVED
Normal flow	1. Actor approves or rejects requirement and enters justification (optional) 2. System executes validation

Post	Selected requirement is validated
	If (pre=CHANGED and realized) and post=APPROVED Then dependent model elements got an unreliable state
	If (pre=REMOVED and realized) and post=deleted then dependent model elements got an unreliable state

Alternate flow	.1 Stakeholder is not the owner of selected requirement 1. System cancels scenario

Uc7	Enter new category
Actor	Participant
Pre	
Normal flow	1. Actor enters name of category 2. Actor selecte owner stakeholder of category (optional) 3. System creates new category
Post	New category is created
Alternate flow	.1 Category name is already in use 1. Actor goes back to step 1 or cancels scenario

Uc8	Remove category
Actor	Participant
Pre	
Normal flow	1. Actor selects category 2. Actor deletes selected category 3. System requests for confirmation 4. Actor confirms 5. System deletes category
Post	Category is deleted
Alternate flow	.2 There are requirements which are categorized by this category 1. System shows message with advise to migrate requirements to right category 2. System cancels scenario .4 Actor cancels deletion 1. System cancels scenario

Uc9	Edit category
Actor	Participant
Pre	
Normal flow	1. Actor selects category 2. Actor edits name category and/or chooses owner 3. System saves changes
Post	Category changes are saved
Alternate flow	.2a New name is already in use 1. Actor goes back to step 2 or cancels scenario .2b Owner was already defined 1. System cancels scenario

8

Uc10a	Change owner category
Actor	Stakeholder
Pre	
Normal flow	1. Actor selects category 2. Actor changes owner of category 3. System saves changes
Post	New owner of Category saved; ReviewState of every requirement of this category is CHANGED
Alternate flow	.2b Stakeholder is not the current owner of the category 1. System cancels scenario

Uc10b	Set vice owner category
Actor	Stakeholder
Pre	
Normal flow	1. Actor selects category 2. Actor changes or removes vice owner of category 3. System saves changes
Post	Vice owner of Category saved or removed
Alternate flow	.2b Stakeholder is not the current owner of the category 2. System cancels scenario

Uc11	Move requirements
Actor	Participant
Pre	
Normal flow	1. Actor selects source and destination category 2. System request for confirmation 3. Actor confirms move 4. System moves all requirements of source category towards destination category
Post	Original requirements from source category are belonging now to destination category; If owner of destination category differs from owner of source category then ReviewState of every moved requirement of this category is CHANGED
Alternate flow	.2 Selected source category does possess a owner who is not the same as the actor 2. System cancels scenario

Uc12	Configure Decompose fact requirements
Actor	Participant
Pre	
Normal flow	1. Actor selects category, reviewState, aspects 2. System shows filtered fact requirements
Post	Filter is activated
Alternate flow	

Uc13	Decompose fact requirement
Actor	Participant
Pre	Filter (category, reviewState, aspects) is activated
Normal flow	1. Actor selects fact 2. Actor decomposes fact 3. System adds result of fact decomposition to object model
Post	Fact is realized in object model
Alternate flow	.2 Decomposition is aborted 1. System exits scenario; partial decomposition is realized in object model .2 Fact is rejected 1. Scenario canceled

9

Uc14	Roll back decomposition
Actor	Participant
Pre	
Normal flow	1. Actor selects fact 2. Actor rolls back decomposition 3. System requests for confirmation 4. Actor confirms 5. System removes fact from object model
Post	Fact is removed from object model
Alternate flow	.4 Actor cancels 1. Scenario canceled

10

Quality Attributes

The editor and validator must meet with an important quality attribute on usability. The modal stakeholder and modal project member should experience that this editor and validator makes it possible to enter, change and validate requirements in an easy way. Moreover, it should be possible to monitor the progress of the software development by means of an aggregation of the progress per requirements. Last but not least, the project members can co-operate with the tool while working parallel on the the same project, without experiencing nasty delays or inconsistencies caused by multi-user conflicts.

Qa.1 75% of the stakeholders shall confirm the statement: the requirements can be validated in a proper and easy way.

Qa.2 90% of the project members shall confirm the statement: the requirements can be edited in a proper and easy way.

Qa.3 75% of the project members shall confirm the statement: the progress in the development of the application can be monitored with the help of metrics based on the progress in the individual requirements.

Qa.4 Individual project members can work on the project without negative influences caused by interaction by other project members in the same project.
<SMART expression of this attribute needs to be figured out>

11

Bijlage C: Usability test

EQuA – Symbiosis: Multi user aspects

Usability Test

Domein gegevens

Hier onder vindt je de gegevens die je nodig hebt bij het uitvoeren van de test case. Als je deze usablity test afneemt gaan we er vanuit dat je bekend bent met de EQuA methodiek. Na de test is er een korte enquête die ingevuld moet worden.

User Requirements Specification
Context KNVB systeem
A. Feiten Fact1: VVV – Ajax uit ronde 6 van de eredivisie van seizoen 2011 is in 2-2 geëindigd. Fact2: MVV neemt deel aan de eerste divisie van seizoen 2011. Fact3: Bij de wedstrijd van Heracles tegen Feyenoord uit ronde 6 van de eredivisie van seizoen 2011 waren 13450 toeschouwers aanwezig.
B. Constraints Constr1: tussen twee opeenvolgende wedstrijden van een team zitten ten minste twee wedstrijdloze dagen. Constr2: de (tussen)uitslag van een wedstrijd kan pas bekend zijn als de wedstrijd is begonnen.
C. Kwaliteitsattributen QA1: [performance] gemelde wijzigingen in de uitslagen van de wedstrijden worden binnen één minuut verwerkt. QA2: [schaalbaarheid] wedstrijdhistorie over alle knvb-competities ligt over 30 seizoenen terug publiekelijk ter inzage.

Test Case:

A. Start de Symbiosis applicatie en creëer een nieuw project.
   a. Vul als er om wordt gevraagd "Project 1" in bij projectnaam
   b. Schrijf bij Creator name: je eigen naam
   c. Schrijf bij Creator role: Tester
   d. Kies ervoor om het project als stakeholder aan te maken.

B. Vul de Feiten en kwaliteitattributen beschreven in de User Requirement Sepcification in de applicatie. (de constraints pakken we later aan)
   a. Kopieer en plak de bovenstaande zinnen in het veld van de requirement.
   b. Let goed op het type requirement. De Feiten moeten worden ingevoerd als Fact requirement en de kwaliteitsattributen als Quality Attributes.
   c. Laat de overige velden staan op de standaardwaarden.

C. Filter de requirements met de volgende filter opties:
   a. Category: Default
   b. ReviewState: Approved
   c. Kind: Fact Requirement
   d. Other filters: Owner

D. Selecteer Fact1 en voer een fact Breakdown uit.
   a. Als typeName gebruik: UitslagWedstrijd
   b. Markeer het volgende deel: "VVV – Ajax uit ronde 6 van de eredivisie van seizoen 2011" en geef deze de typeName: Wedstrijd
   c. Markeer nu het volgende deel: "VVV" en geef deze de typeName: Team en laat het de rol vervullen van thuisTeam
   d. Markeer nu het volgende deel: "VVV" (nieuwe node) en kies hier voor de baseType String en laat het de rol vervullen van teamNaam.
   e. Markeer nu het volgende deel: Ajax en kies hier voor een bestaand type: Team geef deze echter de rol: uitTeam
   f. Markeer: "ronde 6 van de eredivisie van seizoen 2011" en geef deze de typeName: Ronde.
   g. Markeer: "6" en kies hierbij voor de baseType Natural en laat het de rol vervullen van "rondenummer".
   h. Markeer: "2-2" en geef deze de typeName: Uitslag
   i. Markeer het eerste getal van de Uitslag en kies hier voor de baseType Natural en laat het de rol vervullen van "doelpuntenThuis".
   j. Markeer het tweede getal van de Uitslag en kies hier voor de baseType Natural en laat het de rol vervullen van "doelpuntenUit".
   k. Voltooi de Fact Breakdown.

Enquête: Symbiosis GUI

Vraag 1: Wat is je afstudeerrichting?

A. ICT & Business
B. ICT & Media Design
C. ICT & Software
D. ICT & Technology
E. Anders: _____

Vraag 2: Zie je jezelf een applicatie ontwikkelen middels de EQuA methodiek?

A. Ja, de methodiek werkt erg prettig en zou hier in de toekomst vaker gebruik van willen maken.
B. Ja, mits de applicatie verbeterd wordt.
C. Ik twijfel nog. Wat is een struikelblok voor het gebruik van de EQuA methodiek?
   _____
D. Nee, de methodiek vind ik niet prettig werken: Wat zou je graag anders zien?
   _____

Onlangs zijn er een hoop wijzigingen gemaakt aan de Symbiosis GUI. De volgende vragen hebben hier betrekking op.

Vraag 3: Heb je voor de GUI wijzigingen al eens met de tool gewerkt?

A. Ja
B. Nee

Vraag 4: Wat vindt je van de algemene look van de applicatie?

A. Zeer goed
B. Goed
C. Matig
D. Slecht

Geef aan of je het met eens bent met de volgende stellingen:

Vraag 5: Het is eenvoudig om requirements in te voeren binnen de applicatie.

A. Helemaal eens
B. Mee eens
C. Neutraal
D. Oneens
E. Helemaal oneens;

Vraag 6: Requirements zijn eenvoudig aan te passen.

A. Helemaal eens
B. Mee eens
C. Neutraal
D. Oneens
E. Helemaal oneens;

Vraag 7: Requirements kunnen eenvoudig gefilterd worden binnen de requirements viewer.

A. Helemaal eens
B. Mee eens
C. Neutraal
D. Oneens
E. Helemaal oneens;

Vraag 8: De breakdown van FactRequirements kan op een prettige manier uitgevoerd worden.

A. Helemaal eens
B. Mee eens
C. Neutraal
D. Oneens
E. Helemaal oneens;

Vraag 9: Met de typeConfigurator is het eenvoudig om objecten naar wens te configureren.

A. Helemaal eens
B. Mee eens
C. Neutraal
D. Oneens
E. Helemaal oneens;

Vraag 10: Requirements zijn eenvoudig te valideren door een stakeholder.

A. Helemaal eens
B. Mee eens
C. Neutraal
D. Oneens
E. Helemaal oneens;

Bedankt voor het invullen van deze enquête. Mocht je nog opmerkingen hebben betreffende de applicatie heb je daar hieronder de mogelijkheid voor.

_____

_____

_____