Afstudeerverslag

Outsourcing van de onderhoudsfase van CCure: Automatisering van oncologische behandeltrajecten

Lucas van Dongen
Tilburg
Juni 2012

Titelpagina

Naam	Lucas van Dongen
Studentnummer	2088881
Afstudeerperiode	1 februari tot 1 juli 2012
Opleiding	HBO Informatica, Fontys Hogeschool Eindhoven
Afstudeerrichting	Software Engineer
Project	Outsourcing van de onderhoudsfase van CCure: Automatisering van oncologische behandeltrajecten
Stagebedrijf	Instituut Verbeeten
Bedrijfsmentor	Ing. Eelko Hondema
Afstudeerdocent	Dr. Martijn Lamers

Instituut Verbeeten

Vestiging Tilburg

Brugstraat 10

5042 SB Tilburg

Telefoon: +31 (0)13 594 77 77

Telefax: +31 (0)13 594 76 83

Fontys Hogeschool

Afdeling ICT

Rachelsmolen 1

5612 MA Eindhoven

Telefoon: +31 (0)8850 77299

Voorwoord

Voordat ik bij Instituut Verbeeten begon met werken had ik nog nooit in een omgeving gewerkt waar het leven en de dood me iedere dag zo nabij stonden. De eerste weken moest ik enorm wennen aan het constante thema kanker, en de confrontatie met soms toch wel erg zieke mensen zowel in het Instituut zelf, als vertegenwoordigd als dossiers in de software. Het gaf een totaal nieuwe betrokkenheid aan mijn werk die ik voorheen nooit gevoeld had. Een fout in de software is hier niet langer vervelend of duur, maar heeft echt invloed op een mensenleven. En ik merkte om mij heen dat eenieder die ik ontmoette binnen het Instituut Verbeeten een enorme betrokkenheid en gevoel van verantwoordelijkheid had.

Dit enthousiasme en betrokkenheid, ook binnen de afdeling ICT zelf heb ik als enorm positief en stimulerend ervaren. Ook de manier waarop er ontwikkeld werd, met een volwaardige OTAP-omgeving, een gespecialiseerde tester en key users met enorm veel kennis over het proces, en het gebruik van de nieuwste technieken waren eerder zaken die bij vorige opdrachtgevers nooit zo professioneel opgezet waren.

Het proces van afstuderen heb ik als enorm leerzaam ervaren, met name de balans vinden tussen het werken aan de releases voor CCure, mijn in deze periode zeer drukke en turbulente privé-leven en het afstudeerproject zelf hebben me een hoop nieuwe inzichten in mezelf verschaft. Ook ben ik tot de conclusie gekomen dat als je echt keuzes maakt in wat je zelf belangrijk vindt in plaats van te denken wat andere mensen belangrijk vinden, je een stuk verder komt.

Ook ben ik tot de conclusie gekomen dat de kunst van afstuderen niet zo zeer draait om het kunnen ontwerpen en vervolgens programmeren van software, maar om het managen van het hele circus er om heen. Het arrangeren van een opdracht, daar voldoende aandacht voor zien te krijgen binnen de organisatie en de combinatie met alle andere activiteiten waar een deeltijdstudent in zijn dagelijkse leven mee te maken heeft.

Graag wil ik aan de volgende mensen mijn dank betuigen: Marcel Maes en Eelko Hondema voor het creëren van de mogelijkheid tot afstuderen en het vertrouwen. Bernard van Erp voor het creëren van de ruimte in de planning en in het algemeen zijn interesse in het toepassen van nieuwe technieken en verbetering van onze werkwijze, wat ik goed heb kunnen toepassen in mijn opdracht. Laura Bonifasi voor de mentale steun en haar geduld. En iedereen die me in het afgelopen roerige half jaar geholpen heeft.

Tilburg, juni 2012

Inhoudsopgave

Samenvatting 6
Summary 7
Verklarende woordenlijst 8
1 Inleiding 11
2 Het Instituut Verbeeten 12
2.1 Over het instituut.....12
2.2 Geschiedenis.....14
3 Opdracht 15
3.1 Aanleiding voor de afstudeeropdracht.....15
3.2 Omschrijving van de afstudeeropdracht.....15
3.3 Deliverables.....16
3.3.1 Onderzoek 16
3.3.2 Presentatie 16
3.3.3 Verbeteringen ontwikkelproces 17
4 Over CCure 18
4.1 Wat doet CCure?.....18
4.2 Wat is de geschiedenis van CCure?.....20
4.3 Wat is de toekomst van CCure?.....21
4.4 Welke technieken worden er gebruikt in CCure?.....22
4.4.1 Architectuur 22
4.4.2 Windows Workflow Foundation 23
4.4.3 Windows Presentation Foundation 24
4.4.4 Catel MVVM Framework 24
4.4.5 LLBLGen 25
4.4.6 MediSpeech van G2Speech 26
5 Aanpak 27
5.1 Onderzoeksgedeelte.....27
5.1.1 Vooronderzoek 27
5.1.2 Interviews 27
5.1.3 Adviesrapportage 28
5.2 Praktijkgedeelte.....28
5.2.1 Inventarisatie 29
5.2.2 Uitvoering 29
6 Onderzoek 30
6.1 Wat is outsourcing?.....30
6.2 Welke vormen van outsourcing bestaan er?.....30
6.2.1 Offshoring 30
6.2.2 Nearshoring 31
6.2.3 Closeshoring 32
6.3 Welke andere opties zijn er?.....33
6.3.1 Inhuur 33
6.3.2 Vaste krachten 33
6.4 Hoe werden de scenario's beoordeeld?.....33
6.4.1 Communicatie 33
6.4.2 Waarborging kennis 34
6.4.3 Waarborging kwaliteit 34
6.4.4 Testen en acceptatie 34

6.4.5 Financieel 35
6.4.6 Juridisch 35
6.4.7 Menselijk aspect 35
6.5 Conclusies per scenario.....36
6.5.1 Offshoring 36
6.5.2 Nearshoring 36
6.5.3 Closeshoring 36
6.5.4 Inhuur 36
6.5.5 Vaste krachten 37
6.6 Eindadvies na onderzoek outsourcing.....38
7 Verbeteringen in het ontwikkelproces 39
7.1 Inleiding.....39
7.2 Geautomatiseerd testen.....39
7.2.1 Unit Testing 40
7.2.2 Integration Testing 41
7.2.3 Code coverage 41
7.2.4 Geautomatiseerd testen in CCure 42
7.3 Documenteren van functionaliteit 42
7.3.1 Bestaande documentatie 42
7.3.2 Schermontwerpen 43
7.3.3 Use case scenario's 45
7.4 Best practices vastleggen.....47
7.4.1 Stamtabelbeheerscherm 48
7.4.2 Scherm met ViewModel 50
7.4.3 Workflow 50
7.4.4 ValueConverter 50
7.4.5 Helper 51
7.5 Coding Guidelines.....51
8 Conclusies en aanbevelingen 52
Evaluatie 53
Appendix A: Project Initiatie Document 54
Appendix B: Geautomatiseerde tests email validatie 58

Samenvatting

Dit document bevat het eindverslag van het afstudeerproject van Lucas van Dongen voor Fontys Hogeschool bij het Instituut Verbeeten te Tilburg. Het Instituut Verbeeten is een ziekenhuis dat gespecialiseerd is in radiotherapeutische oncologie en nucleaire geneeskunde, en heeft in-house software ontwikkeld ter ondersteuning van het gehele oncologische behandelproces binnen het instituut, genaamd CCure.

CCure is al geruime tijd in ontwikkeling en gaat nu de onderhoudsfase in. In het verleden is er veel geld gestoken in de ontwikkeling van het pakket, en hebben ook veel verschillende programmeurs gewerkt aan de software waardoor er veel verschillende manieren van ontwikkelen terug te vinden zijn. Hierdoor is de vraag enerzijds is gerezen om te kijken naar kostenbesparende maatregelen zoals outsourcing, maar ook om de testbaarheid, onderhoudbaarheid en overdraagbaarheid van de applicatie in zijn algemeenheid te onderzoeken en verbeteren.

Hiertoe is enerzijds een onderzoek ingesteld naar de mogelijkheden en risico's bij het outsourcen van CCure, waarbij zowel mensen met diverse expertises zijn geraadpleegd binnen het instituut, als een onderzoek is gepleegd in literatuur om een beeld te krijgen over de algemene ervaringen bij het outsourcen van software. Ook is er onderzocht welke problemen er nu nog spelen rond het project waardoor outsourcing risicovoller of onmogelijk wordt, met het idee in het achterhoofd dat veel verbeteringen ook bij het niet doorzetten van outsourcing toch een waardevolle verbetering blijven het project zijn in de onderhoudsfase.

De onderzoeksresultaten rondom outsourcing zijn verwerkt in een rapportage, en worden gepresenteerd aan de Raad van het Bestuur. Het onderzoek naar probleempunten in het project heeft geresulteerd in een reeks aan aanbevelingen op het gebied van proces, testen, documentatie en manier van het schrijven van code. Op ieder gebied is alles direct in de praktijk getest en daar waar de ervaringen positief waren zijn deze direct doorgevoerd.

Doordat het project reeds in de onderhoudsfase zit en de hoeveelheid nog te ontwikkelen software niet zo groot meer is, is het niet kosteneffectief om dit nog onder te brengen in een ander land, gezien de initiële aanlooptijd en benodigde investeringen. Hooguit kan closeshoring een alternatief zijn, mits het team voor een deel bestaat uit mensen die eerder intern bij het instituut gezeten hebben.

Rondom de verbeteringen aan de software en het proces zijn er veel zaken te benoemen en aan te bevelen, maar de speerpunten uit dit onderzoek zijn dat het doorvoeren van geautomatiseerd testen voor reeds bestaande code niet lonend is behalve voor compleet nieuw te ontwikkelen delen. Het documenteren van nieuwe functionaliteit en changes kan sterk verbeterd worden. De kennis en ontwikkelstandaarden rondom de applicatie moeten sterk verbeterd worden.

Summary

This document is a report about the thesis project for the graduation internship of Lucas van Dongen for the Fontys University at the Institute Verbeeten in Tilburg. The institute is a hospital specialized in nuclear medicine and radiation oncology, and in the past decade it has developed it's own software solution to support the whole process of diagnose and treatment in the institute, called CCure.

After automating the whole treatment process from intake until aftertreatment, the software development cycle has entered the maintenance phase. The investments into CCure have been quite vast and over a long period of time there were a lot different software engineers involved with their own habits and coding styles. Because of this is that the institute is looking for ways to minimize the maintenance costs, for instance by outsourcing the software, but also to improve the testability, maintainability and portability of CCure in general.

The assignment that came out of this was to research the possibilities and risks regarding outsourcing of the maintenance and future development of CCure, and to investigate what issues in the code and development process would be causing problems if the project would be outsourced. The recommendations for the optimization of the code and development process would also be used even if no outsourcing trajectory was started.

A report has been made with the results of the outsourcing research, followed by a presentation to the board. The investigation of issues in the code and development process resulted in a range of recommendations, example code and documents regarding the process, testing, documenting and coding guidelines. Every recommendation has been put to the test in practice and most of them were judged as improvements and are being in use now.

After conducting the research and weighing in all factors, the conclusion was that offshoring the project would not be cost effective, because the initial time and costs that would be needed would not be earned back anymore at this stage of the project. Closeshoring could be a alternative, as long as the team also contains engineers that have previously been working at the institute on the project.

The improvements regarding code and process are manyfold and not all can be dealt with separately in this short space. However it's concluded that creating automated tests for existing code is not very economic, and only new parts built from the ground up with automated testing will reap benefits from them. Documenting changes and new functionality needs to be improved. Knowledge about the application and standards for developers need to be documented and acted upon.

Verklarende woordenlijst

Bug, Bugfix	Een bug is een probleem in software die veroorzaakt wordt doordat een onderdeel niet betrouwbaar of niet volgens specificaties werkt. Een bugfix is een oplossing voor zo'n probleem. Het komt vaker voor dat een klant na testen er achter komt dat de specificaties niet correct waren omdat het niet helemaal in het proces past, of dat de specificaties simpelweg niet correct bleken te zijn. In zo'n geval spreken we niet over een bug, maar moet er een change request ingediend worden.
Change (request)	Een change is een wijziging in de functionaliteit van reeds gebouwde software, dan wel een wijziging aan de oorspronkelijke specificaties op het moment nadat de bouw van de software al gestart is, maar de specifieke feature nog niet opgeleverd is. Zolang het nog om een wijziging in het oorspronkelijke ontwerp gaat nog voordat er aan de bouw is begonnen, of het een oplossing voor een bug betreft, is het geen change request.
CT-scan, CAT-scan, Computertomografie	Computertomografie (meestal afgekort tot CAT of CT-scan) is een tomografische onderzoeksmethode van het menselijk lichaam. Een van de eerste vormen hiervan maakte gebruik van röntgenstraling. De doorlaatbaarheid van het onderzochte lichaamsdeel voor de gebruikte straling wordt vanuit een groot aantal hoeken rondom in dunne „plakjes" gemeten, waarna een computer uit de resultaten een driedimensionale weergave van het onderzochte lichaamsdeel opbouwt. Tegenwoordig wordt ook veel gewerkt met magnetische resonantie; men spreekt dan van Magnetic Resonance Imaging of MRI.[1]
Edge cases	Het vaststellen wanneer iets nog wel werkt, en wanneer het niet meer zou mogen werken. Wordt dikwijls gebruikt om functies mee te definiëren en vervolgens te testen. Bijvoorbeeld het proberen uit te voeren van een functie die altijd een object verwacht als parameter met een null-referentie in plaats van een instantie van dat object. Dat zou altijd een uitzondering moeten geven, en dus niet een resultaat of alleen een null-referentie als uitvoer terug gegeven. Maar ook zou het ophalen via het id van een database record dat daadwerkelijk in de database bestaat altijd iets terug moeten geven. Zowel het database record als de uitzondering behoren tot de gewenste resultaten als het om het valideren van een edge case gaat. Vervolgens kunnen deze edge cases omgezet worden in Unit of Integration Tests en kan daarmee doorlopend getest worden of de functie nog aan deze edge cases voldoet.
Integration Testing	Integration testing betreft het geautomatiseerd testen van software wat verder gaat dan het testen van één afgesloten deelfunctie zoals bij een Unit Test, waardoor er een aantal stappen in een proces, een overkoepelende functie die meer stappen doorloopt, of een call

1 http://nl.wikipedia.org/wiki/Computertomografie

	over meerdere lagen binnen een applicatie getest wordt.
Key user	Een eindgebruiker met uitgebreide proceskennis die is verkozen om in het ontwikkelingsproces bij het ontwikkelen van specificaties te helpen en de opgeleverde software te accepteren
MVVM	Een afkorting voor **M**odel **V**iew **V**iew**M**odel, wat een design pattern is voor het presenteren van gegevens via een grafische interface. Model staat voor de gegevens, (datamodel), View staat voor de grafische interface, en ViewModel is een klasse die tussen het Model en de View in zit, waar normaliter er in de View (in een code-behind) de logica zit. Hierdoor is de logica in het ViewModel onafhankelijk van de interface geautomatiseerd te testen.
Normaliseren	Normalisatie is een proces waarbij gegevens in een database zo gestructureerd worden, dat er zo geen informatie meer dubbel opgeslagen wordt. Hiermee wordt er voor gezorgd dat de grootte van een database beperkt wordt, wat tegenwoordig niet zo'n groot probleem is als vroeger vanwege sterk toegenomen opslagcapaciteit. Ook wordt er voor gezorgd dat er minder kans is op inconsistente data, doordat bijvoorbeeld de zelfde data in de ene tabel wel bijgewerkt wordt, maar de andere vergeten wordt. [2]
Onderhoudsfase	Tijdsperiode in de software-levenscyclus waarin een software wordt gebruikt in zijn operationele omgeving, waarbij er op wordt toegezien of het naar bevrediging werkt en het, wanneer nodig, wordt aangepast om probleemgevallen te corrigeren of om aan gewijzigde behoeftes tegemoet te komen[3]
OR/Mapper, ORM	Staat voor **O**bject **R**elational **M**apper[4], wat staat voor een techniek die er voor zorgt dat databasetabellen toegankelijk worden als objecten, zodat deze weer makkelijk in te passen zijn in een OO-programmeeromgeving. Hierbij worden niet alleen de regels in een tabel omgezet in objecten, maar blijft ook de databasestructuur zoals de koppeling tussen tabellen door middel van keys intact.
OO, Object Oriented	**O**bjectge**O**riënteerd-programmeren[5] is een manier van het structureren van programma's waarbij er gewerkt wordt met objecten die een afgesloten doel en functie hebben, en die vaak ook voor een deel weerslag hebben op het proces waar de software voor ontwikkeld wordt. Het maakt het zowel makkelijker om te begrijpen hoe een programma in elkaar zit, als modulairder, doordat veel classes afgesloten stukjes functionaliteit zijn die (her)gebruikt kunnen worden zonder dat men precies hoeft te weten hoe ze intern functioneren.
OTAP	Is een afkorting voor **O**ntwikkel, **T**est, **A**cceptatie en **P**roductie, wat staat voor de stadia waarin de software volgens deze methode ontwikkeld wordt. De ontwikkelaars werken aan de ontwikkelversie,

2 Het SQL leerboek, Rick F. van der Lans, ISBN 90 395 2248 0
3 http://www.woorden-boek.nl/woord/operationele%20fase%20en%20onderhoudsfase
4 http://en.wikipedia.org/wiki/Object-relational_mapping
5 http://nl.wikipedia.org/wiki/Objectgeori%C3%ABnteerd

	welke alleen door de ontwikkelaars gebruikt wordt. Alle code die ontwikkeld is en ingecheckt, wordt via een nightly build gereed gezet op de testversie. Deze testversie wordt vervolgens getest door de testers, en op het moment dat die goedgekeurd is, wordt deze naar de acceptatie doorgezet. Bij de stap acceptatie gaan de acceptatietesters, meestal eindgebruikers met veel domeinkennis, testen of alles werkt volgens de specificaties. Als ook deze stap goedgekeurd is, wordt er een release gedaan naar de echte productieomgeving.
ReSharper	Een extensie voor Visual Studio ontwikkeld door JetBrains[6] welke extra functionaliteit toevoegt aan Visual Studio, waarmee de productiviteit of effectiviteit van de programmeur verbeterd wordt. Het helpt de programmeur betere code te schrijven door meer hints te geven dan Visual Studio zelf doet, het helpt de programmeur met het sneller schrijven van code doordat het meer sneltoetsen en automatische aanvulling heeft, en het helpt programmeurs met het verbeteren van code (refactoring).
Unit Testing	Het geautomatiseerd testen van de kleinste onderdelen van de ontwikkelde software, namelijk een functie of een constructor die geen andere zelf ontwikkelde functies aanroepen en maar één ding tegelijk doen. Zo gauw een functie meerdere dingen tegelijk doet, bijvoorbeeld een Initialize functie die weer drie andere functies aanroept, dan wordt het een integration test. Om code zo goed mogelijk testbaar te maken met Unit Testing, wordt het gestimuleerd om hele kleine, afgebakende functies te maken en deze ook kort te houden, desnoods door bepaalde delen van de code in een lange functie te refactoren naar losse functies.
Windows Workflow Foundation, WF	Een door Microsoft ontwikkelde library die (een daarvoor geschikt deel) de logica van een applicatie in zogenaamde Workflows onderbrengt. Een Workflow is een softwarematige versie van een activiteitendiagram

6 http://www.jetbrains.com/resharper/

1 Inleiding

Een vergelijking van uurtarieven tussen verschillende landen belooft in eerste instantie veel. Maar gaat outsourcing echt de besparing brengen die het in eerste instantie belooft? En is CCure in de huidige staat überhaupt te outsourcen?

Dat zijn de vragen waar het Instituut Verbeeten, een Tilburgs ziekenhuis gespecialiseerd op het gebied van radiotherapeutische oncologie en nucleaire geneeskunde mee worstelde.

Jarenlang was er met een flink budget in de software geïnvesteerd, en inmiddels was het project in de onderhoudsfase beland. Helaas ontbrak het binnen de organisatie aan tijd steeds aan tijd hier voor, en werd er een afstudeerder gezocht die met een verse blik op het project en wat meer tijd om dingen goed uit te kunnen zoeken en deze vragen eindelijk te beantwoorden.

De opdracht is tweeledig geweest, namelijk enerzijds het eigenlijke onderzoek naar de mogelijkheden en risico's van een outsourcing traject en aan de andere kant een meer praktische invulling waarbij er verbeteringen aan de software en aan het ontwikkelingsproces gedaan moeten worden die er voor zorgen dat het project beter overdraagbaar wordt.

Het onderzoek is uitgevoerd door allereerst een studie te doen naar alle factoren die van belang zijn bij een outsourcing traject, namelijk Communicatie, Waarborging kennis, Waarborging kwaliteit, Testen en acceptatie, Financieel, Juridisch en Menselijk aspect en te kijken waar dat van toepassing was voor Instituut Verbeeten. Vervolgens is door middel van interviews en verder onderzoek achterhaald wat de haalbaarheid was van diverse mogelijkheden tot outsourcing.

De verbeteringen in het proces en aan de software zijn gedaan na een eerst analyse van de knelpunten die er op dat moment lagen in het project op dat gebied, en welke risico's zouden kunnen opleveren op het moment dat de ontwikkeling geoutsourced zou worden. Voor deze problemen zijn oplossingen bedacht en in de praktijk getest. Daar waar de oplossingen daadwerkelijk verbeteringen opleverden zijn ze omgezet in nieuwe richtlijnen.

In de komende drie hoofdstukken wordt het project geïntroduceerd, met in Hoofdstuk 2 meer achtergrond over het instituut, in Hoofdstuk 3 de opdrachtbeschrijving, en in hoofdstuk 4 een korte uitleg van wat CCure is en hoe het ongeveer werkt. Daarna wordt in Hoofdstuk 5 uitgelegd hoe het project vervolgens aangepakt is, waarna in Hoofdstuk 6 en 7 de uitvoering van het onderzoek respectievelijk de verbeteringen aan het ontwikkeproces en de software. Daarna volgen in Hoofdstuk 8 de conclusies en aanbevelingen.

2 Het Instituut Verbeeten

Afbeelding 1: De hoofdvestiging in Tilburg

2.1 Over het instituut

Het Instituut Verbeeten is een ziekenhuis dat gespecialiseerd is in radiotherapeutische oncologie en nucleaire geneeskunde. Het instituut verzorgt het volledige traject van diagnose tot behandeling van kankerpatiënten. Het instituut werkt vanuit de overtuiging dat de patiënt centraal staat en recht heeft op zorg van hoogwaardige kwaliteit. De manier waarop met patiënten wordt omgegaan, is de kracht van het instituut. In de benadering van en contacten met patiënten wordt door alle betrokkenen binnen Instituut Verbeeten zoveel mogelijk persoonlijke aandacht voor de patiënten nagestreefd.

Instituut Verbeeten werkt nauw samen met zorgpartners in de regio. Hierdoor wordt de juiste kennis en kwaliteit bij elkaar gebracht rond oncologische patiënten en kan een snelle diagnose en optimaal behandeltraject worden gerealiseerd. Het instituut streeft naar een diagnostisch behandeltraject zonder wachttijden.[7]

Het streven naar een optimale omgang met patiënten uit zich ook in de architectuur en inrichting bij het instituut. Er wordt gezorgd voor een prettige omgeving met veel licht, vriendelijk kleurgebruik wat bovendien ook functioneel is toegepast als een soort bewegwijzering in het pand. “Neemt u maar plaats bij de rode stoelen” in plaats van dat een bezoeker verwezen wordt naar het bordje “Oncologie” om daar zelf een wachtplek bij te mogen gaan zoeken.

7 http://www.verbeeten.nl/organisatie

Afbeelding 2: Lichtblauwe zetels in een wachtruimte in Tilburg

Dit streven uit zich nog meer in de nieuwe behandelcentra in Breda en Den Bosch. Aanvankelijk werden alle bezoekers steeds verwacht om vanuit heel Brabant naar Tilburg te komen, wat vaak qua afstand niet alleen vervelend was, maar ook vanwege het feit dat toch veel patiënten of hun partners nog gewoon aan het werk waren. Niet alleen vermindert met deze twee nieuwe vestigingen het instituut het aantal benodigde bezoeken voor de patiënt per behandeltraject, maar ook de afstand die afgelegd moet worden voor de bezoeken.

Afbeelding 3: Bestralingsruimte in Den Bosch

Het Healing Environment[8] idee is hier nog verder doorgevoerd, omdat het nu mogelijk was het hele gebouw hier op toe te passen. Zo hebben de bestralingsruimtes LED-verlichting die door de patiënt zelf ingesteld kan worden tot een prettige kleur en is er qua bouw bewust gekozen voor ronde vormen, hout en is er ook aan het uitzicht gedacht. Inmiddels is de locatie in Breda zelfs

8 http://www.verbeeten.nl/organisatie/overinstituutverbeeten/healingenvironment

genomineerd voor de Hedy d'Anconaprijs voor excellente zorgarchitectuur[9].

2.2 Geschiedenis

Instituut Verbeeten werd in 1953 opgericht als Radiotherapeutisch Instituut Tilburg, door een gelijknamige stichting die zich één jaar eerder vormde. Van de oorspronkelijke locatie aan de Professor Donderstraat, verhuisde de instelling in 1981 naar de Brugstraat, naast het Twee Steden ziekenhuis. Na de verhuizing werd het hernoemd naar de eerste geneesheer-directeur Dr. Bernard G.J.M. Verbeeten als dr. Bernard Verbeeten Instituut. Dit is ook nog de statutaire naam, maar het instituut voert sinds 2009 de handelsnaam: Instituut Verbeeten.[10]

Voor het Gamma Knife Centrum leverde Instituut Verbeeten voorheen expertise en menskracht rond kanker. Een gamma knife behandelt hersenaandoeningen door met straling weefsel weg te snijden met de nauwkeurigheid van een chirurgisch mes. Het centrum is tot stand gekomen door samenwerking met dit ziekenhuis en met steun van het Universitair Medisch Centrum Utrecht. Het was het eerste gamma knife-centrum in Nederland.

9 http://cultuurgids.avro.nl/front/detailkunstuur.html?item=8268295
10 http://nl.wikipedia.org/wiki/Instituut_Verbeeten

3 Opdracht

3.1 Aanleiding voor de afstudeeropdracht

CCure is ontwikkeld binnen Instituut Verbeeten en nadert de onderhoudsfase. Omdat een volledige ontwikkelafdeling in deze fase geen vanzelfsprekendheid meer is, moet Instituut Verbeeten een keuze gaan maken of dit nog in-house gedaan blijft worden, of dat er voor een bepaald scenario waarin outsourcing een rol speelt gekozen wordt. Instituut Verbeeten heeft op dit moment de mogelijkheden en de bijbehorende kosten en risico's van de verschillende mogelijkheden niet in kaart.

Ook is het pakket in de loop van de jaren door verschillende programmeurs ontwikkeld en uitgebreid. Hierbij zijn verschillende ontwikkelmethodes gebruikt en geïntroduceerd, waardoor de staat waarin (bepaalde delen van) de applicatie verkeert niet helemaal duidelijk meer is. Dit zou mogelijk ook gevolgen kunnen hebben voor de beheersbaarheid bij eventuele outsourcing.

3.2 Omschrijving van de afstudeeropdracht

Er moet achterhaald worden wat de kosten en risico's zijn die horen bij diverse outsourcing trajecten en bij niet outsourcen. De mogelijke scenario's moeten eerst bepaald worden, en daarna verder onderzocht. De volgende zaken zijn hierbij belangrijk:

- Wat zijn de kosten (direct en indirect)
- Wat zijn de risico's en bij wie liggen die
- Gevolgen op menselijk vlak, bijvoorbeeld communicatie en tevredenheid van het personeel
- Wat zijn de juridische risico's en mogelijkheden
- Borging en beheersbaarheid van kennis over de applicatie

Bovendien moeten deze resultaten zo uitgelegd worden dat de Raad van Bestuur deze kan gebruiken om een afgewogen beslissing te kunnen nemen. Ook moeten de resultaten gepresenteerd worden aan de Raad van Bestuur.

Het praktijkgedeelte betreft het verbeteren van het ontwikkelproces, de testbaarheid, onderhoudbaarheid en overdraagbaarheid van de applicatie, met in het achterhoofd dat de applicatie eventueel ook geoutsourced zou moeten kunnen worden. Aandachtspunten hierbij zijn de wijze van het documenteren van functionaliteit, geautomatiseerd testen en best practices en coding guidelines.

3.3 Deliverables

- Onderzoek naar de mogelijkheden en gevolgen van outsourcen van CCure
- Presentatie voor de Raad van Bestuur
- Verbeteringen aan het ontwikkelproces en de software om CCure voor te bereiden op de onderhoudsfase en eventuele outsourcing

3.3.1 Onderzoek

De impact van het outsourcen van het ontwikkelen van een softwareproduct is niet alleen te meten in kosten, maar heeft op vele vlakken gevolgen voor een organisatie en kan ook kosten en risico's met zich meebrengen die een organisatie wellicht niet wil dragen. Om dit in kaart te brengen moet er een onderzoek gedaan worden naar aan de ene kant wat er al bekend is rondom de risico's en gevolgen van outsourcing, en aan de andere kant wat de gevolgen (kunnen) zijn voor specifiek deze organisatie.

Daartoe worden er met vijf mensen uit verschillende disciplines of rollen binnen die disciplines interviews gehouden, met daarin vragen die verband houden met hun specifieke rol. Om goed in kaart te kunnen brengen waar de problemen kunnen ontstaan tijdens een outsourcing traject, en in het algemeen een beeld te krijgen van outsourcing, is er een literatuuronderzoek aan vooraf gegaan.

Alle resultaten van de interviews en de algemene informatie rondom outsourcing worden vervolgens samengevat in een onderzoeksresultaat. Dit uitgebreide onderzoeksresultaat bevat alle informatie rondom dit onderzoek, een overzicht van voor- en nadelen per outsourcing model en een uiteindelijk advies gebaseerd op deze informatie.

3.3.2 Presentatie

De onderzoeksresultaten moeten vervolgens gepresenteerd worden aan de Raad van Bestuur van Instituut Verbeeten. Deze presentatie bevat vooral de informatie uit het overzichtsgedeelte van het onderzoek, het advies wat daarop gebaseerd is plus een algemene inleiding over outsourcing. Zodoende is de Raad van Bestuur in staat om een beslissing te kunnen nemen inzake de keuze om te gaan outsourcen of door te blijven werken zoals er nu gewerkt wordt.

3.3.3 Verbeteringen ontwikkelproces

Het praktijkgedeelte rondom dit afstudeerproces concentreert zich rondom verbeteringen in het ontwikkelproces en het klaar maken van de applicatie voor de onderhoudsfase en outsourcing. De technische ontwerpen voor changes en nieuwe functionaliteit kunnen sterk verbeterd worden, in het bijzonder als er rekening mee gehouden moet worden dat deze in de toekomst uitgevoerd zouden moeten kunnen worden door een externe partij. Er moeten een aantal changes en ontwerpen van nieuwe functionaliteit gemaakt worden volgens verbeterde methodes, en vervolgens getoetst worden of deze inderdaad de verwachte verbetering brachten.

Hoewel de testbaarheid van de applicatie in principe goed zou moeten zijn vanwege de gebruikte technologieën, werd er tot nu toe totaal geen gebruik gemaakt van geautomatiseerd testen. Dit houdt in dat de menselijke testers nu verantwoordelijk zijn voor het opsporen van alle fouten in de software. Er moet worden gekeken of zaken als Unit Testing, Integration Testing en de bijbehorende code coverage een verbetering brengen tijdens het ontwikkelen, en of het zin heeft om de reeds bestaande software ook te voorzien van zulke tests.

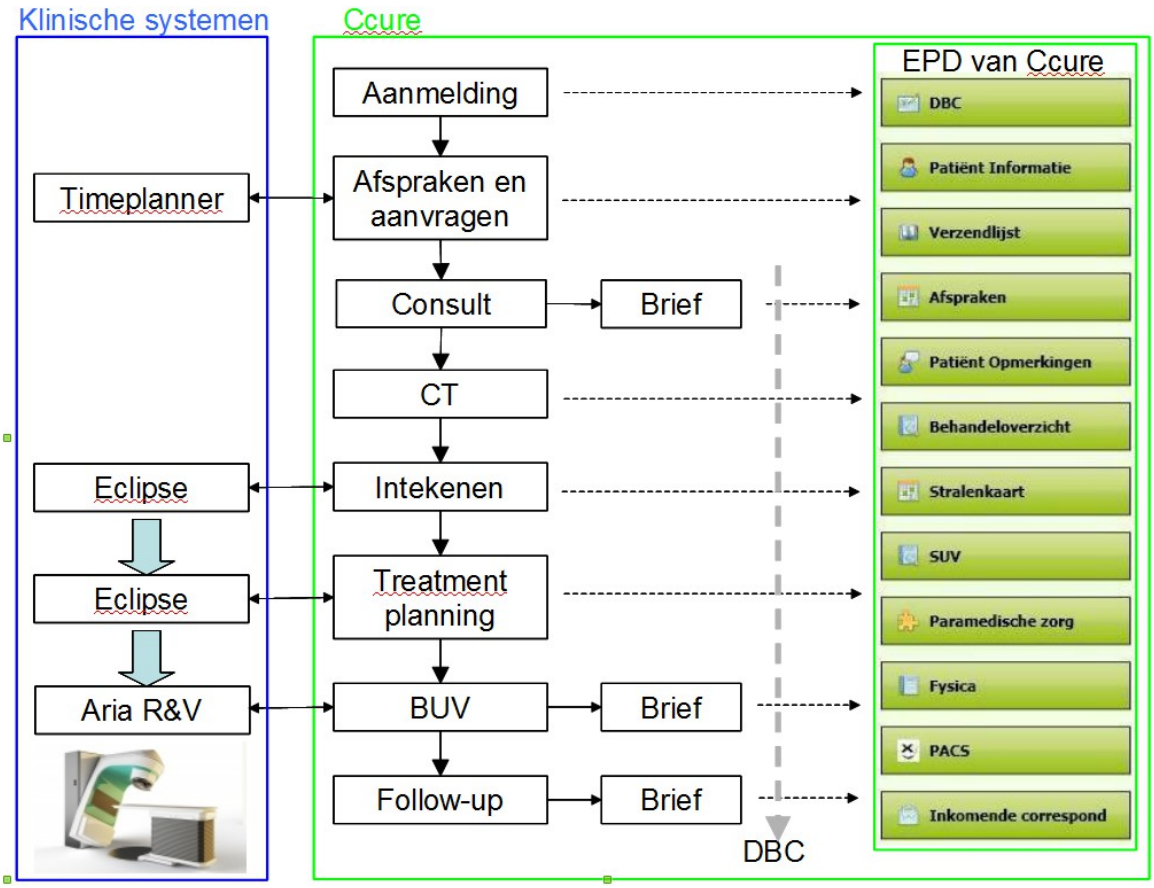
4 Over CCure

De ontwikkelafdeling van Instituut Verbeeten heeft maar één echt softwareproduct dat ontwikkeld wordt, en dat is CCure. Alle overige zelf ontwikkelde software zijn met name tools en koppelingen die de normale ICT activiteiten ondersteunen. CCure is al reeds zes jaar in ontwikkeling en nadert inmiddels de afrondende fase waarna het de onderhoudsfase in gaat. Vanwege de grote hoeveelheid functionaliteit die inmiddels opgeleverd is, is het project (exclusief database en andere files die geen programmacode bevatten) 1 GB groot en bevat miljoenen regels code.

Afbeelding 4: Logo CCure

4.1 Wat doet CCure?

CCure omvat het totale proces van diagnose, behandelingstraject en followup van kankerpatiënten. Bij iedere stap in dit traject wordt er door de medewerkers gewerkt in CCure. Diagnoses door artsen worden ingesproken via de voor hen vertrouwde methode, maar direct electronisch verwerkt in het dossier. Alle gekoppelde systemen, inkomende documenten en brieven rondom dit proces zijn inmiddels ook verwerkt in CCure. In vogelvlucht ziet het proces van CCure er als volgt uit (zie afbeelding 5):



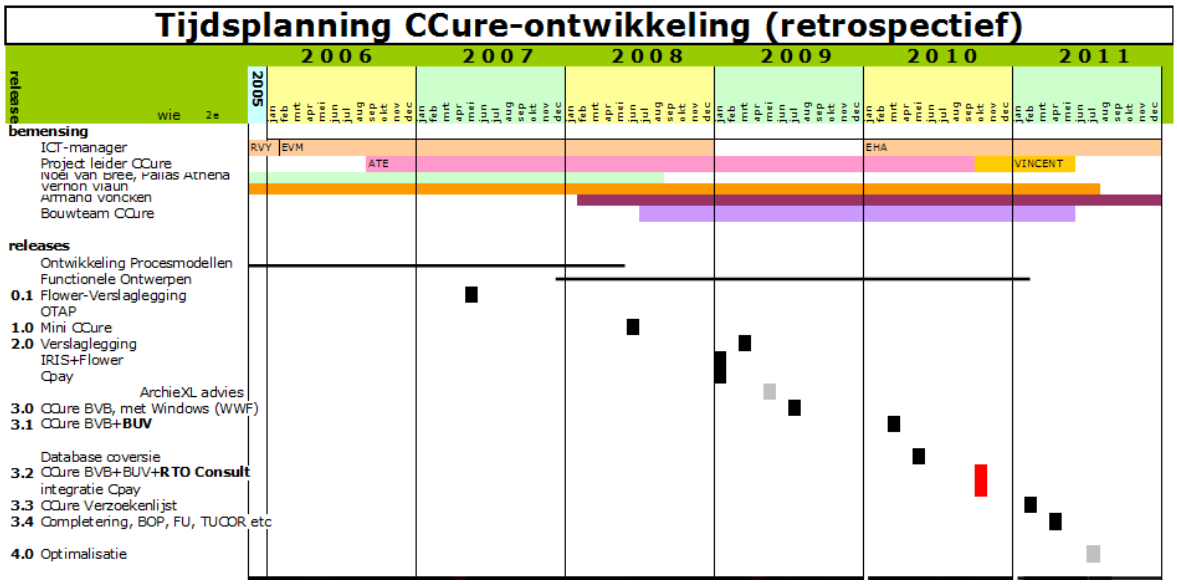
Afbeelding 5: Schematische procesflow van het proces binnen CCure

Termen:

BUV	Bestralingsuitvoering
CT	Computertomografie, CAT-scan
DBC	Declarabele Behandelcombinatie

4.2 Wat is de geschiedenis van CCure?

CCure is een project met een lange historie. Inmiddels is er ruim zes jaar aan gewerkt door meer dan 20 verschillende programmeurs, en zijn er in die tijd ook enkele malen grote wijzigingen in de architectuur en het doel van de applicatie geweest.



Afbeelding 6: Overzicht ontwikkelperiodes afgelopen jaren

Tot 2006 werd er een verzameling van eenvoudige applicaties gebruikt, vaak geschreven in Visual Basic met een Acces database zonder directe samenhang. Echter in dat jaar begon men aan een onderzoek naar de integratie van alle software en het gebruiken van workflows door de afdelingen heen. Het volgende jaar wordt er een eerste poging gedaan, genaamd Flower, wat een Workflowapplicatie is geschreven in Java. Dit komt nog niet zo goed van de grond, maar Flower dient een jaar later wel als Workflowengine voor de eerste echte CCure versie.

Vanaf 2009 wordt Flower vervangen door Windows Workflow Foundation en wordt de applicatie ook voor het eerst echt bedrijfsbreed in gebruik genomen. Ook wordt er een verdere stap genomen in het makkelijker maken van het ontwikkelen voor en testen van een WPF applicatie door het gebruiken van het Catel framework.

Uiteindelijk is in de eerste helft van 2011 het ontwikkelen van CCure volledig afgerond voor alle afdelingen, en begint de onderhoudsfase. Het hele jaar 2012 zal gebruikt worden om CCure klaar te krijgen voor de onderhoudsfase.

4.3 Wat is de toekomst van CCure?

Aangezien het volledige proces binnen Instituut Verbeeten nu verwerkt is in CCure, nadert nu de onderhoudsfase. Het onderzoek naar de mogelijkheid van outsourcing is hier dan ook een uitvloeisel uit, aangezien het echte ontwikkelen inmiddels achter de rug is en er in de organisatie nu ook het gevoel bestaat dat een eigen (dure) ontwikkelafdeling hebben niet per sé meer nodig is.

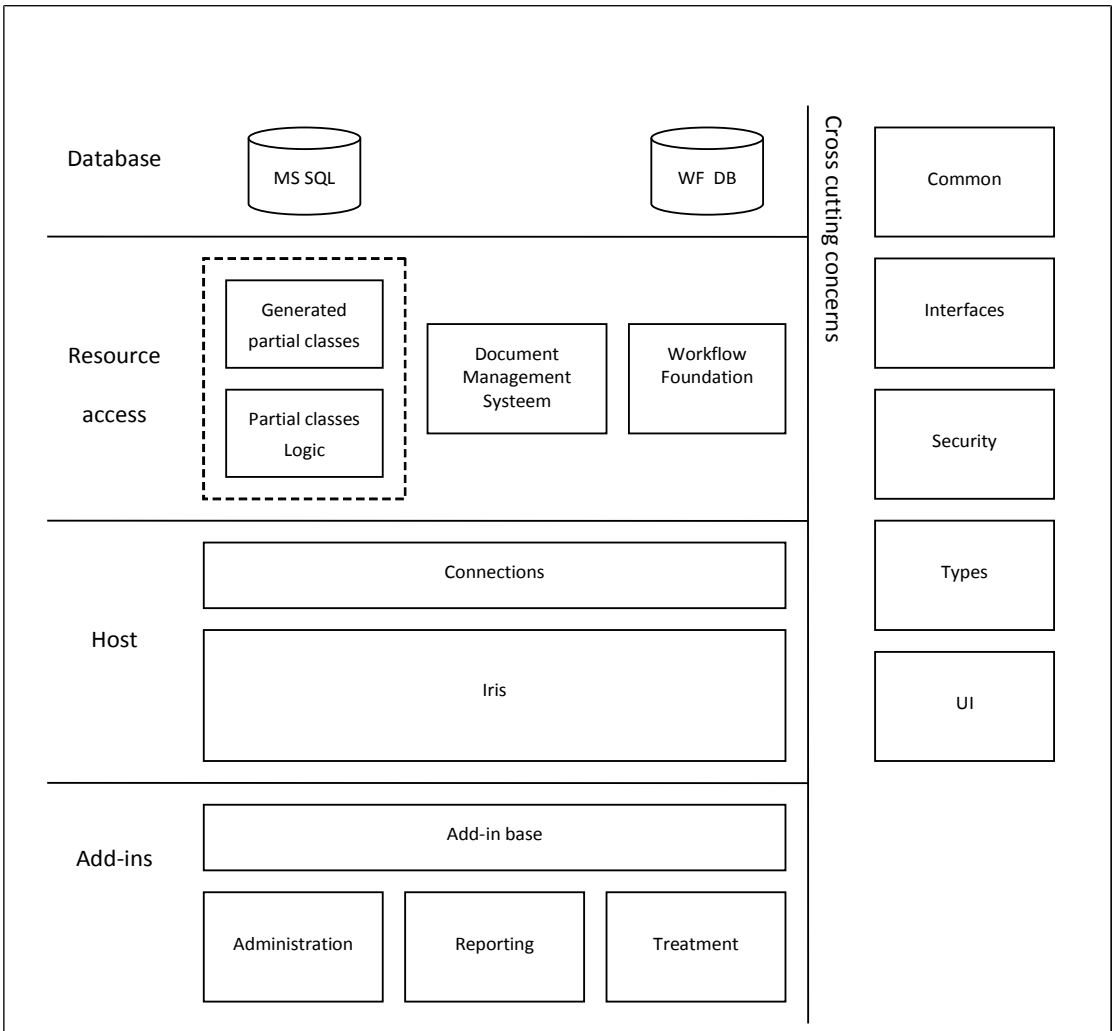
Wel is er interesse om het product te kopen vanuit een partij, en bestaat er de behoefte ook binnen de organisatie om ook (bepaalde delen van) de applicatie te ontsluiten voor Windows 8-tablets, omdat niet alle werkzaamheden direct in de buurt van een Windows-PC plaatsvinden en een tablet een ideaal draagbaar instrument is.

4.4 Welke technieken worden er gebruikt in CCure?

4.4.1 Architectuur

CCure is opgedeeld in sterk gesegregeerde lagen, wat te zien is op afbeelding 7. Er is een persistencylaag waarbij twee databases er voor zorgen dat zowel de gewone data als de Workflowdata opgeslagen wordt, plus alle documenten opgeslagen worden (als binaries in de database).

Vervolgens zit daar bovenop LLBLGen als OR-mapper die Entities en Collections genereert, met deels automatisch gegenereerde code en deels handmatige code, en de aansturing van de documenten en Workflows. Daarna komt de hostlaag, wat de kern van de applicatie is. Hier zitten de verbindingen naar de datalaag en wordt er bepaald welke Workflow geopend wordt, welk scherm getoond wordt en in welke context het programma zit.



Afbeelding 7: Architectuurdiagram

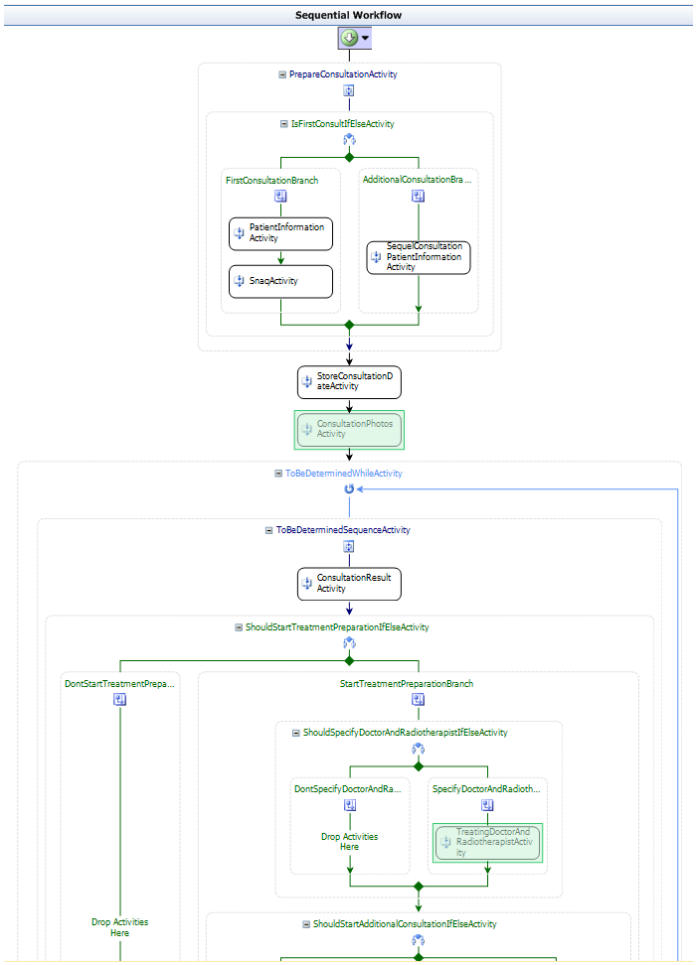
Voor CCure is het mogelijk om add-ins te ontwikkelen die los van elkaar staan. In deze add-ins worden alle schermen en ViewModels geplaatst, samen met de ondersteunende classes specifiek voor die add-in. De volgende add-ins zijn inmiddels gerealiseerd:

- Administration, voor het instellen van de software
- Reporting voor managementinformatie
- Treatment, waar het echte proces zich in bevindt

Verder zijn er nog de zogenaamde cross-cutting concerns, op zichzelf staande libraries die door iedere laag en module gebruikt mogen worden, bijvoorbeeld met standaard UI-elementen of Helpers.

4.4.2 Windows Workflow Foundation

Windows Workflow Foundation[11] (afgekort WF) is een framework van Microsoft dat er op gericht is een proces vast te leggen in een flowchart, waarin de business decisions zitten en waar omheen weer calls naar normale code zit. Zodoende gaat een ontwerp bijna 1:1 van UML flowchart naar een softwarematige implementatie, en blijft ook altijd in de gedachtegang van het ontwerp aanwezig. Ook is de persistentie van Workflow states zeer eenvoudig en kan bijvoorbeeld zonder extra code te schrijven automatisch in de database gebeuren.



Afbeelding 8: Deel van de consultatieworkflow

Vanwege de manier waarop het proces plaatsvindt binnen Verbeeten, was Windows Workflow Foundation een goede manier om dit proces vast te leggen in de software. Iedere patiënt bij Verbeeten doorloopt namelijk altijd min of meer het zelfde proces, met wat kleine variaties mogelijk per traject, maar in grote lijnen altijd het zelfde.

Binnen CCure zijn de Workflows zo ingericht dat deze dermate abstract zijn

11 http://msdn.microsoft.com/library/dd851337.aspx

gebruikt dat het eventueel mogelijk zou zijn om een andere Workflow provider te gebruiken, mocht daar behoefte aan zijn. Echter, aangezien enerzijds WF prima voldoet en anderzijds de extra abstractielaag vrij complex is, leidt dit er juist toe dat programmeurs niet zo snel geneigd zijn Workflows toe te voegen of aan te passen.

4.4.3 Windows Presentation Foundation

Windows Presentation Framework (WPF) is een relatief nieuwe techniek die gebruikt wordt vanaf Windows XP om native Windows software mee van een grafische schil te voorzien. Het voordeel van WPF is dat dit beter aansluit op het .Net-framework, makkelijker is in het gebruik en ook ondersteuning biedt voor 2D en 3D acceleratie door de inmiddels standaard aanwezige 3D-kaarten in iedere computer, en er mooier en moderner uit ziet.

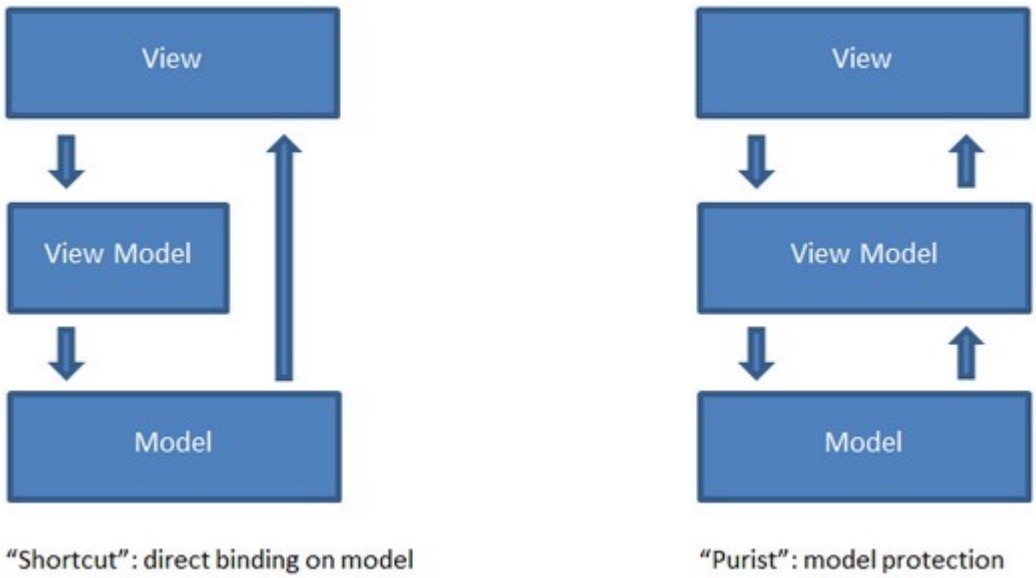
WPF maakt gebruik van een XML-standaard voor het opmaken van schermen, genaamd XAML. Door het gebruik van een zogenaamde binding kan dynamische data weer gekoppeld worden aan het scherm, bijvoorbeeld tekst in een tekstvak of de datum in een datumpicker.

4.4.4 Catel MVVM Framework

Het Catel Framework[12] is een MVVM framework dat geschreven is door Geert van Horrik, een van de programmeurs die ook aan CCure gewerkt heeft. De versie die binnen CCure gebruikt wordt is een oudere versie die niet geheel compatible is met de huidige versie van Catel, en waar aan ook modificaties zijn aangebracht die specifiek voor CCure gedaan zijn. Hierdoor wordt er getwijfeld of er doorgegaan moet worden met het gebruiken (van deze versie) van Catel.

Catel werkt samen met de O/R Mapper LLBLGen, en in het project wordt óf direct een LLBLGen Collection of Entity gebruikt (in MVVM termen het Model) in de code behind van een (eenvoudig) scherm, óf worden meerdere Entities en Collections gebruikt in een ViewModel. Ook is Catel met WPF geïntegreerd zodat deze naadloos aansluit op de bindings die in XAML opgegeven worden.[13]

12 http://catel.codeplex.com/
13 http://catel.catenalogic.com/2.4/index.html?different_interpretations_of_m.htm



Afbeelding 9: Gemengd directe mapping Entities versus puur ViewModels

4.4.5 LLBLGen

LLBLGen[14] is een O/R Mapping (kort ORM) library ontwikkeld voor het Microsoft .Net-platform, waarmee een koppeling bewerkstelligd wordt tussen een database en de software waardoor veel voorkomende databasehandelingen niet meer via handgeschreven code verlopen maar via objecten die grotendeels automatisch gegenereerd worden via een designer, die de structuur van een database inleest en vervolgens omzet naar kant-en-klare code.

Deze code bestaat ruwweg uit Entities en Collections, waarbij Collections kunnen bestaan uit simpelweg de hele inhoud van een tabel, of het resultaat van een query. Queries kunnen in code opgebouwd worden zodat het gebruik van SQL in principe geheel overbodig wordt. Toch is het ook mogelijk om voor complexe queries nog gebruik te maken van bijvoorbeeld Stored Procedures.

Omdat LLBLGen als generieke tussenlaag gebruikt wordt is het in theorie heel eenvoudig om te switchen van databaseengine, er worden door LLBLGen veel SQL(-achtige) databases ondersteund zoals Oracle, PostgreSql, MS Access, IBM DB2, MySQL en SyBase. Echter is het wel zo dat niet iedere database exact de zelfde features heeft, waardoor er alsnog lichte verschillen kunnen ontstaan in werking waardoor er vaak alsnog kleine aanpassingen in de code gedaan moeten worden.

LLBLGen en Catel zijn dermate sterk met elkaar verweven in dit project, dat de code die LLBLGen genereert is aangepast zodat het direct werkt in Catel. Dit bespaart weliswaar tijd tijdens het ontwikkelen voor CCure maar zorgt er wel voor dat het heel lastig wordt om beide frameworks nog apart van elkaar te

14 http://www.llblgen.com/defaultgeneric.aspx

gebruiken.

4.4.6 MediSpeech van G2Speech

Afbeelding 10: Medispeech logo

Er wordt in de medische wereld traditioneel veel gebruik gemaakt van het dicteren van analyses en diagnoses. Medici zijn vaak behoudend qua nieuwe technologieën, en zien daarom ook het liefste dat de technologie zich aan hen aanpast, in plaats van andersom.

Daardoor zijn er verschillende partijen actief op het gebied van spraakherkenning specifiek voor de medische wereld, waarvan MediSpeech van G2Speech er een is. Hoewel spraakherkenning tegenwoordig gemeengoed is en het zelfs al standaard in Windows of een iPhone aanwezig is, is het niet eenvoudig om spraakherkenning voor de medische wereld toe te passen waarbij het niet meer nodig is om handmatig achteraf teksten te corrigeren.[15]

Zo moet er rekening gehouden worden met het feit dat een bepaalde medische term maar op één manier correct geschreven kan worden terwijl dat niet uit de uitspraak te halen is (denk aan spelling of hoofdlettergebruik), en dat bovendien in sommige delen van de medische wereld dat weer net anders ligt bij het zelfde woord. Zodoende zijn er alleen al binnen Verbeeten drie verschillende MediSpeech-libraries voor G2Speech in gebruik.

MediSpeech is via een API geïntegreerd in CCure, en alles wat gedicteerd wordt komt linea recta terecht in de patiëntendossiers in CCure. De techniek is dermate succesvol dat blijkt dat 96-98% van alle invoer zonder handmatige correcties achteraf gebruikt kan worden.

15 http://www.g2speech.com/solutions/medispeech.html

5 Aanpak

5.1 Onderzoeksgedeelte

Het onderzoek naar de mogelijkheden tot outsourcing van CCure is grofweg te verdelen in drie delen, namelijk het allereerste onderzoek, de interviews met de gebruikers die daar op volgen, en vervolgens het maken van een adviesrapportage en de daarbij behorende presentatie aan de Raad van Bestuur van Instituut Verbeeten.

5.1.1 Vooronderzoek

Het vooronderzoek beslaat de voorbereidende fase. Allereerst moest er duidelijk in kaart gebracht worden welke zaken voor Instituut Verbeeten het belangrijkste waren om te onderzoeken. Na overleg met Eelko Hondema zijn de volgende speerpunten gedefinieerd:

- Wat zijn de kosten (direct en indirect)
- Wat zijn de risico's en bij wie liggen die
- Gevolgen op menselijk vlak, bijvoorbeeld communicatie en tevredenheid van het personeel
- Wat zijn de juridische risico's en mogelijkheden
- Borging en beheersbaarheid van kennis over de applicatie

Vervolgens is er literatuur en informatie gezocht die aansloot op de vragen die rezen uit de genoemde speerpunten. Er was veel te vinden over communicatie, kosten en risico's. Wat meer moeite kostte het om de gevolgen op het gebied van personeel te onderzoeken, of juridische problemen. Deze vragen moesten uiteindelijk beantwoord worden in de interviews.

5.1.2 Interviews

Nadat het onderzoek uitgevoerd was, kon in kaart gebracht worden waar de risico's lagen en waar er vragen ontstonden specifiek voor de organisatie. Deze vragen zijn opgesteld, en vervolgens verdeeld over diverse medewerkers binnen Instituut Verbeeten die eenieder hun eigen expertisevlak hebben, namelijk:

Medewerker	Functie
Tiny van de Klooster	Manager P&O
Irene Verkuylen	Secretaresse Raad van Bestuur
Helen Vermeulen	Financial controller
Eelko Hondema	Manager ICT
Bernard van Erp	CCure teamleader
Vernon Vlaun	Procesexpert en acceptatietester

Afbeelding 11: Geïnterviewde medewerkers en functie

Zij konden zowel schriftelijk reageren als de vragen mondeling beantwoorden. Ook zijn er een aantal vragen aan de afstudeerder toegekend, om te onderzoeken in de diverse literatuur over deze onderwerpen.

5.1.3 Adviesrapportage

De antwoorden van de interviews, en de inzichten verworven door het literatuuronderzoek hebben uiteindelijk geleid tot een adviesrapportage. De adviesrapportage is zo ingericht dat voor de vijf mogelijke scenario's (drie outsourcing varianten en twee met mensen op locatie) op een identieke manier behandeld worden:

Een beschrijving, vervolgens de voor- en nadelen op een rijtje, en dan over zeven vaste onderdelen (Communicatie, Waarborging kennis, Waarborging kwaliteit, Testen en acceptatie, Financieel, Juridisch en Menselijk aspect) beoordeeld met de informatie die eerder verzameld is. Vervolgens worden ze vergeleken op deze onderdelen, afgewogen en wordt er een eindconclusie gegeven met daarin ook een advies over de te volgen weg.

Deze adviesrapportage wordt vervolgens in een gecondenseerde vorm gepresenteerd aan de Raad van het Bestuur van het Instituut Verbeeten, waarna zij een uiteindelijk besluit kunnen nemen over de te nemen route.

5.2 Praktijkgedeelte

Dit deel had betrekking op het overdraagbaar maken van de applicatie voor een eventuele outsourcing partij. Hoewel het mogelijk zou zijn dat het uiteindelijke resultaat van het onderzoek zou zijn dat er niet gekozen zou worden voor het een outsourcing oplossing, bestond wel heel erg het gevoel binnen de organisatie dat alle verbeteringen ook als dat niet door zou gaan het ontwikkelproces wel sterk zou vergemakkelijken.

In het afgelopen half jaar zijn bijvoorbeeld vier nieuwe mensen begonnen met werken aan CCure, zij moesten allemaal ingewerkt worden, en zij liepen allemaal tegen de zelfde vragen aan. Ook ontdekten we vaak dat programmeurs

eerst de zelfde fouten maakten bij het uitvoeren van changes en bugfixes die vervolgens daardoor eerst een keer afgekeurd werden door de tester. Ook bleek de applicatiekennis heel dun te zijn, bij afwezigheid van bepaalde personen bleek het soms lastig te zijn voor anderen om nog door te werken.

5.2.1 Inventarisatie

Om te achterhalen welke maatregelen genomen moesten worden, moesten eerst de knelpunten geïdentificeerd worden. Deze kwamen allereerst uit de dagelijkse praktijk, waarbij we bijgehouden hebben tegen welke problemen nieuwe programmeurs aan liepen. Verder kwamen door het onderzoek naar mogelijkheden tot outsourcing ook risico's naar voren waar nu nog minder last van ondervonden werd en overheen gestapt kon worden.

Zo werden veel changes en bugfixes nog met heel veel tekst aangeleverd. Soms was dat voor Nederlanders als lastig te interpreteren, mocht dat van Nederlands naar Engels naar bijvoorbeeld Pools gaan dan gaat er nog meer van de oorspronkelijke intentie verloren. Daarom is er gekozen om wat gedefinieerder en gestructureerdere aanpak van zulke zaken te zoeken.

Hoewel er wel degelijk een sprake is geweest van een gestructureerde inventarisatie en uitvoering, is er wel altijd ruimte gebleven voor het inpassen van nieuwe ideeën en inzichten, zodat er ook nog tijdens de uitvoering nieuwe zaken zijn toegevoegd

5.2.2 Uitvoering

Nadat alle knelpunten in kaart zijn gebracht, is er een lijst van opgesteld is daar per punt een verbetervoorstel voor opgesteld. Nadat deze voorstellen goedgekeurd of licht aangepast waren, ben zijn deze uitgevoerd. Deze mogelijke oplossingen zijn vervolgens weer beoordeeld door alle betrokkenen, en is er een keuze gemaakt of en zo ja op welke manier we de verbetering zijn gaan doorvoeren.

Wat vaak een knelpunt was in de uitvoerende fase was zelden het bepalen wat er verbeterd moest worden en hoe dit te doen, maar er voor te zorgen dat alle betrokken mensen ook daadwerkelijk hier gebruik van gingen maken. Hierdoor zijn nog niet alle voorgestelde verbeteringen in gebruik, en kan het zijn dat ze uiteindelijk nog licht aangepast moeten worden voordat ze voor iedereen tot volle tevredenheid zijn.

Verder bleek het ook zo te zijn dat een aantal zaken die in theorie heel erg veelbelovend waren, in de praktijk enorm tegenvielen, zoals bijvoorbeeld het doorvoeren van Code Coverage, of het gebruik van Microsoft Expression Blend om schermen mee te schetsen. Hierdoor hebben we andere keuzes moeten maken dan initieel voorgesteld was.

6 Onderzoek

6.1 Wat is outsourcing?

"Uitbesteding, soms aangeduid met de Engelse term outsourcing of het vernederlandste outsourcen, is de uitvoering van een proces als gevolg van een strategische keuze door een organisatie, om één of meer bedrijfsactiviteiten uit te besteden aan een dienstverlenende onderneming of toeleverancier." [16]

In deze context gaat het om het outsourcen van het ontwikkelen en beheren van CCure, een softwareproduct. Hiertoe zou een ICT-outsourcing partner gezocht moeten worden die kennis en ervaring heeft met het ontwikkelen en beheren van software, ook op de kennisgebieden die vereist zijn voor het programmeren aan CCure, zoals WPF, LLBLGen en Windows Workflow Foundation.

6.2 Welke vormen van outsourcing bestaan er?

Er wordt tegenwoordig onderscheid gemaakt in drie vormen van outsourcing, namelijk:

- Offshoring
- Nearshoring
- Closeshoring

Alle drie de vormen hebben zo hun specifieke voor- en nadelen. Er is getracht om al deze vormen op een gelijkwaardige manier te vergelijken en daar uit te bepalen welke vorm het beste zou zijn voor het instituut.

6.2.1 Offshoring

Offshoring is het uitbesteden van bedrijfsactiviteiten naar een ander continent of werelddeel. Hiertoe behoort ook Azië voor Europese bedrijven, ondanks dat deze twee werelddelen via land zijn verbonden en het dus eigenlijk geen andere kust is. De meest bekende landen voor offshoring zijn:

- India en Pakistan
- China
- Brazilië
- Zuid-Afrika

Alle landen hebben zo hun specifieke voor- en nadelen. Er zijn altijd behoorlijke verschillen in taal, cultuur, en tijdzone of seizoen, bij de ene partij anders dan bij de andere

16 http://nl.wikipedia.org/wiki/Outsourcing

Voordelen:

- In eerste instantie goedkoper
- Makkelijker krachten te vinden
- Inzet krachten heel flexibel

Risico's:

- Communicatieproblemen
- Geen echte binding met het product
- Overhead doordat er meer moeite in communicatie en management gestoken moet worden
- Verlies van kennis
- Moeten aan het handje gehouden worden wat betreft regels rondom kwaliteit

6.2.2 Nearshoring

Nearshoring is het uitbesteden van bedrijfsactiviteiten naar landen die weliswaar op het zelfde continent liggen, maar toch dermate verschillen qua loonkosten en arbeidsaanbod dat het interessant is om activiteiten daar heen te verplaatsen. Voor Nederlandse bedrijven betreft dat eigenlijk altijd Midden- en Oost-Europese landen uit het voormalige Oostblok. De belangrijkste landen zijn Roemenië, Polen, Tsjechië, Hongarije en Bulgarije. Ook Turkije biedt mogelijkheden aangezien er inmiddels een redelijk aantal geschooldeNederlandse sprekende mensen naar zijn teruggekeerd.[17] [18]

Voordelen

- Zelfde als offshoring
- Face-to-face meetings eenvoudiger
- Minder cultuurverschil dan offshoring

Risico's

- Zelfde als offshoring, maar minder sterk
- Duurder dan offshoring

17 http://www.computable.nl/artikel/ict_topics/outsourcing/3148598/1276946/oosteuropa-een-lonkend-alternatief.html#ixzz0WYxSfVJU
18 http://www.gpic.nl/om-nearshoring.pdf

6.2.3 Closeshoring

Deze variant van outsourcing blijft het dichtste bij huis, namelijk een in redelijke nabijheid gelegen Nederlandse of Vlaamse outsourcing partner die vanuit eigen huis werkt aan de applicatie.

Voordelen:

- Zeer eenvoudig face-to-face te communiceren wanneer nodig
- Geen cultuurverschil
- Echte kennis van en betrokkenheid bij de applicatie
- Hoeven minder strikt gemanaged te worden om de zelfde kwaliteit op te leveren

Nadelen:

- Lastiger te schalen dan off- of nearshoring
- Duurder dan off- en nearshoring en vaste krachten
- Afhankelijkheid van één leverancier

6.3 Welke andere opties zijn er?

6.3.1 Inhuur

Het inhuren van krachten behelst het inhuren van ZZP'ers danwel inhuurkrachten van consultancybedrijven, met als doel er voor te zorgen dat er makkelijker personeel gewerfd kan worden, en dit personeel ook makkelijker weer los te laten is in het geval deze arbeidskrachten niet meer nodig zijn.

Voordelen

- Ingehuurde programmeurs hoeven minder applicatiekennis te hebben aangezien er veel kennis in-house blijft
- Meer controle over de output omdat er dagelijks gemonitord kan worden
- Makkelijker te schalen in capaciteit t.o.v. vaste krachten

Nadelen

- Ongeveer net zo duur als Closeshoring, maar ook kosten werkplek
- Mensen die vertrekken nemen kennis mee

6.3.2 Vaste krachten

Vaste krachten zijn mensen die direct in loondienst zijn bij het Instituut Verbeeten met een vast of tijdelijk contract.

Voordelen:

- Goedkoper dan inhuur of Closeshoring
- Kennis blijft langdurig geborgd binnen het bedrijf
- Intieme kennis van de applicatie

Nadelen:

- Niet flexibel in te zetten
- Lastig te werven
- Lastiger weer kwijt te raken
- Duurder dan Offshoring of Nearshoring

6.4 Hoe werden de scenario's beoordeeld?

6.4.1 Communicatie

Bij communicatie lag de focus vooral op de kosten van de communicatie, de kwaliteit van de communicatie en de risico's tijdens communicatie. Hoeveel wordt de communicatie bemoeilijkt door tijds-, taal- en cultuurverschillen? Hoe ver moet er gegaan worden bij het formaliseren van de contacten? Hoe zorgt het instituut er voor dat een wijziging of nieuwe functionaliteit in één keer wordt

opgeleverd op de manier zoals het instituut het voor ogen heeft? En vooral het voorkomen van geschillen door slechte communicatie was de meest zwaarwegende factor.

6.4.2 Waarborging kennis

In het verleden is er erg veel kennis weggevloeid door vertrekkende teamleden, in technische zin dan wel proceskennis. De (waarschijnlijk terechte) angst bestaat, dat als het gehele ontwikkelproces buiten het instituut gaat plaatsvinden, de organisatie nog gevoeliger gaat worden voor verlies van kennis.

Daarom draait het onderdeel waarborging kennis er dan ook om om niet afhankelijk te worden van de kennis die in bepaalde personen zit, maar om die kennis in die personen naar buiten te brengen en op te slaan. Een belangrijk deel van het onderzoek gaat er daarom om dat de juiste zaken op de juiste manier gedocumenteerd worden.

6.4.3 Waarborging kwaliteit

Omdat een outsourcing partner altijd een ander primair doel heeft (op korte termijn winst maken op softwareontwikkeling) dan Instituut Verbeeten (kwaliteits- en efficiencyverbeteringen om op lange termijn meer winst te maken), moet er voor gewaakt worden dat desondanks het verschil in doel wel het uiteindelijke resultaat van de ontwikkeling van de software, zowel op korte als lange termijn, is wat het instituut er van verwacht.

Daarom ligt de focus hier vooral op het afdwingen van kwaliteit en onderhoudbaarheid van code, en het voorkomen van het ontstaan van lange termijnproblemen.

6.4.4 Testen en acceptatie

Één van de belangrijkste onderdelen van een succesvol software product is het proces van testen en acceptatie. Niet alleen kan hiermee worden voorkomen dat er fouten in de software sluipen die een programmeur zelf niet ontdekt (aangezien een programmeur een hele andere focus heeft dan een tester), ook kan er mee een brug geslagen worden naar de organisatie, waarbij een key user die de acceptatietest doet er voor kan zorgen dat een noodzakelijke wijziging die niet door iedere eindgebruiker geapprecieerd wordt, toch geaccepteerd wordt.

Het proces van testen en acceptatie wordt binnen het instituut op dit moment als zeer goed ervaren, en zouden verbeteringen niet noodzakelijk zijn, hoewel er altijd wensen zullen liggen. Echter als er sprake is van een outsourcing scenario, dan wijzigt de procedure.

Een reactie van een tester of acceptant heeft veel meer gevolgen, of het nou een goedkeuring of juist een afkeuring is. Ook zal de externe partij eerst zelf

moeten testen, maar zal het instituut ook zelf moeten testen. Komen er daardoor extra kosten kijken? Kan een deel van het testen overgebracht worden naar de outsourcing partner?

6.4.5 Financieel

Dit is het belangrijkste onderdeel van het onderzoek, aangezien de aanleiding tot dit onderzoek primair de mogelijkheid tot het besparen van kosten was. Daarom moet er gekeken worden wat de kosten tot nu toe zijn, welke kosten we verwachten per scenario en welke risico's we daarbij lopen.

6.4.6 Juridisch

Het juridische aspect is een zeer lastig aspect, aangezien Instituut Verbeeten zowel met wettelijke bepalingen te maken krijgt als met juridische risico's in contractuele zin. Hoeveel toegang kan en mag een outsourcing partij verleend worden, wie is er verantwoordelijk voor medische fouten die ontstaan door fouten in de software, en hoe zorgt het instituuut dat dit technisch en procedureel goed afgevangen wordt?

6.4.7 Menselijk aspect

Het laatste onderdeel is het menselijke aspect. Wijzigingen in de personeelssamenstelling leveren vaak een hoop onrust op binnen een organisatie en kunnen ook bij de mensen die niet direct getroffen worden toch een negatieve impact hebben. Er moet daarom onderzocht worden welke directe impact, en welke indirect impact een outsourcing traject heeft op het personeelsbestand.

6.5 Conclusies per scenario

6.5.1 Offshoring

Verreweg de goedkoopste oplossing, gekeken naar de kosten per programmeur per uur is offshoring. Ook is er een gigantisch aanbod van geschoolde programmeurs te vinden.

De kosten en risico's die optreden door de communicatieproblemen zijn echter groot te noemen, waardoor de schaalgrootte voor een project dermate groot moet zijn om toch nog te kunnen profiteren van de goedkope lonen dat het onderhouden van CCure op een offshore-locatie niet rendabel is.

6.5.2 Nearshoring

Nearshoring is te vergelijken met offshoring, echter zijn er wel een heleboel communicatie- en cultuurproblemen opgelost doordat er binnen Europa gebleven wordt. Toch heeft CCure niet de schaalgrootte om er genoeg profijt van de goedkopere lonen te kunnen trekken, omdat er naar verhouding te veel begeleiding vanuit Instituut Verbeeten nodig blijft om het in goede banen te leiden. Uitsluitend wanneer er een wat groter en afgesloten project gebaseerd op CCure ontwikkeld zou gaan worden, kan het rendabel zijn om aan Nearshoring te gaan denken.

6.5.3 Closeshoring

Ook voor closeshoring blijft het verhaal over schaalgrootte gelden, echter is het wel zo dat de huidige partners waarmee het instituut samenwerkt ook in staat zijn om programmeurs vanuit hun eigen locaties te laten werken. Hoewel een permanente oplossing met alleen programmeurs op afstand nog steeds tegen een aantal van de bezwaren aan loopt als Nearshoring en Offshoring, is het wel een interessante optie om tijdelijk extra capaciteit te sourcen op het moment dat er niet genoeg personeel in-house is om een groter aan CCure gerelateerd project mee uit te voeren. Zo kan voorkomen worden dat het nodig is om tijdelijk een hele ontwikkelafdeling vol te zetten met mensen, maar werken de extra medewerkers gewoon vanaf hun eigen werkplek

6.5.4 Inhuur

Op dit moment wordt er gewerkt met een flexibele hoeveelheid programmeurs op inhuurbasis, tussen de twee en zes gemiddeld. Hoewel het inhuren van programmeurs de duurste optie is, heeft het wel de meeste flexibiliteit zonder dat het direct enorm ten koste gaat van de betrokkenheid of hogere extra kosten of risico's met zich meebrengt.

Zeker gezien het feit dat CCure inmiddels zo goed als uitontwikkeld is, is het

wellicht het verstandigste om buiten de huidige vaste krachten met inhuurkrachten te blijven werken, en pas op het moment dat er grotere projecten opgetuigd gaan worden te kijken naar Near- of Closeshoring.

6.5.5 Vaste krachten

Werken met ervaren vaste programmeurs als beheerders van de software is erg aantrekkelijk, maar de vraag is of er nog echt genoeg werk ligt voor deze mensen in de verdere toekomst. Ook is het lastig om in deze context genoeg mensen te vinden, of juist kwijt te raken. Daarom is het geen goed idee om nog full-time programmeurs in vaste dienst aan te nemen. Wel is het aan te raden om op het gebied van testen en aansturing mensen te hebben in de organisatie die met name met CCure bezig zijn, zodat de kennis gewaarborgd wordt. Echter pur sang voor ontwikkeling niet.

6.6 Eindadvies na onderzoek outsourcing

Het eindadvies is gebaseerd op de huidige wensen rondom CCure, namelijk de onderhoudsfase die zich vanaf 2013 zal gaan afspelen, waarin op dit moment geen grote wijzigingen of toevoegingen meer gepland zijn.

Gebaseerd op deze kleine hoeveelheid werk vallen Offshoring en Nearshoring, door hun vereiste projectgrootte om winstgevend te zijn allebei af. Zij vereisen beide dermate veel inzet vanaf de zijde van het instituut dat de risico's de verwaarloosbare – theoretische – winst teniet doen.

Vaste krachten zijn te kostbaar, tenzij het gaat om één of twee personen met veel technische kennis gecombineerd met proceskennis rondom CCure, die ook in staat zijn om externe mensen aan te sturen. Het permanent in dienst houden van uitsluitend uitvoerende programmeurs is gezien de hoeveelheid werk niet aan te raden.

Het advies luidt dan ook om minimaal één persoon met een gedegen technische kennis van CCure te borgen binnen de organisatie, één persoon met de proceskennis, er voor te zorgen dat het project overdraagbaar is, en vervolgens eventuele pieken in vraag op te vangen met inhuurkrachten dan wel mensen die op nearshorebasis werken, waarbij inhuur de voorkeur heeft omdat mensen dan met kortere lijnen aangestuurd kunnen worden.

7 Verbeteringen in het ontwikkelproces

7.1 Inleiding

Feature requests, changes of bugs werden tot op heden op dusdanige wijze ingediend, dat deze in principe alleen uit te voeren zijn door mensen met routineuze kennis van zowel de applicatie als het proces binnen Verbeeten. Vaak ontbrak het aan de context voor een wijziging (wat is de wijziging in het proces dat we hier proberen te bewerkstelligen), was het niet altijd duidelijk wat voor gevolgen een bepaalde change had op andere delen van de applicatie, of was het simpelweg niet duidelijk welke delen van een scherm nieuw waren, welke gewijzigd moesten worden en welke konden blijven staan.

Voor een ervaren programmeur maar onbekend met de organisatie en werkwijze is het hier mee onmogelijk zelfstandig werk uit te voeren aan de hand van de beschrijving van het probleem of de vraag, samen met de documentatie van de software. In de meeste gevallen moest er toch weer met een persoon die het proces of de techniek helemaal in zijn hoofd had overlegd worden voordat het compleet duidelijk was wat gemaakt zou moeten worden. Daarom was al snel de conclusie getrokken dat een externe outsourcing partij zeker tegen de zelfde problemen aan zou lopen, en daarmee ofwel vaker niet zouden opleveren wat er eigenlijk gebouwd zou moeten worden, dan wel er veel tijd verloren zou worden met vragen stellen en wachten op de antwoorden.

Voorts bleek het dat, doordat de applicatie al vele jaren onderweg was, sommige delen van de code nog volgens 'oude' principes geschreven waren, en de nieuwere delen vaak weer anders geprogrammeerd waren. Ook was niet altijd iedere programmeur precies op de hoogte hoe bepaalde zaken het beste geregeld konden worden in MVVM, en werden er sluiproutes gebruikt die het erg lastig maken om fouten nog terug te vinden, en bovendien eventueel weer als voorbeeld kunnen dienen voor andere programmeurs die deze werkwijze ook in hun code gaan toepassen.

7.2 Geautomatiseerd testen

Tot nu toe werd er in CCure altijd initieel getest door een menselijke gespecialiseerde hoofdtester, en een aantal hoofdgebruikers met veel applicatie- en domeinkennis die de acceptatietest deden voordat er definitief gereleased werd. Hoewel deze testen ontzettend belangrijk en zelfs onmisbaar zijn voor niet alleen een stabiele maar ook praktisch bruikbare applicatie, leidde het er wel af en toe dat aangezien er alleen werd getest waar aan gewerkt was, dat er regressiebugs in andere delen van de code slopen.

Door geautomatiseerde tests te schrijven voor code wordt er voor gezorgd dat allereerst een testhandeling niet handmatig steeds herhaald hoeft te worden, maar dat er met een vaste set aan data steeds de zelfde zaken getest blijven

worden, en dat is in de praktijk sneller en completer dan door steeds de GUI op te starten en het handmatig te proberen. Een bijkomstig en nog veel belangrijker voordeel is dat code ook getest wordt op het moment dat de focus helemaal niet meer op dat onderdeel ligt. Hierdoor komen regressiebugs sneller aan het licht.

Wel is het zo dat het schrijven dan geautomatiseerde tests een hoop tijd kost, en dat het ook goed moet gebeuren om alle edge cases af te vangen. Een vaak gehoorde klacht is dat er een paar uur tests geschreven moet worden om een uur code te kunnen maken. Daarom moeten geautomatiseerde tests niet klakkeloos ingezet worden maar moet er altijd nagedacht worden over hoeveel nut het nu en later heeft om ze te bouwen. Helaas blijkt het in de praktijk toch altijd neer te komen op óf altijd, óf helemaal nooit geautomatiseerde tests bouwen.

7.2.1 Unit Testing

Unit Testing[19] is het geautomatiseerd testen van programmacode op het laagst mogelijke niveau, van de individuele functie. Dat houdt ook in dat de te testen functie maar één duidelijke taak heeft en niet drie dingen tegelijk doet. Code die niet aan die eis voldoet zal eerst genormaliseerd moeten worden, of (zoals bijvoorbeeld bij een initialisatiefunctie) pas bij Integration Testing aan bod moeten komen.

Er moeten per functie zowel tests geschreven worden die binnen de verwachte parameters vallen, als aanroepen met onverwachte parameters of parameters die een foutmelding zouden moeten opleveren. Deze laatste categorie Unit Test is dan geslaagd op het moment dat de juiste uitzondering naar boven komt.

Een tool die behulpzaam is bij het vinden van edge cases, of gaten in het afdwingen hier van is Pex[20]. Deze tool gaat net zo lang een bepaalde functie uitvoeren totdat alle branches binnen deze functie geraakt worden. Pex kijkt hier bij op een intelligente manier naar de geschreven code, en probeert net zo lang verschillende parameters uit totdat alle code geraakt wordt, of geeft een melding dat er wellicht onbereikbare code in de functie zit.

Het nadeel van Pex is dat ondanks dat het wel degelijk goed doorheeft wat er technisch gezien gebeurt in een functie, het niet herkent wat het normale gebruik van een functie is, zodat bepaalde standaardwaardes van parameters juist niet zo vaak getest worden. Wel komt hier door juist vaak zwakheden tegen die met handbepaalde edge cases niet gevonden worden. Daarom moet Pex altijd in combinatie met handgeschreven Unit Tests gebruikt worden.

In Appendix B op bladzijde 58 staat een voorbeeld van zowel handgeschreven Unit Tests als door Pex automatisch gegenereerde tests, waarbij goed te zien is waarom beide manieren van het genereren van testen hun eigen voordelen

19 Unit Testing op MSDN
20 http://research.microsoft.com/en-us/projects/pex/

hebben.

Unit Tests zijn tot voor kort niet gebruikt in CCure. Inmiddels wordt er wel gebruik van gemaakt, daar waar nog echt nieuwe stukken code geschreven worden. Echter door de enorme grootte van de applicatie zullen ze niet meer met terugwerkende kracht geschreven worden.

7.2.2 Integration Testing

Integration Tests[21] zijn tests die volgen op de Unit Tests. Waarbij een Unit Test één afgesloten functie test, test een Integration Test een functie die weer andere subroutines aanroept, een functie die over meerdere lagen gaat (bijvoorbeeld een database call), of een serie van functies die normaliter achter elkaar zouden plaatsvinden, bijvoorbeeld het openen van een verbinding, het ophalen van gegevens, en het vervolgens weer sluiten van een verbinding.

In Appendix B staat ook een voorbeeld van een set integration tests voor het testen van een functie die controleert of de ingegeven reeks tekens volgens de daartoe opgestelde RFC's een geldig emailadres is. Omdat de functie niet uit kleine stukken afgebakende functionaliteit bestaat, en ook weer andere functies aanroept, spreken we nu van een integration test.

Een goede kandidaat in CCure zou het testen van WorkFlows zijn, aangezien deze volgens een bepaalde flow zouden moeten werken bij een bepaalde input op een heel voorspelbare manier. Integration Tests worden in CCure net zoals Unit Tests echter pas sinds kort en op kleine schaal gebruikt.

7.2.3 Code coverage

De term Code coverage wordt gebruikt voor hoeveel van de code geraakt wordt door automatische tests. De code coverage kan aan het einde van een testrun van alle tests bepaald worden, omdat Visual Studio (of een plug-in die de code coverage test) dan kan zien waar de tests wel en niet komen in de code.

Stel dat er op een gegeven moment in de code een gedeelte afhandeling zit voor het ophalen van een lege collectie uit de database, en de Integration Test die dat deel van de code test is alleen geschreven met een gevulde collectie, dan kun aan de hand van de code coverage gezien worden dat die code nog niet getest wordt.

In een ideale situatie zou altijd 100% van de code ook daadwerkelijk getest moeten worden, en er zijn ook projecten waar dat in de praktijk ook echt gebeurt. Echter is het in CCure zo dat de applicatie niet vanaf het begin met automatische testen is opgezet, en er ook bepaalde libraries (zoals Catel) als source code in het project zijn opgenomen, terwijl dit eigenlijk wel gewoon net zo “afgesloten” code is als een geïntegreerde dll. Ook met automatische test tools als Pex blijft het toch altijd zaak om met de hand vervolgens duiding te geven aan de tests en de resultaten, bijvoorbeeld om te begrijpen dat een

21 Integration Testing op MSDN

bepaalde Exception juist wel of geen gewenst resultaat is.

Daarom is het zeker niet aan te raden om voor een 100% code coverage te gaan. Wel kan code coverage gebruikt worden als één van de eisen die gesteld worden aan een eventuele outsourcing partner voor een volgende module voor CCure.

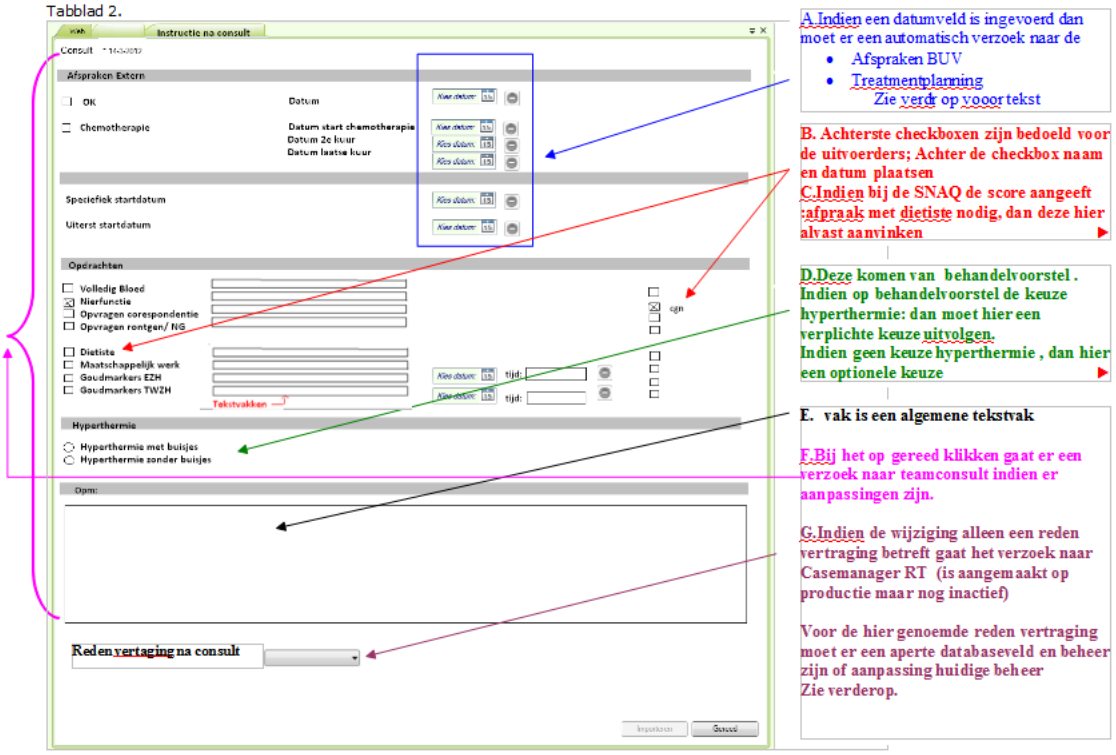
7.2.4 Geautomatiseerd testen in CCure

In CCure is tot voor kort nooit gebruik gemaakt van geautomatiseerde tests, laat staan code coverage. Daarom is het af te raden, om het nu nog heel breed in te zetten, omdat de winst die er bij behaald kan worden nihil is. Wel is het een goed idee om nieuwe functionaliteit, zeker waar het nieuwe afgesloten stukken code zijn (dus geen aanpassingen op bestaande Workflows of ViewModels), wel te voorzien van geautomatiseerde testen.

7.3 Documenteren van functionaliteit

7.3.1 Bestaande documentatie

De bestaande manier van documenteren van grote changes en nieuwe functionaliteit en was langzaam tot een bepaalde vorm geëvolueerd zonder dat daar ooit echt een keer heel kritisch naar gekeken was. Er werd een scherm aangepast of gemaakt door te knippen en te plakken in eenvoudige applicaties als Word en Paint (zie Afbeelding 12), voorzien van enig commentaar en er werd aangegeven welke databasetabellen er opgehaald of opgeslagen moest worden.



Afbeelding 12: Voorbeeld van een schermontwerp volgens de oude methode

7.3.2 Schermontwerpen

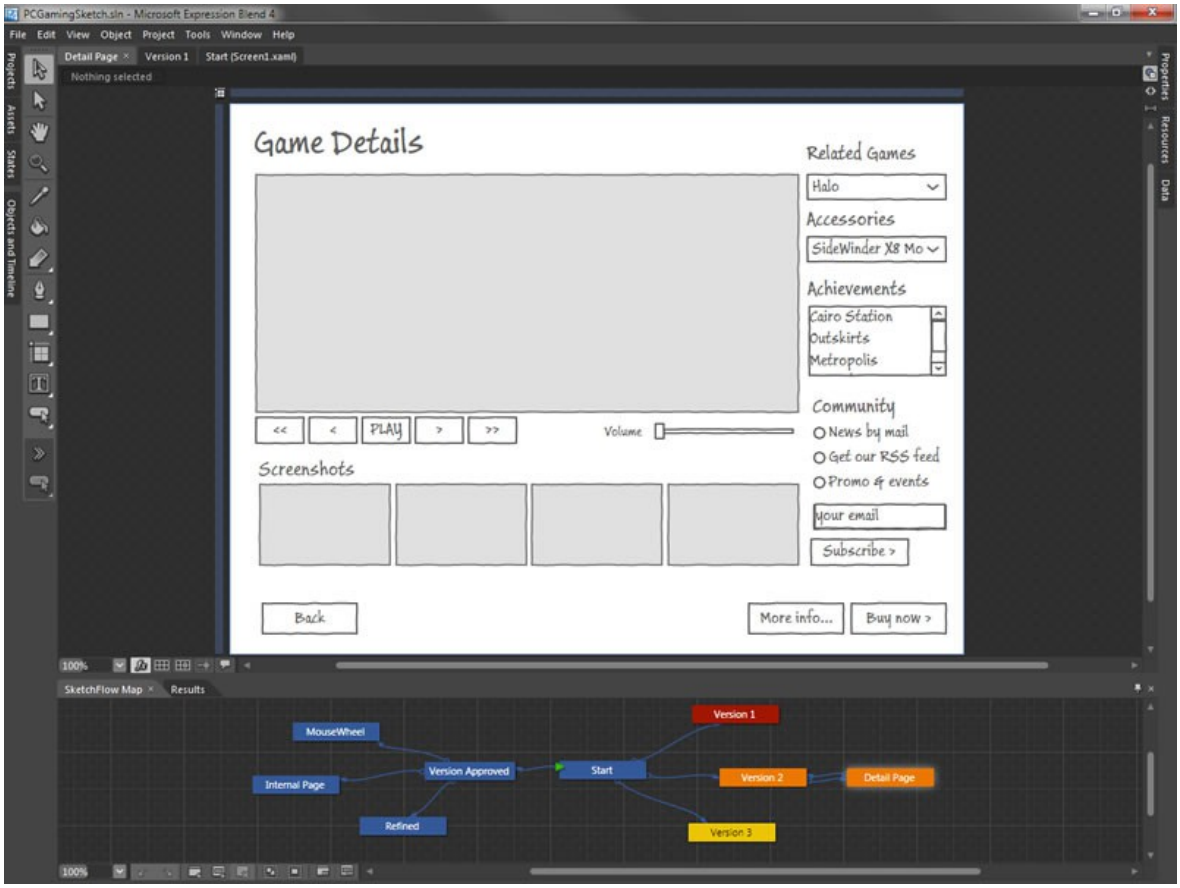
De huidige schermontwerpen zijn screenshots van bestaande schermen welke met fotobewerkingsprogramma's worden bewerkt en in Word worden voorzien van tekst en duiding. Dit maakt het lastig om te bepalen wat er nieuw is in een gewijzigd scherm, en lijkt af en toe gedetailleerder te zijn dan bedoeld door het realisme van de interface. In de software industrie zijn mockup-tools tegenwoordig de standaard voor het functioneel ontwerpen van schermen. Mockup-tools hebben als voordeel dat er niet op detailvormgeving van een scherm ingegaan wordt, maar eerder op functioneel en op interactieniveau.

Er is onderzoek verricht naar diverse pakketten die het mogelijk maken om schermen te ontwerpen als mockups. Hier zaten pakketten in die de mogelijkheid hadden om te integreren met andere softwareontwerptools of zelfs direct code te genereren, en anderzijds pakketten die alleen maar het ontwerp van het scherm konden doen.

Hier onder een vergelijking van alle pakketten:

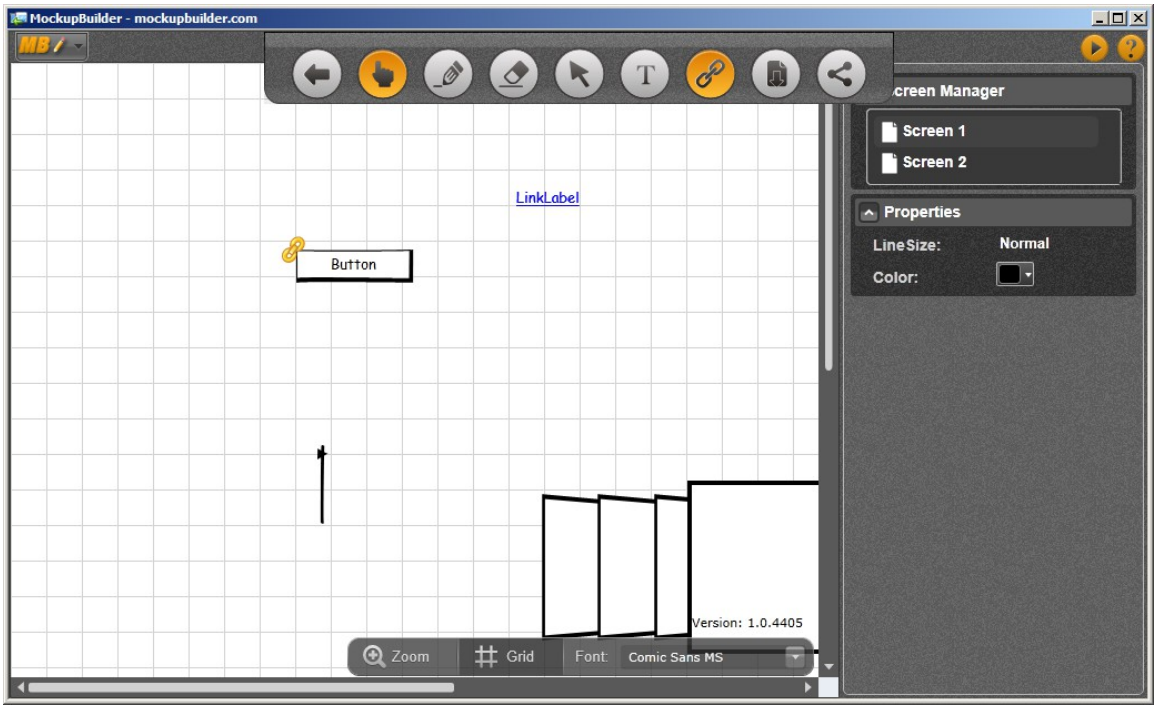
Applicatie	Prijs	Schermweergavesoorten	Linken tussen schermen	Interactieve schermen
Balsamiq	$79,- per user per versie	Schets	Ja	Nee
Visio Professional	+/- €550,-	Wireframe, Realistisch, Schets	Ja	Ja
Expression 4 Ultimate (Sketchflow)	+/- $599,-	Schets	Ja	Ja
Pencil	Gratis	Schets, diverse OS widgets	Ja	Nee
Mockup Builder	Gratis	Schets	Ja	Nee

Uiteindelijk na een schifting uit vijf pakketten is er besloten om met twee pakketten verder te gaan totdat blijkt welke het beste werkt voor alle betrokkenen. De eerste is Mockup Builder en de andere is Microsoft Expression Studio. Expression Studio is een programma wat specifiek bedoeld is om interfaces voor XAML-based technologieën zoals SilverLight en WPF te ontwerpen, en kan ook in een soort mockup-mode werken.



Afbeelding 13: Expression Studio 4 Ultimate

Mockup Builder is een programma wat alleen maar schermen kan ontwerpen en eenvoudig schermen aan elkaar kan linken. Het is echter eenvoudig om te leren, gratis en werkt heel snel. Expression Blend kan kant-en-klare XAML genereren en zou in theorie een stap kunnen overslaan. Echter is het niet gratis, heeft het een hoge leercurve en is op het moment van schrijven nog niet uitgezocht of het wel compatible is met de opzet van CCure. Mocht Blend zijn belofte niet inlossen als mockup tool die direct naar XAML output, dan wordt Mockup Builder het.



Afbeelding 14: Mockup builder

7.3.3 Use case scenario's

Een van de problemen waar de meeste programmeurs aan CCure tegen aan liepen was het ontbreken van een echt verhaal rondom changes en bugs. Wat probeert de gebruiker hier te doen? Wat gaat er aan vooraf?

Met een use case scenario wordt alle functionaliteit vastgelegd in afgebakende use cases. Bijvoorbeeld inloggen is een use case, waarbij het eindresultaat normaliter is dat de gebruiker ingelogd is, maar er ook een fout kan optreden als de inloggende gebruiker geen geldige inloggegevens invoert.

Ook wordt er aangegeven welke use case scenario's er vooraf moeten gaan, bijvoorbeeld is de use case scenario “zoeken” niet uitvoeren, voordat de use case “inloggen” is uitgevoerd. Ook moet er aangegeven worden, mits er sprake is van verschillende rollen, welke rollen of gebruikers de use case wel of niet kunnen uitvoeren.

Verder kunnen prerequisites zijn, bijvoorbeeld dat de gebruiker een werkende internetverbinding moet hebben of een username en password, en wordt aangegeven wat het eindresultaat is van de use case, bijvoorbeeld dat de gebruiker ingelogd is in het systeem, de start van de sessie opgeslagen wordt en de gebruiker wordt doorgeleid naar het startscherm van de applicatie. Op de volgende bladzijdes volgt een voorbeeld van een use case rondom inloggen.

Use case nummer	UC001
Naam	**Inloggen op de Applicatie**
Beschrijving	Vanuit een opgestart programma gaan de gebruikers
Actoren	Alle geregistreerde gebruikers
Precondities	- Applicatie is opgestart - Gebruiker is nog niet ingelogd - Inlogscherm wordt getoond - Gebruiker heeft een gebruikersnaam en wachtwoord - De gebruiker heeft nog geen onthouden inlogsessie voor deze computer
Triggers	Opstarten van de applicatie

Tabel 1: Use case inloggen, overzicht

Deze eerste tabel geeft een algemene uitleg over deze use case, namelijk wat de voorwaarden zijn voordat deze opgestart kan worden. Ook wordt hier aangegeven dat de gebruiker een wachtwoord moet hebben, iets wat niet door de applicatie afgedwongen kan worden, maar gewoon aangeleverd moet worden door bijvoorbeeld de programmeur.

Flow	UC001-01
Type	Normal flow
Naam	**Login met onthouden password**
Stappen	
1) Gebruiker voert gebruikersnaam en wachtwoord in, vinkt “onthoud wachtwoord” aan en klikt op “Verzenden” 2) De applicatie verzendt een verzoek naar de validatiewebservice 3) De validatiewebservice retourneert een antwoordbericht met de mededeling dat er succesvol is ingelogd 4) De applicatie toont het standaarscherm na inloggen	
Postcondities	
1) De gebruiker heeft een geldige sessie voor de applicatie 2) De gebruiker hoeft vanaf deze computer de volgende keer niet meer in te loggen 3) De validatiewebservice heeft een nieuwe sessie gelogd	

Tabel 2: Normal flow 1, inloggen met onthouden van wachtwoord

Dit is de eerste zogenaamde normal flow, een use case die verloopt zoals de software bedoeld is. Er wordt beschreven welke stappen er genomen worden door het systeem of de systemen en de gebruiker, en wat het uiteindelijke resultaat daarvan is.

Flow	UC001-02
Type	Normal flow
Naam	**Login zonder onthouden password**
Stappen	
1) Gebruiker voert gebruikersnaam en wachtwoord in, vinkt “onthoud wachtwoord” aan en klikt op “Verzenden” 2) De applicatie verzendt een verzoek naar de validatiewebservice 3) De validatiewebservice retourneert een antwoordbericht met de mededeling dat er succesvol is ingelogd 4) De applicatie toont het standaarscherm na inloggen	
Postcondities	
- De gebruiker heeft een geldige sessie voor de applicatie - De validatiewebservice heeft een nieuwe sessie gelogd	

Tabel 3: Normal flow 2, inloggen zonder onthouden password

Soms kan het ook interessant zijn om meer dan één normal flows op te nemen, omdat het om verschillende varianten van het zelfde gaat.

Flow	UC001-03
Type	Alternative flow
Wijkt af vanaf	UC001-1 en UC001-2 stap 3)
Naam	**Foutieve username en password**
Stappen	
3) De validatiewebservice retourneert een antwoordbericht met de mededeling gebruikersnaam en/of wachtwoord fout is 4) De applicatie toont een pop-up foutmelding in beeld	
Postcondities	
1) De validatiewebservice heeft een foutieve inlog gelogd voor de ingevoerde username en het IP van de computer	

Tabel 4: Alternative flow, foutieve login

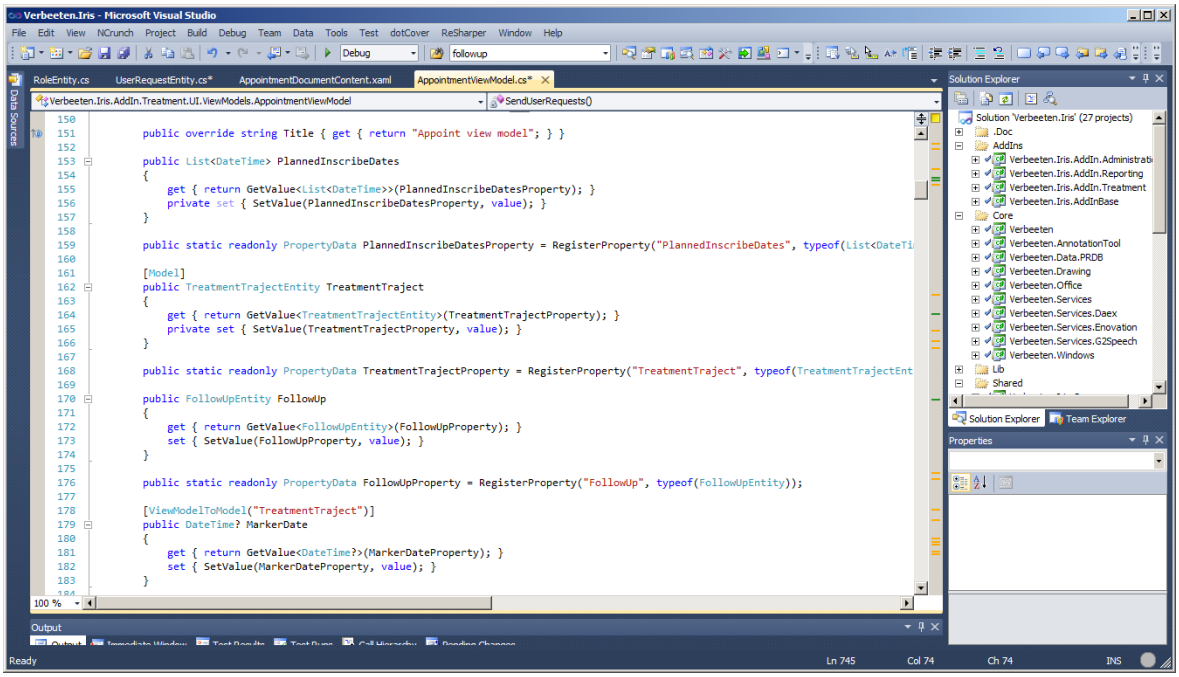
De alternative flow is een flow waarin beschreven wordt wat er gebeurt als er bij een bepaalde stap uit een of meerdere workflows iets fout gaat. Er wordt dan doorgeteld vanaf de stap waarin de afwijking begint, beschreven wat er gebeurt en vervolgens wat daar het eindresultaat van is.
Het uiteindelijke doel hier van is dat een tester kan aangeven precies in welke stap van welke flow het resultaat dan wel de flow afwijkt vanaf de specificaties. Ook is het makkelijker om de prerequisites mee te geven om de fout te herhalen, in dit geval zou dat bijvoorbeeld een bepaalde username / password combinatie zijn waarmee het fout gaat.

7.4 Best practices vastleggen

Er waren binnen het bedrijf al heel wat best practices vastgelegd, bijvoorbeeld rondom testen of databasestructuur, echter blijken er geen duidelijke richtlijn te zijn hoe nieuwe functionaliteit te implementeren in code. Aangezien er een

behoorlijke tijd gewerkt is aan het project, en inzichten en programmeurs zich nogal eens gewisseld hebben, is nu niet meer uit de bestaande code te herleiden wat de beste manier is om een scherm op te zetten. Om de kwaliteit en onderhoudbaarheid van de applicatie te verbeteren, moeten schermen echter wel op de best mogelijke manier in elkaar gezet worden en ook allemaal op een consistente manier.

We hebben daartoe de meest voorkomende schermtypes en code bekeken en bepaald wat de juiste aanpak is per scherm of onderdeel, en daar gedocumenteerde voorbeeldcode voor gemaakt. Zodoende kan een programmeur die een bepaald scherm wil maken of aanpassen hier uit putten om te bepalen welke werkwijze de standaard werkwijze is.



Afbeelding 15: Voorbeeld van broncode van CCure

7.4.1 Stamtabelbeheerscherm

Een stamtabel is een ondersteunende tabel met daar in stamgegevens, zoals tumorkenmerken, soorten verwijzingen of gebruikersgroepen. Deze tabellen worden uitsluitend bijgehouden door gebruikers me beheerderskenmerken. Qua tabelstructuur zijn deze allemaal in de basis het zelfde, en daardoor de schermen ook. Altijd is er sprake van een overzichtscherm, en een detailscherm waarin de entiteit aangemaakt of bewerkt kan worden.

Waar we vaker tegen aan lopen is dat er verschillende standaarden zijn voor het opzetten van zulke beheerschermen, die ook allemaal net weer verschillend werken. Ook moeten deze schermen altijd non-destructive delete zijn, terwijl LLBLGen standaard gewoon de hele Entity verwijdert. Daarom is er nu een

standaard aanpak bepaald, met een code voorbeeld waarin alle standaard aanroepen en knoppen al verwerkt zitten.

Overigens is het zo dat een stamtabelbeheerscherm niet voldoet aan een MVVM-implementatie, omdat deze schermen geen ViewModel gebruiken tussen de Entities en de schermen. Echter vanwege de simpele opzet van de meeste van deze schermen leek het nooit lonend te zijn om deze allemaal te voorzien van een ViewModel als tussenlaag.

De volgende zaken komen minimaal voor in een stamtabel (waar StamTabel een generieke naam is voor een stamtabel), en zitten dus ook al in de template:

Kolomnaam	**Datatype**	**Omschrijving**
StamTabelID	INTEGER	Primary key, auto increment
StamTabelIDAsText	VARCHAR	Unieke string waarmee deze regel gevonden kan worden in de database, deze gebruiken we omdat deze niet automatisch telt zoals de primary key, zodat we over de verschillende databases van de OTAP omgeving geen last hebben van wijzigende ID's. Wordt altijd geschreven als NAAM_VAN_ITEM. Mag alleen tijdens creatie worden aangepast.
StamTabelNaam	NVARCHAR	Naam zoals die zichtbaar wordt voor de gebruikers in de applicatie
StamTabelCreatedBy	VARCHAR	Drieletterige windows-username aanmaker
StamTabelCreatedDate	DATETIME	
StamTabelChangedBy	VARCHAR	Drieletterige windows-username laatste wijziger
StamTabelChangedDate	DATETIME	
StamTabelIsDeleted	BIT	Wordt niet meer getoond op het moment dat deze op TRUE staat
StamTabelDeletedBy	VARCHAR	Drieletterige windows-username verwijderaar
StamTabelDeletedDate	DATETIME	

7.4.2 Scherm met ViewModel

Bij het correct toepassen van het MVVM-model hoort altijd een ViewModel. Een ViewModel vangt zowel af wat er in de code-behind van een scherm gebeurde, als een deel van de validaties en functionaliteit die normaal in de Entities gebeuren. Vuistregel hier bij is dat als het voor iedere instantie van een Entiteit geldt, dat het dan in de Entiteit geregeld wordt, en als het gerelateerd is aan het scherm wat geopend is, het in het ViewModel thuis hoort.

De reden dat er met een ViewModel gewerkt wordt is vanwege de mogelijkheid om verschillend te valideren per scherm, en om er voor te zorgen dat er testbare code onstaat voor de logica van een scherm, dat is bij een code-behind niet mogelijk. Bij een ViewModel is er geen terugkoppeling naar de View meer, wat aan de ene kant omslachtig is, maar het wel een stuk porteerbaarder maakt.

We hebben een bepaald scherm uitgezocht waarin naar ons idee MVVM perfect werd toegepast, en dit ViewModel opgeschoond en beter becommentarieerd zodat het alleen nog maar demo-code bevatte.

Zaken die aan de order zijn gekomen zijn de bindable parameters, omgaan en het koppelen van Events, validatie en het tonen van fouten, initialisatie en opslaan.

7.4.3 Workflow

Een WorkFlow is een van de meest complexe onderdeel van applicatie, en voor veel programmeurs ook een redelijk onbekende manier van werken. Daarom merkten we ook dat sommige dingen die wel degelijk in een WorkFlow thuis hoorden, daar toch niet in geplaatst werden vanwege de angst “iets” om te gooien de WorkFlow waardoor het hele proces zou vastlopen.

Deels is dat ook te verklaren door de op het eerst wat nodeloos abstracte en dubbele manier waarop de WorkFlows geïmplementeerd zijn.

7.4.4 ValueConverter

Een ValueConverter is een WPF specifieke class, die gebruikt wordt om bindings van gegevens te vertalen van en naar een bepaald formaat. Zo is er bijvoorbeeld een converter voor het omzetten van een booleanse-waarde (ja of nee) naar de Visibility property van een control, welke zichtbaar, ingeklapt of verborgen kan zijn. Met een BooleanToCollapsedVisibility-convertor kan een boolean-waarde uit een Entity of ViewModel vertaald worden naar het wel of niet tonen van een bepaalde Control.

Hoewel deze classes zeer eenvoudig zijn, was het toch nodig om een voorbeeld te geven van robuuste checks op de van en naar waardes, en bovendien is er ook een naamgeving standaard voor deze controls opgesteld.

7.4.5 Helper

Een Helper class is een class die gebruikt wordt voor bepaalde standaard functionaliteit, bijvoorbeeld het verzenden van een brief, printen of het uploaden van een document, maar niet direct gerelateerd is aan een Entity of Workflow. Deze Helper-classes zijn meestal ook bedoeld om op meerdere plaatsen gebruikt te kunnen worden en daarom wat generieker.

7.5 Coding Guidelines

Coding Guidelines zijn samen met de best practices bedoeld om er voor te zorgen dat de programmeerstijl van programmeurs meer op elkaar gaat lijken, zodat alle code er consistent uit ziet en makkelijker te onderhouden is. Zeker bij een programma wat geoutsourced gaat worden, is het van belang dat problemen of verzoeken niet alleen opgelost worden, maar dat de code die opgeleverd wordt uiteindelijk onderhoudbaar, consistent en leesbaar is. Met Coding Guidelines heeft het instituut de mogelijkheid om na een code review te kunnen zeggen dat de code niet aan de eisen voldoet, ondanks dat hetgeen wat opgeleverd is in principe wel gewoon doet wat het moet doen.

De Coding Guidelines waar Instituut Verbeeten zich op is gaan baseren zijn de Coding Guidelines die gepubliceerd zijn door Dennis Doomen[22], welke ook voorzien zijn van FXCop[23] en ReSharper[24] code style files om actief deze guidelines af te dwingen via Visual Studio. Op dit moment is er binnen de afdeling nog een discussie gaande over welke guidelines wel en niet dwingend opgenomen moeten worden (die dus worden gemarkeerd als fout waardoor er niet meer gecompileerd kan worden) in ReSharper, aangezien de regels in sommige gevallen erg strikt zijn, en ook problemen zouden toegeven als ze op de huidige codebase toegepast zouden worden.

Daartoe zijn er twee meetings georganiseerd, een waarin een korte demonstratie is gegeven van de guidelines en de combinatie met ReSharper, waarna stap voor stap de guidelines doorgenomen zijn en er over gediscussieerd is. Vervolgens is men met een aangepaste ruleset proberen te gaan werken, aanpassend en noterend waar het in de praktijk niet bleek te werken. Deze resultaten worden vervolgens nogmaals met het team besproken en wordt er een definitieve set regels voor ReSharper gemaakt. Daarna kan er met deze set regels voor ReSharper bij een iedere partij of programmeur die aan het project gaat werken afgedwongen worden dat ze volgens deze Coding Guidelines werken.

22 C# coding guidelines
23 FXCop op MSDN
24 http://www.jetbrains.com/resharper/

8 Conclusies en aanbevelingen

Na het onderzoeken van alle opties voor outsourcing blijkt dat outsourcing naar goedkopere landen een beetje mosterd na de maaltijd is wat betreft CCure. De initiële aanloopkosten en -tijd zijn hoog bij zo'n outsourcing traject, terwijl de benodigde capaciteit aan programmeurs eigenlijk juist tanende is.

Mocht er aan het begin van het traject, achteraf gezien met de kennis van nu, direct gekozen zijn voor outsourcing had Instituut Verbeeten wellicht goedkoper uit kunnen zijn geweest. Wel zou er een enorm beheersingsrisico zijn op het moment dat een toen op het gebied van ontwikkeling onervaren partij als Instituut Verbeeten had besloten om een outsourcingtraject in te gaan.

Wat betreft het implementatiedeel van de opdracht zijn de resultaten gemengd. Daar waar geautomatiseerd testen aanvankelijk een zeer interessante optie leek te zijn bleek het praktisch totaal niet efficiënt te zijn om de applicatie achteraf nog te voorzien van geautomatiseerde testen. Wel is het een pré om nieuwe delen van CCure of andere grotere projecten binnen het instituut vanaf het begin op te zetten met geautomatiseerde testen.

Wel werkten de nieuwe documentatiemethodes erg prettig en wordt op dit moment gebruikt om nieuwe of veranderende functionaliteit mee te beschrijven. Mockup Builder voor schermontwerp wordt zowel door programmeurs als de key users ervaren als een erg gebruiksvriendelijke tool om mee te werken en de resulterende schermen in de documentatie als zeer duidelijk.

Use Case Scenario's waren voor veel mensen binnen het projectteam nieuw, en de voordelen werden niet direct door iedereen onderschreven. Het leverde initieel redelijk veel werk op omdat niet alleen het nieuwe stuk proces beschreven moest worden, maar ook de weg er naar toe. Gaandeweg bleek echter wel dat er toch minder vragen werden gesteld en software sneller werd opgeleverd. Of het ook zijn voordelen heeft bij het melden van bugs of kleine changes moet nog blijken.

De Coding Guidelines leverde erg interessante en leerzame discussies op, waarbij er uiteindelijk gekozen is om de regels niet in Visual Studio te af te dwingen aangezien dit er toe zou leiden dat er erg veel bestaande code aangepast moet worden omdat deze niet voldoet aan deze regels. Wel zou het aan te bevelen zijn om met een aangepaste versie van de regels te gaan werken om er voor te zorgen dat de programmeerstijl dichter bij elkaar komt te liggen van alle programmeurs die er aan werken.

Het voordeel van de best practices voorbeeld staan nu al reeds buiten kijf, vragen van het jongste projectteamlid kunnen vaak al direct beantwoord worden met een verwijzing naar de best practices. Mocht het niet voorkomen in deze code maar zou het wel kennis zijn die ervaren wordt als essentieel rondom de applicatie, wordt het alsnog toegevoegd.

Evaluatie

Terugkijkend op het project blijft bij mij het gevoel over dat de uitdaging in het project niet zozeer lag in het praktische gedeelte of zelfs het onderzoek, maar vooral het managen van tijd en prioriteit de grootste uitdagingen waren.

In de eerste plaats moest ik afdwingen bij mijn sollicitaties eind vorig jaar naar een nieuwe werkgever dat er ofwel een bedrijf waar ik gedetacheerd zou worden gevonden werd wat ook wilde ondersteunen bij mijn afstudeerproject, of intern een project zou kunnen doen, bij een volledig salaris. Uiteindelijk bleek Senet het bedrijf te zijn wat mij daar bij wilde helpen, en had als detacheerder bij Instituut Verbeeten voor mij een werk- en afstudeerplek gevonden.

Vervolgens tijdens het project bleef het altijd lastig om zowel tijdens werk- als privétijd om te gaan met zaken die ook prioriteit hadden en daardoor vaak botsten met mijn werk aan mijn afstudeerproject. Een relatiebreuk en de daar op volgende verhuisperikelen hebben hier mede aan bijgedragen.

Als werknemer heb ik eigenlijk nooit leren nee te zeggen omdat ik simpelweg mijn werk uitvoerde, en het voor mij eigenlijk niet zo veel uitmaakte of ik aan A of aan B werkte. Als afstudeerder heb ik geleerd om af en toe echt op mijn strepen te gaan staan en op te treden als mijn belangen dreigden onder te sneeuwen.

Al met al heeft het me een stuk assertiever gemaakt als werknemer, en merkte ik ook dat op het moment dat ik mijn tijd voor het project af dwong, het op een bepaalde manier ook wel gewaardeerd werd en er altijd wel een mouw aan te passen viel.

Het kwam het uiteindelijk vaak neer dat ik rondom releases meer beschikbaar was om problemen op te vangen en deadlines te halen, en dit daarna compenseerde met meer tijd voor mijn afstudeerproject in de tijd daar na. Ook bleek dat mijn initiële investering om eerst weer voor een stabiele en rustige thuisomgeving te zorgen in het tweede deel van het project duidelijk zijn vruchten af te werpen.

Er is één duidelijk moment geweest waarin het project echt bijgestuurd en zelfs een beetje aangepast is, en dat is helemaal in het begin geweest. Nadat ik zelf ook al mijn vermoedens had, werd door Martijn Lamers ook bevestigd dat het praktische deel in de initiële opzet wat te klein was.

We zijn toen bij Instituut Verbeeten nogmaals rond de tafel gaan zitten, waarbij ik aangegeven had dat ik als nieuwe programmeur op het project tegen allerlei zaken aan liep die zeker verbeterd zouden moeten worden in het geval het project geoutsourced zou worden, maar waar het project ook zeker profijt van zou hebben op het moment dat we niet door zouden gaan met outsourcing. Veel zaken die ik aankaartte bleken ook al gesignaleerd te zijn door andere mensen binnen het ontwikkelteam, en mijn voorstellen werden dan ook als zeer zinvol ervaren als toevoeging aan het project.

Appendix A: Project Initiatie Document

Inleiding

Grote delen van het Project Initiatie Document (inleiding, beschrijving van het bedrijf, de aanleiding, de opdracht en de projectomschrijving) eigenlijk allemaal al zijn behandeld binnen dit afstudeerverslag. Er is daarom gekozen om alleen de delen van het oorspronkelijke Project Initiatie Document op te nemen die nog niet benoemd zijn binnen het verslag. Dat zijn de projectfasering, de planning en de beheersaspecten.

Projectfasering

Het project is opgedeeld in drie fases, onderzoek, uitvoering en afronding. Hierbij moet gesteld worden dat het onderzoek naar outsourcing weliswaar een onderzoek heet, maar uitgevoerd wordt tijdens de uitvoerende fase.

Onderzoeksfase

Deze fase is vrij kort en bestaat uit het opstellen van dit Project Initiatie Document en het inventariseren van bestaande problemen rondom het ontwikkelproces en in de code zelf van CCure. Uiteindelijk wordt er bepaald welke problemen aangepakt worden, en aan welke eisen het rapport over outsourcing moet voldoen. Voor het onderzoek voor outsourcing wordt eerst een ondezoek gedaan in literatuur en rapportages om te achterhalen waar de risico's liggen rondom outsourcing, en wat de algemene ervaringen zijn.

Uitvoerende fase

Er wordt gestart met het zoeken naar manieren om de geïdentificeerde problematiek te verbeteren, en deze manieren worden vervolgens in de praktijk toegepast en getest. Zijn de ervaringen positief, worden deze verbeteringen verheven tot een nieuwe werkstandaard.

Ook wordt er met de in de onderzoeksfase opgedane kennis over outsourcing gekeken welke zaken onderzocht moeten worden specifiek voor Verbeeten, en worden er experts binnen het instituut geïnterviewd om een beter beeld te krijgen van de gevolgen van een outsourcingtraject. Dit wordt vervolgens vastgelegd in een eindrapportage.

Afrondende fase

Zo gauw de verbetervoorstellen voor alle aandachtspunten getest en waar goedgekeurd doorgevoerd zijn, is het proces ter verbetering voor dit afstudeerproject afgerond.

Na het opleveren van de eindrapportage is het outsourcing onderzoek afgerond,

en rest alleen nog het presenteren van de resultaten aan de Raad van Bestuur, zodat zij een uiteindelijke beslissing kunnen nemen over het al dan niet en hoe outsourcen van CCure.

Beheersaspecten

Tijd

Vanuit Instituut Verbeeten ligt er een toezegging om 3/5 van de werkweek van de afstudeerder voor werken aan het afstudeerproject te reserveren. De start van de afstudeerstage is de eerste week van februari, de inleverdatum van het afstudeerverslag ligt op woensdag 6 juni, en de verdediging ligt rond de eerste week van juli. Aan de hand van deze gegevens is er een planning opgesteld, welke terug te vinden is onder het kopje “planning”.

De risico's in de planning zijn dat er überhaupt niet goed gepland kan zijn en er te veel werk verricht moet worden in een te korte tijd, of dat doordat niet de volledige werkweek gewijd wordt aan het afstudeerproject de andere taken de overhand krijgen en er minder tijd dan gepland gewerkt wordt aan het afstudeerproject.

Geld

Er is tussen Senet BV, de detacheerder waar Lucas van Dongen voor werkt, en Instituut Verbeeten een afspraak gemaakt dat alle uren die besteed worden aan dit project voor inkoopprijs (dus zonder winst aan de kant van Senet) gedaan worden. Buiten de kosten van arbeid zijn er echter geen verder kosten aan te wijzen, en zijn er geen risico's aan te wijzen.

Kwaliteit

De kwaliteit van de deliverables is eigenlijk alleen te kwantificeren door te stellen of de opdrachtgever wel of niet tevreden is, het allemaal binnen de tijdsmarge is opgeleverd, en of het voldoet aan de eisen die Fontys Hogeschool stelt aan een afstudeerstage.

Om deze kwaliteit te waarborgen zijn er wekelijks ingeplande gesprekken tuseen Lucas van Dongen en Eelko Hondema om de voortgang te bespreken en zo nodig bij te sturen. Ook volgt er tussen Lucas van Dongen en Martijn Lamers een gesprek over de afstudeeropdracht en wordt het afstudeerverslag door hem beoordeeld voordat het ingeleverd wordt.

Dit alles zou er voor moeten zorgen dat er dermate veel bijstuurmomenten zijn dat er aan de eisen van Fontys Hogeschool voldaan wordt en Eelko Hondema tevreden is over het resultaat op het moment dat de planning gehaald wordt.

Informatie en communicatie

Op de volgende manieren zal er informatie of documentatie uitgewisseld worden binnen het project:

- Tijdens de wekelijkse meetings met Eelko Hondema over de voortgang van het project
- Twee bedrijfsbezoeken van Martijn Lamers
- Iedere persoon die geïnterviewd wordt kan de vragen persoonlijk mondeling of schriftelijk beantwoorden
- Tijdens de stand-up meetings iedere ochtend wordt het praktische deel van het project besproken met het team

Organisatie

De organisatie van het project is vrij eenvoudig opgezet. Als begeleider binnen Instituut Verbeeten is Eelko Hondema aangesteld, Hoofd ICT. Voor de praktijkimplementatie schiet zijn inhoudelijke technische kennis echter soms tekort. Op dat punt zal Bernard van Erp, als teamleider van het ontwikkelteam wat betreft de praktische zaken ook betrokken zijn.

Planning

Week	Datum	Activiteiten afstuderen	Activiteiten outsourcing rapportage
6	06-02-2012	Begin afstuderen	Begin project - Start definitiefase
7	13-02-2012		Afronden definitiefase
8	20-02-2012		Start uitvoerende fase en onderzoek
9	27-02-2012		
10	05-03-2012		
11	12-03-2012		Voorbereiden vragenlijsten
12	19-03-2012	Vakantie	Vakantie
13	26-03-2012		Inleveren vragenlijsten
14	02-04-2012		Interviews
15	09-04-2012		Interviews
16	16-04-2012		
17	23-04-2012		Einde onderzoek
18	30-04-2012	Start afstudeerverslag *	Start rapportage
19	07-05-2012		
20	14-05-2012		Rapportage af - einde uitvoerende fase
21	21-05-2012	Afstudeerverslag af	Begin afrondende fase
22	28-05-2012	Corrigeren verslag	
23	04-06-2012	Afstudeerverslag definitief	
24	11-06-2012	Voorbereiding afstuderen	Opleveren adviesrapport
25	18-06-2012		COM - presentatie en vergadering
26	25-06-2012		
27	02-07-2012	Afstudeermoment 1	Einde periode tijdelijke inhuur
28	09-07-2012	Schoolvakantie	
29	16-07-2012		
30	23-07-2012		

Appendix B: Geautomatiseerde tests email validatie

In deze appendix wordt het geautomatiseerd testen van een afgebakend deel van de applicatie getoond als voorbeeld van hoe dit wordt toegepast. Er is gekozen voor een e-mailadres validatiefunctie omdat dit redelijk herkenbaar is voor de lezer en begrepen kan worden zonder kennis van het proces of bepaalde frameworks.

Eerst worden er Unit Tests gebouwd voor alle afgebakende functies, en vervolgens worden er Integration Tests toegevoegd die controleren of het grotere geheel ook werkt zoals verwacht.

Unit Tests

De Unit Tests zijn gebaseerd op de functie FindNextQuote welke simpelweg zoekt naar het eerstvolgende dubbele quote teken wat niet geescaped is:

```
private static int FindNextQuote(string localPart, int previousQuote)
{
    int foundIndex = localPart.IndexOf('"', previousQuote + 1);

    // If we found an escaped quote keep searching
       if (foundIndex > 0 && localPart[foundIndex - 1] == '\\')
       {
           return FindNextQuote(localPart, foundIndex);
       }

       return foundIndex;
}
```

De Unit Tests tonen aan dat de handling van lege of null strings nog niet correct is. Bij de een na laatste wordt de index niet gevonden, en bij de laatste treedt een null pointer exception op. Deze twee Exceptions zijn ongewenst en moeten in vervangen worden door een specifieke Exception (een ArgumentNullException) bij de waarde null, of in het geval van een lege string -1 retourneren aangezien dat staat voor “niet gevonden”.

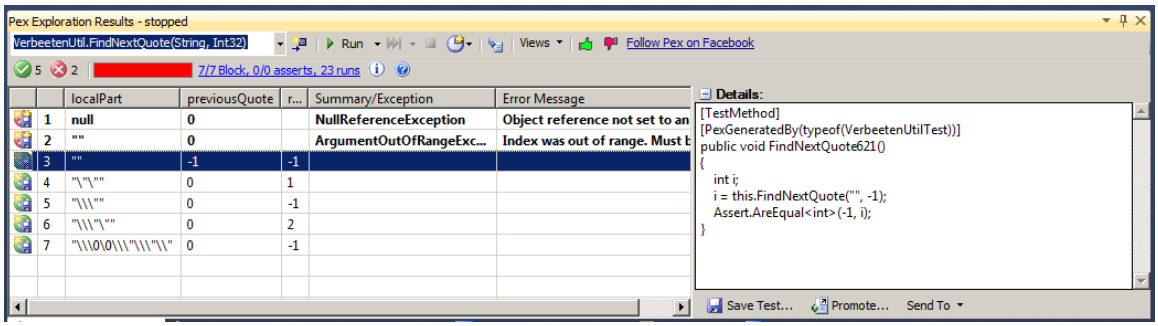
Hoewel deze fouten niet optraden in het wild omdat de aanroep in dit geval nooit met null of string.Empty gebeurden, is door het toepassen van Unit Tests op de edge cases nu wel een fout gevonden in de software die veel later grotere gevolgen zou kunnen hebben.

Op de volgende pagina staan de Unit Tests voor deze functie weergegeven.

```
/// <summary>
///A unit test for FindNextQuote
///</summary>
[TestMethod()]
[DeploymentItem("Verbeeten.dll")]
public void FindNextQuoteTest()
{
    Assert.AreEqual(0, VerbeetenUtil_Accessor.FindNextQuote("\"", -1));
    Assert.AreEqual(-1, VerbeetenUtil_Accessor.FindNextQuote("\\\"", -1));
    Assert.AreEqual(4, VerbeetenUtil_Accessor.FindNextQuote("asdf\"asdf", -1));
    Assert.AreEqual(4, VerbeetenUtil_Accessor.FindNextQuote("asdf\"", -1));
    Assert.AreEqual(10, VerbeetenUtil_Accessor.FindNextQuote("asdf\\\"asdf\"asdf",
-1));
    Assert.AreEqual(11, VerbeetenUtil_Accessor.FindNextQuote("asdf\\\"asdf\"asdf",
11));
    Assert.AreEqual(-1, VerbeetenUtil_Accessor.FindNextQuote(string.Empty, -1)); //
Throws exception!
    Assert.AreEqual(-1, VerbeetenUtil_Accessor.FindNextQuote(null, -1)); // Throws
exception!
}
```

Unit Tests uit Pex

Dit zijn de resultaten uit Pex. Pex heeft wel de twee plaatsen gevonden waar een Exception optreedt, maar stuurt standaard de verkeerde previousQuote waarde in van 0, die altijd met -1 begint. Hoewel Pex het voor elkaar heeft gekregen om alle code in deze functie te raken, wordt de code niet getest zoals het in de dagelijkse praktijk gebruikt wordt.



Afbeelding 16: Automatisch door Pex gegenereerde functiecalls

Integration Tests

De integratietests voor de emailfunctie bevatten een aantal edge-cases binnen de emailvalidatie zodat deze ook minder gebruikelijke maar volgens de RFC's correcte vormen toegelaten worden, maar ook een aantal niet correcte adressen getest worden om te controleren dat deze geweigerd worden.

Deze overkoepelende integratietests garanderen dat nadat we weten dat alle individuele onderdelen werken doordat onze Unit Tests slagen ook de hoofdfunctie werkt zoals verwacht.

```
/// <summary>
///An Integration Test for ValidateEmail
///</summary>
[TestMethod()]
public void ValidateEmailTest()
{
    string[] correctEmailAddresses =
        {
            "me@example.com", "a.nonymous@example.com", "name+tag@example.com",
            "a.name+tag@example.com", "much.\"more\\ unusual\"@example.com",
            "!#$%&'*+-/=.?^_`{|}~@[1.0.0.127]"
            , "!#$%&'*+-/=.?^_`{|}~@[IPv6:0123:4567:89AB:CDEF:0123:4567:89AB:CDEF]",
            "me(this is a comment)@example.com"
            , "niceandsimple@example.com", "a.little.unusual@example.com",
            "a.little.more.unusual@dept.example.com"
        };

    string[] incorrectEmailAddresses =
        {
            "Abc.example.com", "Abc.@example.com", "Abc..123@example.com",
            "A@b@c@example.com", "a\"b(c)d,e:f;g<h>i[j\\k]l@example.com", "me@",
            "\"spaces may be quoted\"@example.com", "just\"not\"right@example.com",
            "this is\"not\\allowed@example.com", "@example.com", "me.@example.com",
            ".me@example.com", "spaces\\ are\\ allowed@example.com", "me@example..com",
            "me.example@com", "me\\@example.com", "name\\@tag@example.com",
            "very.unusual.\"@\".unusual.com@example.com"
        };

    foreach (var emailAddress in correctEmailAddresses)
    {
        Assert.AreEqual(VerbeetenUtil.ValidateEmail(emailAddress), true,
            "Address '" + emailAddress + "' should validate");
    }

    foreach (var emailAddress in incorrectEmailAddresses)
    {
        Assert.AreEqual(VerbeetenUtil.ValidateEmail(emailAddress), false,
            "Address '" + emailAddress + "' should not validate");
    }
}
```

Email functie

```
public static bool ValidateEmail(string emailAddressInput)
{
    // Null values or empty strings are no email addresses
    if (string.IsNullOrEmpty(emailAddressInput))
    {
        return false;
    }

    // Find the occurrence of the last @-sign
    int lastAtSignOccurrence = emailAddressInput.LastIndexOf('@');

    // The last @-sign can never be escaped
    if (lastAtSignOccurrence > 0 && emailAddressInput[lastAtSignOccurrence - 1] == '\\')
    {
        return false;
    }

    bool domainIsIp = false;
    string domainPart = emailAddressInput.Substring(lastAtSignOccurrence + 1,
        emailAddressInput.Length - lastAtSignOccurrence - 1);

    // We should have a domain part, otherwise it's not an e-mail address
    if (string.IsNullOrEmpty(domainPart))
    {
        return false;
    }

    // Remove brackets of IP address if it's not a regular DNS name
    if (domainPart[0] == '[' && domainPart[domainPart.Length - 1] == ']')
    {
        domainIsIp = true;
        domainPart = domainPart.Substring(1, domainPart.Length - 2);

        // Filter out IPv6 string
        domainPart = domainPart.Replace("IPv6:", string.Empty);
    }
    else
    {
        // The DNS name should contain at least one dot, otherwise it's only TLD
        if (domainPart.IndexOf('.') < 0) return false;
    }

    // If we still didn't find an @-sign it's not a valid address
    if (lastAtSignOccurrence < 0)
    {
        return false;
    }

    string localPart = emailAddressInput.Substring(0, lastAtSignOccurrence);

    // If there is no local part it's not a valid address
    if (string.IsNullOrEmpty(localPart))
    {
        return false;
    }

    // Check the domain
```

```
    UriHostNameType hostNameCheck = Uri.CheckHostName(domainPart);
    switch (hostNameCheck)
    {
        case UriHostNameType.Unknown:
        case UriHostNameType.Basic:
            return false;
        case UriHostNameType.Dns:
            if (domainIsIp)
            {
                return false;
            }
            break;
        case UriHostNameType.IPv4:
        case UriHostNameType.IPv6:
            if (!domainIsIp)
            {
                return false;
            }
            break;
        default:
            throw new ArgumentOutOfRangeException();
    }

    // Check the local name
    // Remove all comments
    localPart = RemoveComments(localPart);

    // There is a difference between double-quoted and non-double-quoted parts, so we
need to chop the localPart into quoted and non-quoted sections
    bool allQuotedSplitsFound = false;
    List<string> quotedStrings = new List<string>();
    while (!allQuotedSplitsFound)
    {
        int firstQuote = FindNextQuote(localPart, -1);
        if (firstQuote < 0)
        {
            allQuotedSplitsFound = true;
            continue;
        }

        // We don't want backslashes outside of double quotes
        if (firstQuote != 0 && localPart[firstQuote - 1] == '\\')
        {
            return false;
        }

        // Second quote should exist after the first one
        int secondQuote = FindNextQuote(localPart, firstQuote);
        if (secondQuote < 0)
        {
            // Otherwise we don't have to split
            allQuotedSplitsFound = true;
            continue;
        }

        // The quotes should be first and last, or surrounded by dots
        if (!(firstQuote == 0 || localPart[firstQuote - 1] == '.') || !(secondQuote !=
localPart.Length || localPart[secondQuote + 1] != '.'))
        {
            return false;
```

```
        }

        // So there is a quoted part, so we need to put the quoted string in a separate
array
        quotedStrings.Add(localPart.Substring(firstQuote + 1, secondQuote - firstQuote -
1));

        bool doubleQuoteWasLastCharacter = secondQuote == localPart.Length - 1;

        // remove it from the main local part
        localPart = localPart.Remove(firstQuote, secondQuote - firstQuote + 1);

        // Clean up the main local part of remaining dots, because some remaining dots
       would invalidate the local part
        if (localPart.Length != 0)
        {
            // Chop off first dot if the quote started right at the beginning if it
didn't start with a quote
            if (firstQuote == 0 && !doubleQuoteWasLastCharacter && localPart[0] == '.')
            {
                localPart = localPart.Remove(0);
            }

            // Chop off the last dot
            if (doubleQuoteWasLastCharacter && localPart[localPart.Length - 1] == '.')
            {
                localPart = localPart.Remove(localPart.Length - 1);
            }

            // If both cases don't match, it could be a quoted part within the normal
part
            if (firstQuote != 0 && !doubleQuoteWasLastCharacter &&
localPart.Substring(firstQuote - 1, 2) == "..")
            {
                localPart = localPart.Remove(firstQuote - 1, 2);
            }
        }
    }

    // Process all quoted strings according the more lax rules for quoted strings
    foreach (string quotedString in quotedStrings)
    {
        if (!IsValidLocalPartChunk(quotedString, true))
        {
            return false;
        }
    }

    // Validate the main chunk of the address (if applicable) with the normal rules
    if (!(string.IsNullOrEmpty(localPart) && quotedStrings.Count > 0) && !
IsValidLocalPartChunk(localPart))
    {
        return false;
    }

    // If we did not return false in the mean time because an error was found, it's a
valid email address
    return true;
}
```

Afstudeerverslag

Outsourcing van de onderhoudsfase van CCure: Automatisering van oncologische behandeltrajecten

Naam	Lucas van Dongen
Studentnummer	2088881
Afstudeerperiode	1 februari tot 1 juli 2012
Opleiding	HBO Informatica, Fontys Hogeschool Eindhoven
Afstudeerrichting	Software Engineer
Project	Onderzoek naar de mogelijkheden voor het outsourcen van CCure
Stagebedrijf	Instituut Verbeeten
Bedrijfsmentor	dhr. Eelko Hondema
Afstudeerdocent	dhr. Martijn Lamers

Instituut Verbeeten

Vestiging Tilburg

Brugstraat 10

5042 SB Tilburg

Telefoon: +31 (0)13 594 77 77

Telefax: +31 (0)13 594 76 83

Fontys Hogeschool

Afdeling ICT

Rachelsmolen 1

5612 MA Eindhoven

Telefoon: +31 (0)8850 77299