Quadrant Software bv

Productverslag van Ronald de Haan

Stagiair van de Hogeschool Rotterdam

Student	Ronald de Haan
Studentnummer	0809766
Opleiding	Informatica
School	Hogeschool Rotterdam
Stagebedrijf	Quadrant Software bv
Stagebegeleider bedrijf	Martin Bos (martin.bos@quadrant.nl)
Stagebegeleider school	Bob Joziasse (b.joziasse@hro.nl)
Datum	24 september 2011
Documentstatus	Definitief

Eglantierbaan 95 - 2908 LV Capelle aan den IJssel
Postbus 76 - 2900 AB Capelle aan den IJssel
Tel. 010-264 63 00 – Fax. 010-264 63 01
http://www.quadrant.nl

Quadrant Software bv

Inhoudsopgave

Quadrant Software bv

1. Inleiding

De eerste helft van mijn derde studiejaar ben ik bij Quadrant aanwezig geweest als stagiair (programmeur). Gedurende deze periode heb ik ervaren hoe het is om binnen een bedrijf werkzaam te zijn. Ik zat op een afdeling die onderzoek doet naar C# .NET. Ze zijn met name aan het kijken hoe ze dit kunnen inzetten in de toekomst om hun product om te bouwen of te herbouwen. Dit omdat het huidige product, genaamd King, is geschreven in Delphi en dit aardig is verouderd. Ook is dit niet Object georiënteerd geprogrammeerd, wat het onderhoud heel lastig maakt.

In dit document zal ik aan de lezer laten zien welke producten ik heb opgeleverd en wat daarbij goed en wat minder goed ging.

Quadrant Software bv

2. De producten

2.1 Artikelteksten

2.1.1 Beschrijving van wat ik heb gedaan

In het offerteprogramma waar we momenteel mee bezig zijn, kun je offertes invoeren met offerteregels. Deze voer je in door het artikelnummer (of zoekcode) op te geven, waarna er een aantal andere velden als prijs en tekst op offerte worden ingevuld. Dit laatste was echter nog niet geïmplementeerd en zou ik moeten doen. De database er omheen was er al wel en de teksten stonden daar ook al in. Er ontbrak alleen nog een entity en een mapping voor NHibernate naar de database.
Ik ben begonnen met het maken van de entity. Dit houdt niet meer in dan een class met daarin de attributen die uit de database moeten komen. Dus een entity is in principe de in code gebouwde databaseregel. In dit geval bevat de entity een Taal en de Tekst.
De situatie is als volgt: Je hebt een OfferteEntity. Deze bevat OfferteRegelEntities. Hierin zitten weer Artikelen.

In ArtikelEntity wordt door middel van een mapping alle teksten (nu nog in alle talen) opgehaald voor dit artikel. In de offerte wordt bepaald welke taal de offerte zal handhaven. Dit weet hij door de taal van de ingevoerde debiteur op te vragen. Het kan echter zijn dat er geen debiteur is ingevoerd. In dat geval zal de basistaal van de administratie gekozen worden. Zodra je nu een nieuwe offerteregel toevoegt, wordt deze taal doorgegeven en weet de regel het ook. Als je nu in die regel een artikelnummer (of zoekcode) invoert, wordt (weer door een mapping) de juiste artikeltekst (met de juiste taal), uit het artikel gehaald en weergegeven.
Met deze opdracht ben ik ongeveer tweeënhalve week bezig geweest. Dit is op zich vrij lang. Dat komt omdat ik nog heel de structuur van de bestaande code moest doorkrijgen en ook moest leren wat NHibernate nu eigenlijk is en wat het doet en hoe zijn mappings werken. Als ik deze opdracht weer zou moeten doen, verwacht ik dat ik er niet meer dan een week mee bezig ben.

2.1.2 Wat er goed ging

Dit was mijn allereerste bezigheid met de echte code. Ik moest dus nog wel uitzoeken hoe alles in elkaar stak. Dit had ik vrij snel door en ik kon in de ArtikelEntity een array gaan maken die zich vulde met artikelteksten. Hierna zal de OfferteRegelEntity, wanneer het wordt aangemaakt, de juiste tekst uit deze array halen. Het mechanisme dat de juiste taal gebruikt was nog niet aanwezig, dus voorlopig heb ik hardcoded de Nederlandse tekst laten pakken.

Later is de taal beschikbaar gekomen. Hiervoor heb ik ook de selectiemethode gebouwd. Deze moest kijken of er in de offerteregel een debiteur is ingevuld. Als dat het geval is, gebruikt hij de taal van de debiteur, dus is het een Engelse debiteur, dan zal Engels als taal gekozen zijn. Als er echter geen debiteur is ingevuld, dan zal de ingestelde taal van de hele administratie gekozen worden. Als ook deze geen artikeltekst op zal leveren, zal er de omschrijving van het artikel neergezet worden, maar dat gebeurd later pas, omdat dat nog niet in de requirements zat voor deze fase.

2.1.3 Wat er minder ging

Eigenlijk kan ik dat met deze eerste opdracht nog niet zeggen, ik moest er nog helemaal inkomen.

2.2 Unittests en valuetypes

2.2.1 Beschrijving van wat ik heb gedaan

Na de eerste opdracht was het weer tijd voor verdieping. Op internet was een presentatie te vinden over Value Objects[1] (binnen Quadrant valuetypes genoemd). Wat zijn valuetypes? Dat zijn classes met methoden en operators (een operator is bijvoorbeeld plus, min, keer, enz. maar ook de == constructie waarmee je twee objecten kunt vergelijken) die bij elkaar horen. Zo heb je software waarin een telefoonnummer als een string is gedeclareerd en daarna op diverse plaatsten in hun code gecontroleerd wordt of het daadwerkelijk een telefoonnummer is. Als je dit oplost door een valuetype Telefoonnummer aanmaakt, voorkom je dat ergens een onmogelijk telefoonnummer voorkomt. Intern is het valuetype nog steeds wel een string, maar als je een nieuw telefoonnummer aanmaakt, wordt er eerst gecontroleerd of het een valide nummer is. Zo niet, dan heb je al bij de invoer een error en niet ergens later in je code.
Daarnaast heb ik geleerd wat het inhoud om unittests te schrijven. Op school is hier eigenlijk nooit aandacht aan besteed. Achteraf stelt het misschien niet zo veel voor, maar het is toch handig om dat even onder de aandacht te brengen. Voorheen zou ik een test hebben geschreven die alles op allerlei manieren zal testen. Dit is echter overdreven. Je moet alleen code testen waar je problemen verwacht, of code die meer doet dan alleen een getalletje of tekstje teruggeven.

Nadat ik dit heb geleerd, heb ik het direct in de praktijk gebracht. We hebben hier een heleboel valuetypes die nodig eens opgeschoond moeten worden en waarvoor nog geen tests geschreven zijn. Hier ben ik dan ook ongeveer vier weken mee bezig geweest en ik kreeg zelfs complimenten dat ik het testen zo goed doorhad!

2.2.2 Wat er goed ging

Achter deze opdracht zat wat meer theorie vast die ik me eerst eigen moest maken. Gelukkig kon ik dit snel vatten en er mee aan de slag. Er werd ook tegen mij gezegd dat ik de unittests echt maakte zoals het moest, dat ik een snelle leerling ben.

Wat zijn valuetypes? Een valuetype is een class, gebaseerd op een systeemtype als een String, int, double, wat dan ook, die alle methoden bevat die bij een bepaald “item” hoort.
Bijvoorbeeld het valuetype “telefoonnummer” zal bestaan uit het systeemtype String, zal geen letters accepteren, hooguit een min-teken als je dat wilt (ze zijn naar je eigen inzicht te bouwen).
Het valuetype “string40” gebruikten we bijvoorbeeld voor een String die maximaal 40 karakters mocht bevatten. Er zal dan een methode inzitten die kijkt wat de maxLength is.

Valuetypes bieden het voordeel dat je dus alle code die er mee te maken heeft, gebundeld op één locatie hebt staan, in plaats van dat er verspreid over je hele code een paar methodes staan die iets controleren van een systeemtype. Verder garandeer je dat als je een telefoonnummer hebt, er ook daadwerkelijk in staat wat je hebt gemaakt. Want een String telefoonnummer kan per ongeluk ook letters bevatten, maar een Telefoonnummer telNr kan dat niet.

[1] Value Objects van Dan Bergh Johnsson.

Wat heb ik nu geleerd van de unittests?

Een goede unittest test alle methoden die “iets doen”. Een simpele methode die enkel een variabele uit die klas terug geeft, hoeft niet getest te worden, omdat je dan eigenlijk het framework van Microsoft zelf aan het testen bent en dat is niet de bedoeling. Verder is het bij valuetypes gebaseerd op integers of doubles, waar ook berekeningen in zitten, deze ook controleerd of de verwachte uitkomst er ook daadwerkelijk uit komt. Ook moet je, bijvoorbeeld bij string40, controleren wat er gebeurd als je er 40 karakters invoerd, maar ook als je er 41 invoerd. Dit noemen ze dan weer grenswaardeanalyse. Dit geldt ook voor int en double gebaseerde valuetypes, waarbij er een min- en maxValue is opgegeven.

Je test min- en maxValues op de volgende manier. Je verdeelt het in groepen, een groep die onder de minimum zit, een groep die tussen het minimum en maximum zit en een groep die er boven zit. Je kiest voor elke groep een willekeurig getal en kijkt wat er mee gebeurd. Als het goed is, zal in dit geval alleen de middelste groep goedgekeurd worden.

2.2.3 Wat er minder ging

Bij deze opdracht ging alles heel goed.

2.3 Cache remover

2.3.1 Beschrijving van wat ik heb gedaan

Op de woensdag in de 10e week van mijn stage heb ik weer een grotere opdracht gekregen. NHibernate laadt de mappings in het programma om het te kunnen gebruiken. Elke keer dat je het programma opstart, wordt dit uitgevoerd. Dit is niet efficiënt omdat dit vrij lang duurt. In de code is al een switch gemaakt om deze cache naar een bestand op te slaan om het daarna weer in te kunnen laden. Deze stond echter uit omdat er een aantal problemen mee optraden. Eén van de problemen was dat wanneer je zelf iets aan de code had aangepast, de cache verwijderd dient te worden. Dit kon echter niet helemaal door C# worden bepaald. Hiervoor moest ik iets bedenken dat dat wel kon.

Allereerst heb ik dit geprobeerd door te kijken of je van een assembly (het project dat is gebuild door Visual Studio als een .dll bestand) een hashcode kan opvragen en die met de volgende run te vergelijken. Echter wordt bij het compileren altijd ergens iets aangepast door Visual Studio, zodat deze vergelijking altijd zal denken dat de assembly anders is.
Na een week heb ik dit opgegeven en ben ik een ander idee gaan uitwerken. We maken allemaal gebruik van Subversion. Daarin zit versiebeheer. Subversion “weet” dan ook of een bestand lokaal is veranderd, maar ook, als het niet veranderd is, welk versienummer het heeft, zodat het ook weet of er extern wijzigingen zijn gemaakt. Dit is precies wat ik zocht!
Ik heb een programma gemaakt dat de te onderzoeken paden uit een XML bestand inleest en dan via de commandline vraagt of er op dat pad wijzigingen zijn aangebracht. Dit gebeurd via het commando “svn diff <pad>”. Als er wijzigingen zijn, zullen de cache bestanden van NHibernate, die ook in de XML staan, worden verwijderd, zodat het programma ze weer opnieuw kan opbouwen. Om ervoor te zorgen dat nieuwe wijzigingen, als er geen commit is gedaan, toch worden gezien, sla ik de Subversion resultaten op en vergelijk die ook met elkaar. Het CacheCheck programma kan dus wel zeggen dat Subversion wijzigingen ziet, maar als de resultaten met elkaar overeenkomen, zijn er geen nieuwe wijzigingen gemaakt sinds de vorige keer en hoeft de cache niet opnieuw weggegooid te worden.

2.3.2 Wat er goed ging
Deze opdracht was al wat ingewikkelder, omdat ik een losstaand programmaatje moest maken, dat toch kon weten wat de status is van het lokale project ten opzichte van subversion. Het is me uiteindelijk wel gelukt om het werkend te krijgen.

2.3.3 Wat er minder ging
Ik ben eigenlijk op het verkeerde spoor begonnen. Ik ging kijken of je van de gemaakte assembly (meestal een .dll bestand) een hashcode kon opvragen en deze bewaren en vergelijken met volgende builds. Deze hashcode verandert echter bij elke build, ook al is er daadwerkelijk niks veranderd in de code. Dit was dus niet wat ik zocht.
Hierna ben ik het tweede verkeerde spoor ingegaan. Ik ging kijken of je subversion libraries had die je in je code kon gebruiken. Hier heb ik lang over gedaan om dit werkend te krijgen. Op een gegeven moment heb ik dit gestopt en ben ik gaan kijken of het ook anders kan. Toen kwam ik op het idee om het via de commandline te doen, aangezien C# .NET ook de commandline aan kan spreken. Hierna kreeg ik het allemaal vrij snel voor elkaar. Ik zorgde dat er via de commandline het commando “svn diff <solution path>” werd gegeven. Dit zal aangeven of subversion ziet dat er lokaal wijzigingen zijn gemaakt in de code ten opzichte van de laatste versie van de code die uit subversion is gehaald. Ook werd er voor gezorgd dat de diff regels opgeslagen worden in een tekstbestand, zodat ze de volgende keer dat dit programma draait, met elkaar vergeleken kunnen worden. Dit zorgt ervoor de je ook weet of er na je laatste wijzigingen, ook weer nieuwe wijzigingen zijn gemaakt. Dit omdat subversion alleen maar aangeeft OF er wijzigingen zijn, maar niet of dat er net ook al in zat.

2.4 Certificaat check

2.4.1 Beschrijving van wat ik heb gedaan
Ik heb ook onderzoek gedaan naar het gebruik van certificaten, zodat je software beveiligd is tegen hackers. Dit certificaat laat zien dat de software afkomstig is van een erkend bedrijf en dat er niet mee geknoeid is nadat het is gecertificeerd. King bestaat uit een setup.exe Deze zal weer een aantal kleinere setups uitvoeren. Het zou dus kunnen dat het betreffende setup bestand wordt vervangen door een aangepaste versie. Dit is dan ook niet de bedoeling. Ik moest uitzoeken of het mogelijk is om in de setup (opgebouwd door middel van C++ code) het certificaat van de andere setup bestanden op te halen en uit te lezen is. Dit is mij gelukt: Ik kon de naam van de leverancier en de thumbprint (unieke code die elk bedrijf krijgt bij een certificaat) ophalen en dus ook vergelijken of het de correcte gegevens zijn.

2.4.2 Wat er goed ging
De problemen die ik had, bleken bekend te zijn bij de persoon waarvoor ik dit heb gebouwd.

2.4.3 Wat er minder ging
Het eerste en vervelendste probleem was dat er weinig over de certificate methodes is gedocumenteerd. Ik liep tegen problemen aan dat de CRL (Certificate Revocation List) niet gevonden kon worden. Na lang zoeken en vragen aan de persoon waarvoor ik dit heb gebouwd, bleek het te zijn dat je deze moet toevoegen/toestaan in een lokale certificate store op de computer. Ik ben er dus verder niet uitgekomen en heb dit project onderbroken.

2.5 Testdatabases opruimen

2.5.1 Beschrijving van wat ik heb gedaan

Ik heb een aantal testdatabases opgeruimd. Officieel is er één testdatabase. Echter zijn er in de loop van de tijd een aantal bijgekomen om op specifieke delen te testen. Dit is echter erg omslachtig. Daarom zijn de meeste tests verhuisd naar de originele testdatabase. Dit houdt in dat je de desbetreffende unittests moet omzetten naar de originele database en, indien nodig, aanpast zodat ze weer werken. Ik heb drie aardig grote databases kunnen omzetten.

2.5.2 Wat er goed ging

Ik heb zo'n drie databases op kunnen ruimen. Dit kon omdat er al een officiële testdatabase bestond met dezelfde tabellen en dezelfde indeling. Echter moest ik sommige specifieke regels in de unittests aanpassen, want, bijvoorbeeld, op regel 4 bij naam stond nu geen "Piet", maar "Klaas". En de test verwachtte nog "Piet".

2.5.3 Wat er minder ging

Alles ging eigenlijk wel goed.

3. Overige bezigheden

Buiten deze opdrachten, heb ik mezelf ook nog verdiept en heb ik informatie over nieuwe/andere technieken opgedaan. Het was de bedoeling dat ik naar één of meer seminars/bijeenkomsten zou gaan, maar dat is helaas niet gelukt. Dit kwam omdat het meestal op een donderdagavond viel, waarop ik om privéomstandigheden niet kon of op een vrijdag waarop ik naar school moest. In plaats daarvan heb ik een aantal verslagen/samenvattingen van collega's gelezen en heb ik video's/powerpoints van de desbetreffende sessies gezien.

Eén mooi en duidelijk voorbeeld is de video van "Value Objects" door Dan Bergh Johnsson. Dit heeft mij weer een nieuw inzicht gegeven in de bij Quadrant genoemde valuetypes, wat er voor zorgt dat je alle code die bij een type hoort, allemaal bij elkaar hebt. Zoals een bedrag, dan heb je een class met methodes die te maken hebben met bedragen, bijvoorbeeld optellen, afronden, enzovoorts. Dit heb ik kunnen toepassen in de valuetypes die ik gemaakt of herschreven heb, zoals eerder in het verslag is te lezen.

Daarnaast zijn er ook tijdens mijn stage nog interne sessies gehouden. Zo heb ik een sessie meegemaakt waarin een demo werd gegeven over Enterprise Architect. Dit is een programma waarin je je programma van tevoren kan ontwerpen. Ook kun je er requirements en bugs aan toe voegen. Nog een handige optie is dat het ontwerp kan omzetten in code. Hierdoor heb je de classes en methodes al, maar ook kun je dan de bugs als comment in de class zelf laten zetten.

Ook worden nieuwe ontwikkelingen en "probeersels" aan elkaar gepresenteerd. Zo is er iemand bezig om een web applicatie te bouwen in Silverlight. Dit als een research project, maar wat wel gebruikt gaat worden. Hierin zie je ook duidelijk veel ontwikkelingen en er zijn enorm veel mogelijkheden met Silverlight. Een mooi voordeel is dat het gewoon in C# wordt geprogrammeerd en dat het heel mooi in het domain van de rest van de software aansluit.

Als klap op de vuurpijl heb ik een sessie meegemaakt van Anko Duizer van Microsoft. Hij kwam binnen Quadrant vertellen over de cloud van Microsoft. Hij vertelde onder andere dat een applicatie in de cloud schaalbaar is en dat je het gebruik betaald in plaats van een vast abonnement. Daarnaast heeft hij het gehad over het platform waarop je kan ontwikkelen voor de cloud: Windows Azure. Dit is bijna een normale Windows omgeving, met als verschil dat alle tools die je nodig hebt om in de cloud te ontwikkelen aanwezig zijn. Veel meer informatie mag ik hier niet kwijt, omdat dit alleen toegankelijk is voor Microsoft partners.