HOGESCHOOL GENT
LID VAN DE ASSOCIATIE UNIVERSITEIT GENT

DEPARTEMENT TOEGEPASTE INGENIEURSWETENSCHAPPEN
CAMPUS SCHOONMEERSEN - GENT

Uitgebreid voorstel Masterproef Informatica

Titel van het project: Migratie strategieën van C++ naar C#

Datum: 20 december 2011
Naam student: Martijn Saelens
Interne promotor: Leen Brouns
In samenwerking met: Bedrijf / ~~IBBT~~ / ~~UGENT~~ / ~~Hogeschool Gent~~
Algemene informatie voor extern bedrijf:

Naam van het bedrijf: ICORDA NV

- Is dit de 1e masterproef in het bedrijf in samenwerking met onze opleiding? ~~Ja~~/Nee
- Is er in het bedrijf inhoudelijke en technische begeleiding mogelijk? Ja/~~Nee~~
- Kan de student in het tweede semester (februari-mei) 3 dagen per week in het bedrijf/onderzoekscentrum aanwezig zijn om te werken aan de masterproef? Ja/~~Nee~~

Begeleiding:

Externe promotors – andere begeleiders:
Joris Van Maldeghem (jvanmaldeghem@icorda.be)

Bespreking door de werkgroep (niet invullen bij indienen van een voorstel)

Beslissing: goedgekeurd - herwerken tegen … / …
Minimale uitbreidingen:
Opmerkingen:
Advies van collega's:

Doelstelling, bestaande situatie, probleemstelling en gedetailleerde omschrijving van de opdracht die minimaal moet worden verwezenlijkt:

ICORDA bezit een zestal projecten geschreven in Visual Studio 6.0 C++ en Visual Studio 2005. De code van deze projecten is niet compatibel met het .NET framework.
De huidige markt is voor een groot deel overgeschakeld naar C# waarbij men ontwikkelt in een Visual Studio 2010 omgeving. Dit biedt een groot aantal voordelen. Enkele voorbeelden:

- applicaties ontwikkeld in verschillende .NET-talen kunnen zonder problemen naadloos samenwerken
- er zijn veel extra handige functies beschikbaar in .NET, net als veel third-party tools en libraries
- het .NET framework zorgt ook voor een beter abstractie van de applicatie en de onderliggende machine.
- Visual Studio 2010 bevat meer technologieën en recentere versies van reeds aanwezige technologieën dan Visual Studio 6.0 zoals het .NET framework
- C# is ontwikkeld zodat het schrijven van code intuïtiever aanvoelt en minder foutgevoelig is (bv. Garbage collection, het onnodig maken van pointers)
- C# introduceert ook veel nieuwe codestructuren zoals bv. delegates en ondersteund object georiënteerd programmeren beter dan standaard C++

Deze lijst van voordelen is zeker niet beperkt tot de voordelen die hier zijn opgesomd, het is dan ook duidelijk waarom de markt deze overschakeling maakt.
Men wil bij ICORDA niet achterlopen en dus ook voor deze projecten de overschakeling maken naar C# binnen een Visual Studio 2010 omgeving.

Het doel van deze masterproef is om deze projecten op zijn minst om te zetten naar C++.NET zodat men deze kan benaderen met Visual Studio 2010. Eens dit lukt, kunnen we een stap verder gaan naar het ultieme doel: de code van deze projecten te herwerken naar C#.NET.
Dit alles moet op een zo generiek mogelijke manier gebeuren: als het werkt op een project, moet het ook werken op alle andere projecten. Uiteindelijk moeten we dus een procedure ontwikkelen, die deze omzetting zo generiek en geautomatiseerd mogelijk kan uitvoeren.

Dit is het hoofddoel. Het kan zijn dat er hier en daar door het gebruik van C# betere of compactere codestructuren mogelijk zijn, maar het inwerken van die codestructuren is hier maar een bijzaak. We weten trouwens niet welke gevolgen dit zou kunnen hebben voor de applicatie: er kunnen hierdoor ongemerkt bugs ontstaan die de applicatie ontregelen en onbetrouwbaar maken. Dit zou de tijd en moeite van het toenmalig testen van de originele code van het project grotendeels overbodig maken en het uitgebreid testen van de nieuwe code vereisen. Men zal moeten afwegen of de voordelen opwegen tegen de extra moeite en tijd die het inwerken zal vereisen.

De manier waarop men deze doelen moet verwezenlijken ligt niet vast, zolang de doelen worden verwezenlijkt. Men raadt echter vanuit ICORDA de volgende oplossingsmethode aan:
Men begint met een analyse van alle projecten die moeten geconverteerd worden. Uit deze projecten kiest men dan een proefproject. Dit project zal worden gebruikt om de conversiemethode op te stellen en uit te testen. Alle projecten zijn opgesteld volgens het "three-layered architecture" (ook bekend als het "three-tier architecture"). Dit houdt in dat alle projecten opgedeeld zijn in een presentatielaag ("presentation layer"), een functionele laag ("business layer") en een datalaag ("data layer"). Men moet elk van deze lagen van het proefproject uitgebreid analyseren. Op basis van deze analyse kan men dan beginnen met het opbouwen van een generieke conversiemethode. Men moet deze methode vervolgens uitgebreid testen en nazien of deze ook toepasbaar is op de andere applicaties.
Hierna moet men ervoor zorgen dat de geconverteerde projecten kunnen geïmplementeerd worden in het bedrijf. Dit houdt in dat ze op basis van het development framework en de buildserver van het bedrijf kunnen draaien. Het development framework zorgt voor het versiebeheer van de gebruikte technologieën en de buildserver maakt het project onafhankelijk van de computer waarop het wordt ontwikkeld.

Als eindproduct moet men dus een methode hebben die op een zo generiek en geautomatiseerd mogelijke manier de huidige C++ code van de applicaties op zijn minst omzet naar C++.NET compatibel met Visual Studio 2010 en dit dan uiteindelijk converteert naar C#.

Problemen die moeten opgelost worden (niet te gedetailleerd):

Er zijn eigenlijk twee grote problemen die opgelost moeten worden: het converteren van de C++ code van de projecten naar C++.NET en het converteren van de C++.NET code naar C#.

Alvorens we deze twee stappen kunnen nemen, moeten we eerst de C++ kern uit de projecten filteren en onderscheiden van de randtechnologieën. Pas als dit gebeurd is, kunnen we de twee grote stappen nemen.

De eerste stap is het converteren van de huidige C++ code naar C++.NET code compatibel met de Visual Studio 2010 omgeving. Dit hoeft niet noodzakelijk in èèn stap: men kan de code eerst omzetten naar code compatibel met Visual Studio 2005 en dan pas naar code compatibel met Visual Studio 2010. Hoe minder stappen het converteren vereist, hoe efficiënter de conversie.
De uitdaging hier is dat standaard C++ andere libraries gebruikt dan C++.NET. Dit houdt in dat er klassen, functies, … zullen zijn die in C++.NET wel voorkomen maar niet in standaard C++ of omgekeerd, of dat deze dezelfde naam hebben in beide versies, maar niet dezelfde functionaliteit of signatuur. Dezelfde problemen kunnen voorkomen met de overschakeling van Visual Studio 6.0 naar Visual Studio 2010 (met de eventuele tussenstap van Visual Studio 2005). Men zal deze incompabiliteiten moeten opsporen en herwerken, zodat de functionaliteit van het project niet wijzigt en er geen bugs worden veroorzaakt.

De tweede stap is de conversie van C++.NET naar C# (beide binnen een Visual Studio 2010 omgeving). De overstap zou enigszins vergemakkelijkt moeten zijn nu de code en de resulterende code na conversie beide compatibel zijn met het .NET framework. Dit houdt in dat meeste functies en klassen beschikbaar zijn in beide talen en grotendeels dezelfde functionaliteit en signatuur hebben. Toch moeten we dit ook controleren. Niet alles uit C++.NET komt overeen met C#.NET: bv. Wat doen we met pointers? Ook hier moeten we alle incompatibiliteiten opsporen en herwerken zonder enige functionaliteit te wijzigen of bugs te veroorzaken.

De eerste stap is de belangrijkste. Pas als men deze stap succesvol heeft uitgevoerd kan men doorgaan met de tweede stap. Bij elke stap moet men er op letten dat de geconverteerde kern terug gebruikt kan worden in samenwerking met de nodig randtechnologieën (eventueel met hogere versies van die technologieën). Zo gebruikt men in sommige applicaties externe libraries van een extern bedrijf. Deze moeten terug gebruikt kunnen worden in de geconverteerde projecten.

Uiteindelijk moeten we er ook nog voor opletten dat de projecten compatibel zijn met het development framework en de buildserver van het bedrijf, zodat de versies van de gebruikte technologieën overal overeenkomen en de projecten onafhankelijk van de computers waarop deze ontwikkeld zijn probleemloos uitgevoerd kunnen worden.

Technologieën die aan bod komen:

De technologieën die het meest aan bod zullen komen zijn uiteraard de talen C++ en C#, samen met het .NET framework.
Het .NET framework bestaat uit enkele componenten: een verzameling klassen, enkele compilers en een "application virtual machine". Bij het ontwikkelen van een applicatie kan men de klassen uit het .NET framework gebruiken. Als men het project compileert, compileert de juiste compiler voor de broncode het project naar de Microsoft Intermediate Language tussentaal, dat op zijn beurt door de Common Language Runtime wordt omgezet naar machinecode door middel van "just in time" compilatie en wordt uitgevoerd.
Ook wordt er in de oorspronkelijke C++ projecten veelvoudig gebruik gemaakt van de Microsoft Foundation Class library (MFC) en enkele libraries van een externe partner. De MFC library bevat klassen en structuren die het schrijven van applicaties voor windows heel wat vereenvoudigt.
COM+ (Component Object Model+) is ook een technologie waarmee men vaak in contact zal komen en wordt hier vaak gebruikt als een vorm van data acces component. COM+ is de uitbreiding op COM en tevens de opvolger van MTS (Microsoft Transaction Server). COM+ kan beschouwd worden als een serverprogramma dat diensten aanbied aan COM componenten.
Naast COM+ zal men ook met webservices werken zoals bv. RESTful webservices (REpresentational State Transfer). Deze laten toe dat men via standaard webprotocollen en XML op afstand kan communiceren met een server.
Technologieën die eventueel kunnen aangewend worden in geconverteerde projecten zijn ADO.NET (ActiveX Data Objects for .NET) en LINQ (Language Intergrated Query) om data aan te spreken.

LINQ wordt net als ADO.NET gebruikt in de datalaag om op een generieke manier verscheidene soorten data te kunnen aanspreken (bv. databanken, XML-bestanden). het wordt vaak gebruikt omdat het sterk gelijkt op SQL.
Uiteindelijk heeft men ook nog de development framework CurrentVersion om ervoor te zorgen dat alle technologieën dezelfde versie hebben en de FinalBuilder buildserver van VSoft Technologies om ontwikkelde projecten te kunnen runnen op elke computer, onafhankelijk van de computer waarop het ontwikkeld is.

Mogelijke uitbreidingen en opties:

- Coëxistentie van legacy C++ en C# applicaties door integratie van webservices
- Integratie met sharepoint (incl. mobile apps)

Vernieuwende aspecten:

- C++ en C#: voor de analyse van de projecten zal men zich goed thuis moeten voelen in complexe structuren en syntax. Men zal zich zeer goed vertrouwd moeten maken met C++ en C#. Men heeft tijdens de opleiding al C# en C++ gebruikt.
- .NET framework: om de conversie succesvol te kunnen uitvoeren is een zeer uitgebreide kennis van het .NET framework vereist. In de opleiding heeft men de mogelijkheden ervan nog niet echt ten volle gebruikt.
- Microsoft Foundation Class library (MFC): Dit heeft men in de opleiding nog niet gezien. Zoals reeds vermeld is de MFC library een verzameling van klassen die het ontwikkelen van projecten voor windows sterk vereenvoudigt.
- Component Object Model+ (COM+): In de opleiding heeft men tot nu toe enkel gewerkt met COM-componenten. Men gaat ook hier dieper in op zowel COM als COM+, aangezien COM+ de uitbreiding is van COM.
- Webservices: Met technologieën als REST (REpresentational State Transfer) kan men webservices implementeren. Men heeft REST tijdens de opleiding al eens kort benaderd, maar hier gaat men het uitgebreider behandelen.
- Language Intergrated Query (LINQ): LINQ zal voornamelijk voorkomen in de datalaag van C# projecten. Men heeft in de opleiding nog nooit met LINQ gewerkt.
- Development framework en buildserver: Dit is de eerste keer dat men met deze concepten in aanraking komt. Men heeft in de opleiding nog geen development framework of buildserver gebruikt. Tijdens de masterproef zal men hiermee niet veel in aanraking komen.