HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

Kurzfassung der Studienarbeit

Abteilung	**Informatik**
Name der Studierenden	**Thierry Wyss, Marco Busarello**
Studienjahr	**2006/2007**
Titel der Studienarbeit	**CodeML**
Examinatorin / Examinator	**Bruno Feurer**

Kurzfassung der Studienarbeit

Bei den verschiedenen objektorientierten Programmiersprachen ist das Grundkonzept sehr ähnlich. Neben verschieden Konstrukten, die zwischen den Sprachen unterschiedlich sein können, ist vor allem die Syntax anders. Es drängt sich die Frage auf, ob man nicht ein bestehendes Programm oder Teile eines Programmes in eine andere Sprache übersetzen könnte. Ein Java-Programmierer könnte beispielsweise seine eigene Implementation einer Datenstruktur problemlos einem C# Entwickler zur Verfügung stellen. Im Idealfall könnten sogar ganze Projekte Sprachunabhängig entwickelt werden. Jeder Entwickler könnte mit seiner bevorzugten Sprache arbeiten und und seine Codefragmente anschliessend in die gemeinsame Sprache übersetzen lassen.

CodeDOM ist eine Technologie von Microsoft, die es ermöglicht, die Struktur von Quelltext in einem Objektmodell abzubilden. Diese Technologie kann als eine Art Schnittstelle zwischen verschiedenen Programmiersprachen eingesetzt werden. Mittels einer solchen Schnittstelle wäre es denkbar, die oben beschriebene Vision einer Übersetzung von Source Code umzusetzen.

Das Ziel dieses Projektes ist es, ein Plug-in für die Entwicklungsumbgebung Eclipse zu entwickeln, das es ermöglicht, bestehende Java Files in das Objektmodell von CodeDOM zu übeführen und in einem XML Format abzuspeichern. Das generierte XML File soll eine Sprachunabhängige Darstellung des ursprünglichen Java Files sein und somit Programmcode einer anderen objektorientierten Sprache generieren können.

Der XML-Export von Java-Quelltext kann in die drei Schritte Parsen, Translation und Export unterteilt werden. Der gesamte Ablauf wird als Transformation bezeichnet. Eclipse parst Quelltext mit dem internen ASTParser und stellt diesen in einem Abstract Syntax Tree dar. Die Translation, also die Übersetzung des AST nach CodeDOM, ist die grosse Herausforderung dieses Projektes. Für jeden AST-Knoten muss ein Translator geschrieben werden, welcher dieses Element in ein entsprechendes Gegenstück in CodeDOM umwandelt. Der AST definiert mehr Elemente als das Objektmodell von CodeDOM, daher muss bie der Translation darauf geachtet werden, dass keine wichtigen Informationen verloren gehen. Der letzte Schritt in der Transformation ist der XML-Export. Dabei wird das CodeDOM-Objektmodell in ein XML-Dokument exportiert, welches nach dem CodeDOM-Schema aufgebaut ist. Der Export schliesst den Prozess einer Transformation ab.

Um die Qualität der Implementation zu überprüfen, wurde ein Importer für .NET implementiert, mit dessen Hilfe die generierten XML-Dateien in C#-Dateien umgewandelt werden können. Der so entstandene Code kann mit dem ursprünglichen Java-Code verglichen

und auf Fehler geprüft werden. Für ein ausgereiftes Testing wäre ein komplettes Testframework wünschenswert. Die Umsetzung eines solchen Testframeworks ist jedoch sehr aufwendig. Wir haben uns entschieden, das Augenmerk beim Testing auf die korrekte Umsetzung der Translator zu setzen. Dazu haben wir Unit Tests für die Translator geschrieben.

Für eine intuitive Bedienung des Plug-ins wurde eine Benutzeroberfläche implementiert. Die zentrale Komponente für die Benutzung des Plug-ins ist der Export Wizard. Zudem wurde eine Preferencepage implementiert, über die der Benutzer Einstellungen für den Export machen kann.

Durch die zeitliche Begrenzung dieses Projektes ist es nicht möglich gewesen, eine komplette Translation des AST nach CodeDOM zu implementieren. Es war uns daher wichtig, eine möglichst gute Architektur zu entwickeln, welche einfach ergänzt und erweitert werden kann. Eine detaillierte Dokumentation soll eine solche Weiterentwicklung ermöglichen und unterstützen.

Der erste Schritt einer Weiterentwicklung von CodeML müsste vermutlich die Vervollständigung des vorhandenen Codes sein. Dies beinhaltet vorallem die Implementation aller Translator. Danach wäre es interessant, CodeML für neue Programmiersprachen zu erweitern, wodurch sich der Anwendungsbereich erheblich vergrössern würde. Es wäre auch denkbar, dass ein eigenes Objektmodell entwickelt wird, welches analog zu CodeDOM eine Abbildung von Quelltext ermöglicht und in CodeML integriert werden kann.