Erfahrungen bei der Portierung von Delphi Legacy Code nach .NET*

Stephan Reiter, Reinhard Wolfinger
Christian Doppler Labor für Automated Software Engineering
Johannes Kepler Universität, 4040 Linz
{reiter,wolfinger}@ase.jku.at

Abstract: In diesem Artikel erläutern wir die Erfahrungen, die wir bis zum jetzigen Zeitpunkt im Zuge der Portierung einer komplexen Anwendung auf die .NET-Plattform sammeln konnten. Den Ausgangspunkt stellte dabei das mit Borland Delphi entwickelte *New Technology Commercial System* des österreichischen Unternehmens BMD Systemhaus GmbH dar. Unter Verwendung des aktuellen Delphi.NET-Compilers konnten bereits große Teile des Quellcodes an .NET angepasst werden.

1 Einleitung

Das österreichische Unternehmen BMD Systemhaus GmbH ist Hersteller von Business Software und mit seinem Produkt *New Technology Commercial System* (im Folgenden kurz: NTCS) seit langem im Sektor des Enterprise Resource Planning etabliert. Für die Zukunft ist geplant, die Software, welche mit Borland Delphi als Windows-Anwendung entwickelt wird [Kna99], auf die .NET-Plattform zu portieren. Um die Risiken, die mit diesem Vorhaben verbunden sind, zu minimieren, wird ein Pilotprojekt durchgeführt, das Entscheidungsgrundlagen für die Planung der Portierung liefern soll.

Ein weiteres Ziel des Pilotprojekts ist die Evaluierung der Verwendung eines Plugin-Frameworks, wie es am Christian Doppler Labor für Automated Software Engineering entwickelt wird [WDPM06]. Auf dessen Basis soll es Kunden ermöglicht werden, NTCS entsprechend ihren Anforderungen zur Laufzeit zu konfigurieren.

2 Ausgangsbasis und Vorgangsweise

NTCS wurde mit Borland Delphi in der Version 7 als native Anwendung für Windows entwickelt. Der Umfang des Quellcodes ist in den vergangenen Jahren auf rund 4 Mio. Zeilen gewachsen. Davon entfällt in etwa die eine Hälfte auf anwendungsspezifischen Code und die andere auf die *NTCS-Tools*, ein Framework, das von BMD für alle Anwendungen

*Diese Arbeit wurde im Rahmen des Christian Doppler Labors für Automated Software Engineering durchgeführt und von BMD Systemhaus GmbH bzw. von der Christian Doppler Forschungsgesellschaft unterstützt.

seiner Produktlinie eingesetzt wird. Für unser Pilotprojekt sind die NTCS-Tools von besonderem Interesse, da sie eine Vielzahl von unterschiedlichen Bereichen abdecken, wie z.B. den Zugriff auf Datenbanken und die Darstellung von Dialogfenstern, und so einen guten Überblick geben können, welche Probleme bei der Portierung zu erwarten sind.

Im Hinblick auf die Entwicklung von Plugins entschieden wir uns dafür, die NTCS-Tools vor der Durchführung der Portierung in kleinere Einheiten aufzuteilen. Dies hat den Vorteil, dass Abhängigkeiten eines gegebenen Plugins, das eine bestimmte Funktionalität implementiert, genauer angegeben werden können und nur die tatsächlich benötigten Teile der NTCS-Tools referenziert werden.

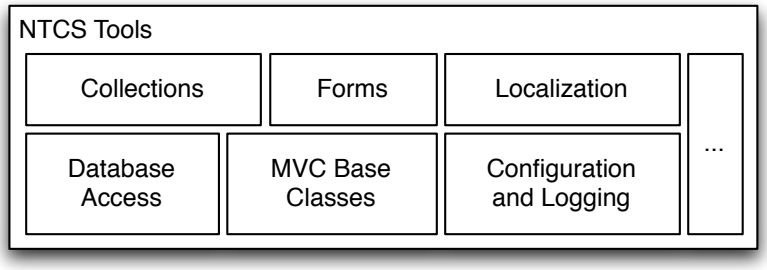

Abbildung 1: Teilbereiche der NTCS-Tools werden als separate Bibliotheken portiert

Die Zerlegung der NTCS-Tools in kleinere Einheiten hat eine Reihe weiterer Vorteile, welche unabhängig vom Einsatz eines Plugin-Frameworks sind:

- Bei der separaten Portierung von Teilen der NTCS-Tools ist es möglich, darauf zugeschnittene Tests durchzuführen. Damit kann sichergestellt werden, dass das Verhalten der Bibliothek unter .NET dem erwarteten nach wie vor entspricht.
- Zudem erlaubt diese Vorgangsweise eine bessere Dokumentation des Projektfortschritts, da der Umfang einer Teilaufgabe besser abgeschätzt werden kann. Somit sind detailliertere Vorhersagen über den weiteren Verlauf der Portierung möglich.
- Die Reorganisation des Codes in mehrere Bibliotheken ermöglicht deren getrennte Versionierung. Dadurch können Patches, die Programmfehler beheben, kleiner ausfallen, da nur mehr betroffene Bibliotheken ausgetauscht werden müssen.

3 Legacy Code und Delphi.NET

Mit dem Developer Studio 2006 bietet Borland eine Version seines Delphi-Compilers an, der die Übersetzung von Code für die .NET-Plattform in den Versionen 1.0 und 1.1 ermöglicht. Besonders von Vorteil für die Portierung bestehender Anwendungen ist die nahezu vollständige Unterstützung von unmanaged Delphi-Code. Trotzdem ist es kaum möglich, komplexen Delphi-Code ohne Anpassungen für die neue Plattform zu übersetzen.

3.1 Verwendung von Zeigern

Der Einsatz von Zeigern stellte in unserem Fall den häufigsten Grund für notwendige Codeanpassungen dar: Werden nämlich Zeiger in einer Methode verwendet, so muss diese mit dem Schlüsselwort *unsafe* gekennzeichnet werden. Zudem muss für die Übersetzung der Delphi-Unit die Compilerdirektive {`$UNSAFECODE ON`} gesetzt werden.

Da Assemblies, die unsicheren Code enthalten, nur mit höheren Rechten ausgeführt werden können, entschieden wir uns dafür, den Einsatz von Zeigern durch andere Lösungen zu ersetzen, was in den meisten Fällen auch möglich war. Ausnahmen bildeten Aufrufe von Funktionen der Windows API, die Verweise auf Objekte als Integers erwarten: Der entsprechende Parameter kann hier nur über den Umweg eines Zeigers gesetzt werden.

3.2 Inline Assembly Code

Unmanaged Delphi erlaubt die Aufnahme von Inline-Assembler in den Quellcode, was in der Regel vor allem zur Optimierung von zeitkritischen Abschnitten einer Methode eingesetzt wird. Unter .NET ist dies allerdings nicht mehr erlaubt; ein Umstand, der eine direkte Portierung des betroffenen Codes unmöglich macht, wodurch sich der Entwickler mit einer Neuimplementierung derselben Funktionalität in purem Delphi-Code konfrontiert sieht.

Einen Ausweg bieten die .NET *Platform Invocation Services* (kurz: P/Invoke) [Nat02], welche die Verwendung von nativen Bibliotheken ermöglichen. Für die Portierung der NTCS-Tools bedeutete dies, dass Prozeduren, die nur schwierig für die .NET-Plattform angepasst werden konnten, in eine separate *unmanaged DLL* übersetzt wurden und durch Wrapper im Delphi.NET-Code verwendet werden.

3.3 Strengere Typenprüfung

Im Zuge der Portierung zeigte sich wiederholt, dass Delphi bei der Überprüfung von Typenumwandlungen größere Freiheiten einräumt als Delphi.NET. Von diesem Verhalten betroffen waren vor allem Prozeduren zur Nachrichtenbehandlung, deren Signaturen von der Borland-Vorgabe abwichen: Ungültige Umwandlungen der Parametertypen waren die Folge, die sich ausschließlich zur Laufzeit durch Ausnahmefehler zeigten, da bei der Übersetzung keine Überprüfung der Signatur von *Message Procedures* durchgeführt wird.

3.4 Änderungen in Bibliotheken und Verfügbarkeit für .NET

Die öffentlichen Schnittstellen der Klassenbibliothek für Delphi.NET unterscheiden sich nur marginal von der Version für unmanaged Delphi, weshalb bei der Portierung in diesem Bereich kaum Änderungen vorgenommen werden müssen.

Kritischer für den Verlauf des Projekts stellte sich die Abhängigkeit der NTCS-Tools von diversen externen Bibliotheken heraus: Diese werden z.B. für den Zugriff auf Datenbanken benötigt oder ermöglichen die Ausführung von benutzerdefinierten Skripten. Ist es nicht möglich, eine .NET-kompatible Version einer Bibliothek zu verwenden, bieten sich unterschiedliche Möglichkeiten an, der Situation zu begegnen:

- Ist der Quellcode der Bibliothek verfügbar, so stellt dessen Portierung nach .NET eine Variante dar. Sie ist allerdings nicht zu empfehlen, wenn vom Hersteller in absehbarer Zeit eine .NET-kompatible Version zu erwarten ist.
- Die Erstellung eines Wrappers, der die Funktionen der Bibliothek für .NET-Anwendungen zugänglich macht, ist ein anderer möglicher Lösungsweg. Er hat den Vorteil, dass Wrapper im Allgemeinen mit geringerem Aufwand entwickelt werden können. Von Nachteil sind allerdings die Leistungseinbußen, die durch die vom Wrapper zusätzlich eingeführte Schicht verursacht werden.

Bei der Portierung der NTCS-Tools im Zuge des Pilotprojekts entschieden wir uns für die Entwicklung von Wrappern für jene Bibliotheken, die nicht .NET-kompatibel waren. Sollte sich dies später bezüglich der Performance als Flaschenhals herausstellen, ist die Portierung der betroffenen Bibliothek zu erwägen.

4 Fazit und Ausblick

Aufgrund der guten Kompatibilität von Delphi.NET zu unmanaged Delphi ließen sich bis zum jetzigen Zeitpunkt bereits große Teile der NTCS-Tools nach .NET portieren. Als gute Wahl stellte sich die diskutierte Vorgangsweise der Unterteilung der Codebasis in separate Einheiten heraus, da die nun für .NET verfügbaren Bibliotheken bereits in diversen Testanwendungen eingesetzt werden können.

Nach Abschluss der Portierung wollen wir noch klären, welchen Aufwand es darstellen würde, Komponenten der graphischen Benutzeroberfläche, die aktuell mit VCL.NET realisiert werden, durch deren Äquivalent in den Windows Forms zu ersetzen. Dadurch würde der Quellcode näher an die .NET-Klassenbibliothek rücken und die Integration von Anwendungsteilen, die in anderen Programmiersprachen erstellt wurden, erleichtern.

Literatur

[Kna99] Markus Knasmüller. Quo Vadis, BMD? Research Projects at BMD Steyr - An Experience Report. In *Proc. European Software Day*, Milano, Italy, September 1999.

[Nat02] A. Nathan. *.NET and COM: The Complete Interoperability Guide*. Sams, 2002.

[WDPM06] Reinhard Wolfinger, Deepak Dhungana, Herbert Prähofer und Hanspeter Mössenböck. A Component Plug-In Architecture for the .NET Platform. In *Modular Programming Languages*, Jgg. Volume 4228/2006, Seiten 287–305, 2006.