Reengineering eines Softwarepakets zur Preisberechnung und Optimierung von Finanzderivaten

Diplomarbeit

zur Erlangung des Grades eines Diplom-Ökonomen der
Wirtschaftswissenschaftlichen Fakultät der Universität Hannover

vorgelegt von

Name: [redacted] Degro Vorname: [redacted] Heiko [redacted]

Erstprüfer: Prof. Dr. Michael H. Breitner

Hannover, den 27. März 2007

Ich bedanke mich bei Herrn Prof. Dr. Michael H. Breitner für die sehr gute und geduldige Betreuung. Ich bedanke mich auch bei Herrn Simon König von der Universität Hannover und bei Herrn Oliver Kubertin.

Inhaltsverzeichnis

1. Einführung

Nicht nur das Produzieren neuer Software, sondern auch das Warten vorhandener Software gewinnt an Bedeutung. Viele Unternehmen scheuen den Umstieg auf neuere Software, weil sie unter anderem mit der vorhandenen zufrieden sind, der Umstieg auf neuere Software zu hohe Kosten verursacht oder weil sie nicht bereit sind die Risiken einzugehen, die mit einem Umstieg auf ein neue Software zusammenhängen.

Trotz dieser Erkenntnis hat das Reengineering von Software nicht die Bedeutung wie das Engineering. Allgemein akzeptierte eigenständige Vorgehensmodelle gibt es für das Software-Reengineering noch nicht. Diese Diplomarbeit befasst sich mit dem Reengineering eines vorhandenen Softwarepakets, Warrant-Pro-2, welches zur Preisberechnung und zur Optimierung von Finanzderivaten, Optionen im Besonderen, dient. Warrant-Pro-2 gibt es zur Zeit nur für das Betriebssystem Linux. Im Rahmen dieser Diplomarbeit soll Warrant-Pro-2 auf das Betriebssystem Windows migriert werden. Warrant-Pro-2 verfügt über eine graphische Benutzerschnittstelle, die durch Matlab-Elemente realisiert ist. Bei der Migration soll als Alternative zum properietären Matlab die Programmiersprache Java geprüft werden. Dabei wird großen Wert darauf gelegt, dass die hochwertige graphische Ausgabe der von Warrant-Pro-2 berechneten Daten erhalten bleibt.

Die Berechnungen und Optimierungen führt in Warrant-Pro-2 ein eigenständiger Fortran-Kern durch. Jedoch existieren acht verschiedene Kompilate dieses Kerns, die sich in ihrem Speicherbedarf unterscheiden. Der Grund dafür ist, dass der Kern mit FORTRAN-77 programmiert worden ist. Dieser Sprachstandard verfügt aber nicht über die Möglichkeit eine dynamische Speicherallokierung. Deshalb besteht eine weitere Aufgabe dieser Diplomarbeit den Kern durch Elemente des neueren Fortran-90/95-Sprachstandards um eine dynamische Speicheralokierung zu erweitern.

Ich werde diese Diplomarbeit mit mit einer Einführung über Optionen beginnen. Insbesondere werden in diesem Abschnitt die zentralen Begriffen erläutert, die im Zusammenhang mit Warrant-Pro-2 von Bedeutung sind. Darüber hinaus wird ein Überblick über die Verwendung von Optionen gegeben.

Anschließend wird erläutert, wie Software-Reengineering in der Wissenschaft gesehen wird. Fortgesetzt wird mit der Beschreibung des Spiralmodells, welches als Vorgehensmodell für diese Diplomarbeit dient. Gemäß dem Spiralmodell wird beim Reengineering die Gesamtaufgabe in verschiedene Phasen zerlegt worden. Die einzelnen Phasen dieses Software-Reengineering-

Projekts, werden ausführlich dargestellt. Diese Phasen beinhalten die grundlegenden Elemente des Spiralmodells, nämlich die Anforderungen der jeweiligen Phase, die Analyse und das Herausarbeiten etwaiger Risiken und die Validierung und abschließend die Planung für die nächste Phase.

Danach werden anhand von Beipielen demonstriert, dass die Migration vom freien Betriebssystem Linux auf das properietäte Betriebsystem erfolgreich gelungen ist. Abgeschlossen wird die Diplomarbeit mit einem Fazit und einem Ausblick.

7. Fazit und Ausblick

7.1. Fazit

Insgesamt erwies sich die Migration von Warrant-Pro-2 als kein leichtes Projekt. Die Gründe hier für waren vielschichtig. Die Schwierigkeiten lagen vor allem bei der Migration des Fortran-Kerns und der graphischen 3D-Darstellung der Ergebnissen. Beim Fortran-Kern bestand die Schwierigkeit hauptsächlich im Aufsetzen des Projekts. Dies lag zum einen an den Abhängigkeiten, die sich auf die Adifor-Bibliotheken begründen, zum anderen lag es am Intel-Compiler, der selbst neue Abhängigkeiten erzeugt. Die Umstellung auf die dynamische Speicherallokierung konnte ebenso realisiert werden wie das Eliminieren der meisten `common`-Blöcken. Nur bei der Migration der Oberfläche konnten die Ziele nicht vollständig umgesetzt werden. Die Oberfläche wird nach wie durch Matlab-Elemente erstellt. Die Portierung zu Java konnte nicht vollzogen werden, da es noch keinen adäquaten Ersatz für die interaktive 3D-Darstellung gibt.

Die Portabilität von Warrant-Pro-2 ist auf die Bestandteile bezogen (Matlab-Komponenten, C- und FORTRAN-77-Bibliotheken, FORTRAN-77- und inzwischen auch Fortran-90-Code) relativ gut. Die einzelnen Bestandteile sind auf eine Vielzahl von Plattformen portierbar. Die Portabilität wird jedoch durch die Abhängigkeit untereinander sowie durch teilweise Ausrichtung des Quellcodes auf bestimmte Compiler eingeschränkt.

Vom Blickwinkel der Quelltexte ist die Wartbarkeit von Warrant-Pro-2 als gut einzustufen. Die vorliegenden Quelltexte sind gut strukturiert und kommentiert. Es gelang mir relativ schnell mich in den Quelltexten zurecht zu finden.

Die Tests zeigten insgesamt sehr gute Ergebnisse. Sie bestätigten insbesondere die bekannten guten Resultate der Linux-Version. Die Tests zeigten jedoch auch, dass die Oberfläche noch Verbesserungspotenzial hat.

Die Beispiele lieferten keine eindeutige Aussage. Im ersten Beispiel wurde das gute Ergebnis der Vorgängerversion bestätigt, während im zweiten Beispiel das Ergebnis nicht zufriedenstellend war. Die Ursache konnte bis zum Ende der Bearbeitungszeit nicht gefunden werden. Vielleicht liegt der Fehler in der Modellierung des Beispiels. Aber ein Programmfehler kann ebenfalls nicht ausgeschlossen werden.

7.2. Ausblick

Ungeachtet der Schwierigkeiten, die bei Migration auftraten, erbringt Warrant-Pro-2 jetzt die gleichen Ergebnissen wie vor der Migration und vor der Erweiterung um die dynamische Speicherallokierung.

Aber Warrant-Pro-2 bedarf noch einiger Änderungen. Auf eine Wiederholung der Anregungen aus der Diplomarbeit von Herrn KUBERTIN[1] wird verzichtet. Es sollte an dieser Stelle jedoch nicht unerwähnt bleiben, dass einige seiner Anregungen, ebenfalls noch nicht realisiert sind. Die Portierbarkeit hat noch nicht den Stand wie er wünschenswert wäre. Um diese zu verbessern müsste m.E. die komplexen und starken Abhängigkeiten der einzelnen Programme bzw. Programmelemente abgebaut werden. So könnte die Oberfläche in einer anderen Programmiersprache als dem properietären Matlab-C realisiert werden. Die Chancen könnten in der Zukunft besser sein als ich sie antraf. Das von mir kritisierte `gnuplot` wird beständigt weiterentwickelt und könnte irgendwann die benötigte Reife und Qualität haben. Auch VTK oder das Octave-Projekt, ein freier Matlab-Clon, könnten irgendwann dafür geeignet sein, eine gleichwertige 3D-Plots zu erstellen.

Einiges hängt aber auch davon, womit die zukünftige Oberfläche erstellt wird. Empfehlenswert ist es jedoch in dieser auf plattformspezifische Kommandos zu verzichten. Das Löschen, Verschieben oder Umbenennen von Dateien kann in vielen Programmiersprachen mit vorhandenen Funktionen erreicht werden. Aber auch Pfadangaben können evtl. plattformunabhängig dargestellt werden. So gibt es in Java einen neutralen Seperator und je nach dem auf welcher Plattform das Programm ausführt wird, wird entweder ein Slash oder ein Backslash verwendet.

Um den Fortran-Kern auch in Zukunft weiterentwickeln zu können, wird man über Kurz oder Lang nicht umhin kommen auch die externen Bibliotheken neu zu übersetzen. Hier könnten aufgrund der geschilderten Multithread-Problematik neue Schwierigkeiten enstehen. Vielleicht ist es deshalb überlegenswert gemeinsam mit den Adifor-Projekt und Npsol-Projekt auf einem neueren Standard basierende Bibliotheken zu erstellen, die dann auch mit neueren Entwicklungsumgebungen kompatibel sind.

Die Wartbarkeit aus Quellcodesicht könnte verbessert werden. So könnte man den Fortran-Kern und, falls weiter Matlab eingesetzt wird, auch die Oberfläche Refactoring-Maßnahmen unterziehen. Die dynamische Speicherallokierung ist zwar technisch realisiert, aber es gibt jetzt einige überflüssige Parameterübergaben in Funktionen oder Subroutinen, wie z.B. Arrays die jetzt global definiert sind. Dieses habe ich aus zeitlichen Gründen nicht mehr geschafft. Eine andere Maßnahme könnte sein, der einen oder anderen Variablen einen sprechenderen Namen zu geben. Vielleicht kann man sich unter `anzpar` noch vorstellen was in dieser Variablen abgespeichert wird.

[1] Siehe Kubertin [2003] S. 47f.

Aber zu erkennen, was in der Variable `V` abgespeichert wird ist nicht so offensichtlich. In den Programmquellen von der Oberfläche tauchen auch wenig sprechende Namen auf. Beispielsweise wurden viele Textfelder `edit`*N* benannt, wobei *N* für eine ein- bis dreistellige Zahl steht. Ebenso gibt es in der Oberfläche wiederkehrender Code, z.B. für das Setzen eines Parameterwertes in den Parameterdateien. Hier wäre es sinnvoll diesen in einer Funktion auszulagern, so dass diese Funktionalität nur noch an einer Stelle gepflegt werden zu braucht.

Die Benutzerfreundlichkeit könnte, wie im Abschnitt mit den Testfällen bereits ausgeführt, noch verbessert werden. Darüber hinaus könnte man für bestimmte Standardaufgaben dem Anwender einen Assistenten *(Wizard)* als Hilfe zur Seite zu stellen.