19 Erfahrungen bei der Entwicklung von Werkzeugen zum Reverse Engineering

Uwe Kaiser
pro et con, Innovative Informatikanwendungen GmbH, Annaberger Straße 240, 09125 Chemnitz
Uwe.Kaiser@proetcon.de

Seit 1994 entwickelt ”pro et con” Software-Werkzeuge für das Reverse Engineering und die Migration von kommerziellen Programmsourcen. Dieser Beitrag vermittelt einen Überblick über die Werkzeuge und stellt resultierende Erfahrungen ihrer Entwicklung zur Diskussion.

19.1 Compiler front-end’s

Für eine Reihe von Programmiersprachen wurden front-end’s entwickelt. Der Begriff ”front-end” besagt, daß der jeweilige Preprozessor der Programmiersprache sowie die syntaktische und die statisch semantische Analyse realisiert wurden, und daß die Abbildung in einen internen, abstrakten Syntaxbaum erfolgt. Die Liste existierender front-end’s besteht aus:

- PL/I,
- COBOL,
- NATURAL,
- TAL (Tandem Application Language),
- SQL,
- C,
- Java.

Im Beitrag kann nur ein Überblick vermittelt werden, Diskussionen von Details sprengen seinen Rahmen. Aus diesem Grund folgen zu den einzelnen front-end’s nur kurze

Anmerkungen:

- Sie laufen auf den Betriebssystemen Windows (NT, 2000, XP) und UNIX (Linux, Solaris).
- Verschiedene front-end's laufen ebenso auf den Host-Rechnern, auf denen die jeweilige Sprache implementiert ist, z.B. PL/I und COBOL auf MVS, COBOL und TAL auf NonStop Servern von HP.
- Für COBOL und PL/I auf MVS werden Panvalet und Librarian unterstützt, embedded SQL, CICS und IMS werden analysiert.
- Die COBOL-Analyse läßt sich für verschiedene Dialekte skalieren (COBOL II, OS/VS COBOL, NonStopCOBOL 85, ScreenCOBOL).
- Die Entwicklung eines front-end's für SQL war notwendig, weil verschiedene Sprachen (PL/I, COBOL, NATURAL,..) eine Schnittstelle zu SQL besitzen (embedded SQL) und die SQL-Abschnitte in den Sourcen ebenfalls analysiert werden.
- Wenn in den Aufzählungen realisierter front-end's C aufgeführt ist, C++ aber fehlt, dann liegt die Ursache darin, daß bisher der hohe Entwicklungsaufwand für C++ gescheut wurde (z.B. bei der Realisierung von Templates).
- Das TAL front-end entstand im Zuge eines Projektes, bei welchem ein kommerzielles Softwarepaket von TAL nach C migriert wurde.
- Das Java front-end unterstützt Java, Version 1.4. Das Werkzeug ist UNICODE-fähig.

19.2 Entwicklungserfahrungen

Aus der Entwicklung der front-end's resultieren eine Reihe von Erfahrungen, welche nachfolgend zusammengefaßt werden. Insbesondere interessieren Unterschiede zwischen der Analyse bei Compilern und bei Reengineering-Werkzeugen.

1. Für Reengineering und Migration ist es wesentlich, daß keine Informationen der Source während des Analyseprozesses verlorengehen. Dazu gehören unter anderem Informationen zu Preprozessoranweisungen und Kommentaren. Die Realisierung erfolgt durch die Aufnahme dieser Informationen in die Tokenliste, d.h., Preprozessoranweisungen und Kommentare bilden eigene Token im Tokenstrom.
2. Für Reverse Engineering und Migration ist das genaue Führen der Sourcecodeposition von Objekten wichtig. Die Sourcecodeposition wird zur Komponente des Wertestroms des Scanners. Eine konkrete Anwendung besteht z.B. darin, beim COBOL-Reengineering die Deklarationsstelle eines Datenobjektes zu dokumentieren (In welcher COPY-Strecke und dort in welcher Zeile und Spalte ist das Datenobjekt lokalisiert ?).
3. Reengineering von komplexen Programmen geht über die Analyse von Sourcecode hinaus. Bekanntes Beispiel sind Java-Programme, für deren Analyse neben verschiedenen in der Import-Liste aufgeführten Java-Sourcen ebenso die Analyse von ".class"-Files bzw. "rt.jar"-Files notwendig ist. Ein alternatives Beispiel ist NATURAL, hier werden Datendefinitionen in spezieller Notation in sogenannten "DATA-AREA"-Files abgelegt, auf die im Sourceprogramm Bezug genommen wird. Als Konsequenz für die Programmanalyse sind spezielle Analysatoren zu entwickeln, z.B. bei Java für das Lesen von ".class"-, bzw. "rt.jar"-Files, bei NATURAL für das Aggregieren der Informationen aus den "DATA-AREA"-Files.
4. front-end's müssen eine gewisse Robustheit gegenüber fehlerhaften bzw. unvollständigen Sourcen besitzen. Sie müssen auch dann noch Informationen liefern. Praktische Anwendungen sind fehlende COPY-Books bei der Analyse oder die Kundenanforderung, separate COPY-Books von COBOL analysieren zu können.
5. Der Arbeitsaufwand für die Entwicklung o.g. front-end's wurde quantifiziert. Für den Entwicklungsprozeß ausgehend von der Spezifikation bis zur GA-Version und nach Durchlaufen des gesamten QA-Prozesses ergibt sich ein durchschnittlicher Zeitrahmen von ca. 18 Monaten und ein durchschnittlicher Aufwand von ca. 3.5 Entwicklungsjahren je front-end.

19.3 Tools und Metatools

Parallel zur Entwicklung der front-end's entstanden eine Reihe von Meta-Tools, welche der Produktion von Reengineering-Werkzeugen dienen. Alle o.g. front-end's wurden mit einem eigenentwickelten Parsergenerator realisiert. Sein Name "Backtracking Compiler Compiler" (BTRACC) verweist auf das zugrundeliegende Grammatikanalyseverfahren der generierten Parser. In der Theorie wird auf die schlechtere Performance des Backtracking-Verfahrens gegenüber anderen Analyseverfahren verwiesen. Praktische Erfahrungen zeigen jedoch, daß diese zu vernachlässigen ist. Durchgeführte Tests ergaben, daß auf einem mit 1.2 GHz getakteten Pentium z.B. eine C-Source von 13.000 LOC in 1.2 Sekunden mit diesem Verfahren analysiert wird. Das Backtracking-Verfahren besitzt den Vorteil, daß Grammatikbeschreibungen von Programmiersprachen, welche in Dokumentationen vorkommen und überwiegend pragmatisch entstanden, keiner aufwendigen Umstellung von Grammatikregeln bzw. Konfliktbereinigung bedürfen, um von BTRACC akzeptiert zu werden.

Die front-end's sind Bestandteil von Werkzeugen für das Reverse Engineering oder sie werden in Form von Service bei Migrationsprojekten eingesetzt. Ein kommerzielles Tool für das Reverse Engineering und die Redokumentation von Programmsourcen ist "Flow Graph Manipulator´´ (FGM).

19.4 Aktuelle Entwicklungen

Aktuelle Entwicklungen betreffen die Weiterentwicklung des Tools FGM:

- Es werden Steuerflußinformationen in Form von Programmablaufplänen entwickelt.
- Ein NATURAL front-end wird in das Werkzeug integriert.

Über die feingranulare Analyse auf der Ebene eines Programmes hinaus soll Applikationswissen aufgebaut werden. Ein Sourceprogramm existiert nicht unabhängig von seiner Umwelt, sondern es existieren Schnittstellen (Datenbanken, Files, Transaktionsmonitore). In FGM soll aus diesem Grund eine SQL-Datenbasis eingebunden werden, welche dieses Applikationswissen zukünftig verwaltet und dem Nutzer durch Anfragen zur Verfügung stellt.